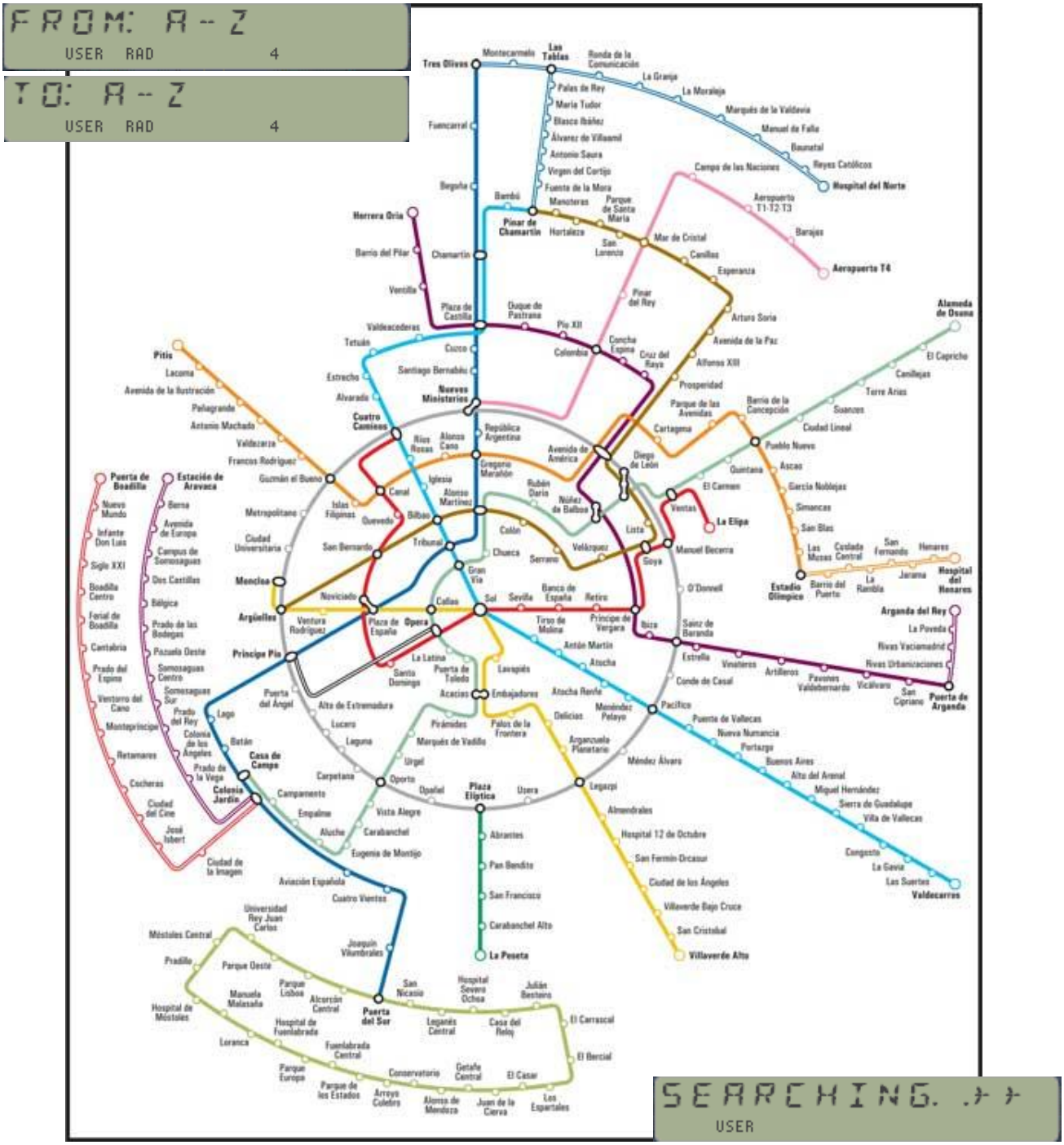
Path-finding on Subway Networks

City Metro Maps for the HP-41



Written and programmed by Ángel M. Martin

Revision 1E – February 2018

This compilation revision 1.3.1

Screen captures taken from V41, Windows-based emulator developed by Warren Furlow.
See www.hp41.org

Acknowledgments.-

Thanks to Greg J. McClure who lent his ear and provided advise and suggestions during the preparation of this project.

Thanks to Jean-Marc Baillard who contributed with the Maze Generator and Traveling Salesman programs.

City Metro Maps for the HP-41

Table of Contents.

The table below shows the function names in alphabetical order with a brief description. Note that besides the Metro Lines there are a few other functions related to the Traveling Salesman Problem and Mazes.

XROM	Function	Description	Input Parameters	Author
30,00	-METROPATHS	Section Header	*n/a*	*Ángel Martin*
30,01	^PATH _	Searches for Path	Prompts FROM:/TO:	*Ángel Martin*
30,02	ALINE _ _ _	Shows line alphabetical	Line# in prompt	*Ángel Martin*
30.03	ANYONE	Random station Pick	None	*Ángel Martin*
30,04	CHANGE	Changes Line	Handle in X	*Ángel Martin*
30,05	DOUBLE	Shows Double Stations	None	*Ángel Martin*
30,06	DRCT?	Checks direct connect	From/To in Y,X	*Ángel Martin*
30,07	FPATH	Shows Full Path	Path handles in stack	*Ángel Martin*
30,08	LEVEL _	Level Launcher	Prompts “0:1:2:3:X:^”	*Ángel Martin*
30,09	LEVEL0	Tries Llevel-0 connect	From/To in Y,X	*Ángel Martin*
30,10	LEVEL1	Tries Level-1 connect	From/To in Y,X	*Ángel Martin*
30,11	“LEVEL2”	Tries Level-2 connect	From/To in Y,X	*Ángel Martin*
30,12	“LEVEL3”	Tries Level-3 connect	From/To in Y,X	*Ángel Martin*
30,13	LHEAD _ _ _	Line Head	Station# in prompt	*Ángel Martin*
30,14	LTAIL _ _ _	Line Tail	Station# in prompt	*Ángel Martin*
30,15	NEARX	Nearest X-fer point	Handle in X	*Ángel Martin*
30,16	NEXTX	Next Station	Handle in X	*Ángel Martin*
30,17	PREVX	Previous Station	Handle in X	*Ángel Martin*
30,18	PLENG	Path Length	Path handles in stack	*Ángel Martin*
30,19	SINGLE	Shows Single Stations	none	*Ángel Martin*
30,20	STNAME _ _ _	Station Name	Station# in prompt	*Ángel Martin*
30,21	STINFO _ _ _	Station Information	Station# in prompt	*Ángel Martin*
30,22	TLINE _ _ _	Shows line in travel order	Line# in prompts	*Ángel Martin*
30,23	TRIPLE	Shows Triple/Quad Stations	None	*Ángel Martin*
30,24	XFER _	X-fer Launcher	Prompts “+:-:*:C:H:T:?”	*Ángel Martin*
30,25	XFER?	Checks for transfer type	Station# in X	*Ángel Martin*
30,26	*“XPLORE”*	Whimsical Explorer	From-To in Y,X	*Ángel Martin*
30,27	XTRAIL	Shows Transfer Stations	Path handles in stack	*Ángel Martin*
30,28	*“*L2”*	Auxiliary routine	Subroutine use	*Ángel Martin*
30,29	?MAP	Checks CityMap existence	None	*Ángel Martin*
30,30	-A_MAZING	Checks for Library#4	n/a	*Ángel Martin*
30,31	ANUMDL	ANUM w/ Deletion	Text in ALPHA	*hp Co.*
30,32	AREV	ALPHA Reverse	Text in APLHA	*Fran DeVries*
30,33	DTOA	LCD to ALPHA	String in LCD	*Ángel Martin*
30,34	RNG	Random Number w/ Timer	Current Time	*JM Baillard*
30,35	*“ASHFL”*	ALPHA Shuffle	Text in ALPHA	*Harald Schumny*
30,36	*“MAZE”*	Maze Generator	Data in Stack/Regs	*JM Baillard*
30,37	*“MAZE+”*	Driver for MAZE	Prompts for data	*Ángel Martin*

XROM	Function	Description	Input Parameters	Author
30,38	*"TS2"*	TSP Bi-dimensional	Data in Registers	*JM Baillard*
30,39	*"TS2+"*	Driver for TS2	Prompts for Data	*Ángel Martin*
30,40	*"TS3"*	TSP Tri-dimensional	Data in registers	*JM Baillard*
30,41	*"TS3+"*	Driver for TS3	Prompts for data	*Ángel Martin*
30,42	*"TSS"*	TSP Spherical	Data in registers	*JM Baillard*
30.43	*"TSS+"*	Driver for TSS	Prompts for data	*Ángel Martin*
30.44	*"TS*"*	Auxiliary routine	Subroutine use	*JM Baillard*

NYC Subway version.

The New York Subway module has a slightly different FAT – Because of the very large sizes of the network tables it is an 8k module that does not use the common Metro Lines engine, but its own one. The Line functions prompt for alphanumeric values instead of numeric, and there is one additional auxiliary function instead of all the TSP and MAZE sections in the FAT.

30,30	-A_MAZING	Checks for Library#4	n/a	*Ángel Martin*
30,31	ANUMDL	ANUM w/ Deletion	Text in ALPHA	*hp Co.*
30,32	DTOA	LCD to ALPHA	String in LCD	*Ángel Martin*
30,33	NYC# _	Codes Line#	Line value in prompt	*Ángel Martin*
30,34	RNG	Random Number w/ Timer	Current Time	*JM Baillard*

You're not supposed to have both versions plugged in the calculator at the same time, make sure the common MetroPaths is removed when you're using the NYC model. The NYC Subway is an independent 8k Module, whereas the others are individual 4k modules than share the same search engine – and are expected to be plugged right above it. Here's a table showing how the different networks work:

City	Search Engine	Network Data
Madrid	Metro Paths	Metro Madrid
London		London Tube
Paris		Paris Metro
Berlin		Berlin U-Bahn
NYC	NYC Subway	

Module Dependencies.

The MetroPaths and NYC Subway modules check for the presence of its dependencies, i.e. the Library#4 and the CX O/S. Note that if the Library#4 module is not plugged in the calculator, a warning message is shown every time the calculator is switched on -- halting the polling points process to avoid likely problems.

You should only use the module functions with the Library#4 module plugged in.

City Metro Maps for the HP-41

Introduction.

Path-finding is a distinct subject within the different math applications to real life problems. This module is but a humble first-contact approach to the subject applied to subway networks, with the Madrid, Paris and London examples of implementation.

Don't expect to find recursive algorithms using sophisticated backtracking, priority queues or cost-weighing methods for path selection optimization within the network. Considering the odds, it's quite remarkable that just one path can be found using the tools and resources at hand - if you have ever programmed in MCODE you'll know what I mean. Thus the functions in the module don't use any of the popular references in the literature (Traveling Salesman, Dijkstra's, et al.), but a set of tricks and routines built from-the-scratch, creating dedicated tools and functions as they were needed with the sole objective to connect the start and end stations.

Even this modest goal was ambitious considering the starting point: I had no previous experience whatsoever in this type of applications, further constrained to the utilization of MCODE as the main vehicle for the implementation – aided by a few short FOCAL routines for the more complicated cases. All things considered, the final result included in the module is nothing short of amazing (if I may say it).

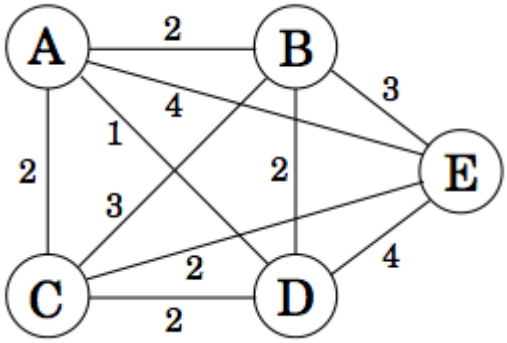
The first implementation used the Madrid metro network as first example (all those rides of my youth had to be honored). The network data takes a complete 4k block, located in the upper page of the module. This one could be swapped with other "maps", so the concept can be extended to other subway networks from other cities, provided that the network is digitized in the format expected by the path-finding routines. This is not difficult but a tedious job, so volunteers are welcome ;-)

Hope you enjoy the rides as much as I did putting these modules together.

These are the restrictions for the network characteristics:

Max. Number of lines: 15
Max. Station multiplicity: 4 lines
Max. Station name length: 12 characters
Max. Data storage: 4,050 bytes

No support for "spurs" – which need to be dealt with as independent lines or as "loops" within a given line.



Describing the Network Lines.

Two functions are available to list the line stations, either in alphabetical order (**ALINE**), or as travelled sequence (**TLINE**) from beginning to end of line. The enumeration will be halted while any key (other than R/Sand ON) is pressed. Once stopped press [<-] to clear the LCD.

The input for these functions is the line number, from 1 to 15. Any other value will trigger the "NONEXISTING" error message.

In the Madrid metro there are 15 lines, with the following start/end stations: (see appendix 2 for the complete list of stations)

Line#	Start	End	Line#	Start	End
[1]	Pinar Chamartín	Valdecarros	[9]	Paco de Lucía	Arganda del Rey
[2]	Las Rosas	Cuatro Caminos	[10]	H. Infanta Sofía	Puerta del Sur
[3]	Villaverde Alto	Moncloa	[11]	Plaza Elíptica	La Fortuna
[4]	Argüelles	Pinar Chamartín	[12]	Metrosur	
[5]	Alda. De Osuna	Casa de Campo	13 (ML1)	Pinar Chamartín	Las Tablas
[6]	Circular		14 (ML2)	Colonia Jardín	Est. de Aravaca
[7]	H. de Henares	Pitis	15 (ML3)	Colonia Jardín	Pta. de Boadilla
[8]	NuevosMinistrs.	Aeropuerto	Radial	Ópera	Ppe. Pío

For the circular lines (#6 and #12 in this case) an arbitrary start/end combination has been established. This has relevance for the search, as the algorithms will use it to determine a change of direction in the tentative targets used in the different connection trials.

Functions **SINGLE**, **DOUBLE**, and **TRIPLE** will show an alphabetically ordered enumeration of the stations with multiplicity number 1, 2, and 3 respectively. The listing can be also halted while any key other than Back Arrow and ON are pressed. Press [<-] to abort.

Finally, when it comes to the individual stations, two functions are available to show the station name and miscellaneous information: **STNAME** and **STINFO**. These functions use the station parameter as input; either at the prompt if used interactively or in the X- registers if used in a program. Entering a zero value in the prompt (in manual mode) will take the number in X instead as a shortcut. The range for these inputs goes from 1 to 275. Any other value will trigger the "NONEXISTING" error message.

The example shown below shows the information for the station "Avenida de América", using its main line (#4) parameter = 082

Note that the line numbers in the "X:" field are shown in hexadecimal (i.e. from 1 to F). This was needed to allow all possible combinations fit in the 12-character LCD.

Station Navigation functions.

These functions provide independent access points for some of the code routines used in the more complex connecting functions. Some were specifically written to facilitate the job of the FOCAL programs **LEVEL2** and **LEVEL3**. They expedite the execution of the programs substantially, and make the code much easier to understand, enhance and maintain.

- **XFER?** and **DRCT?** are test functions that check for multiple stations and whether two stations are directly connected (on the same line). They follow the standard HP-41 rule showing YES/NO and skipping a program line if the result is false.

- **LHEAD** and **LTAIL** retrieve the beginning or the end station respectively of the line the input station belongs to. The input station parameter can be entered at the prompt in manual mode, or taken from the X-register in a running program. The result station parameter is entered in the X-register, and the stack is lifted.

- **PLENG** shows the full path length for a set of transfer stations stored in the stack and Alpha registers. Refer to the table in page 8 for the expected register configurations. The information is presented in the display as shown below – for a route between stations 136 and 228 – i.e. from "Pacífico" to "La_Granja":

- **PREVX** and **NEXTX** locate the previous and next transfer station from a given station. The references are relative to the direction of travel, or in this case to the direction given to the lines with a beginning and an end. When the result is found the user flag 01 is set to indicate success. The boundary conditions for these kick-in at the beginning or end of line – where there's not a previous or following solution possible. In that case the functions will leave the user flag 01 cleared.

- **NEARX** is a condensed version of the above two, as will be content with finding the nearest transfer station on either direction, taken from the given station parameter in the X-register. Currently this function is only used in **XPLORE**, more about that to follow. Here too we can have boundary conditions, if the line is just a single spur with a unique connection to other line. User flag 01 is clear if no near transfer station is found, and set if successful.

- **CHANGE** is the digital equivalent to changing lines at a transfer station. The parameter in the X-register is taken as input, and is replaced by the number in the transferred line. The old parameter is stored in the LastX register L(4). Using it in single stations will trigger the "NO X-FER" error message and stop the program execution.

This is what you'd want to do when getting in a cul-de-sac situation within the line – change to another and continue to ride along towards your goal…

Connecting Functions.{ LEVELn }

There are several functions in the module to help you make a connection between two stations, the "From:" station **A**, and the "To:" station **B**. These are classified in increasing complexity order as **LEVEL0**, **LEVEL1**, **LEVEL2**, and **LEVEL3** – where the index refers to the number of transfers required to establish the connection.

- So for instance, if **A** and **B** are in the same line (no transfers are needed) they will be successfully connected by **LEVEL0**. The complete station list will be shown, known as the "full path" using the function **FPATH** (the execution is transferred automatically there).

- If **A** & **B** are in intersecting lines, then LEVEL0 won't be able to make the connection and you'll need to use **LEVEL1**. The transfer station between them is called **X1** – and even if there can be multiple of these, the function will report the first one found meeting the criteria. This may not be the most convenient - let alone the optimal combination – but at least will take you there. The transfer trails in this case will be: A – X1 – B.

- If two transfers are required to complete the route (**X1**, and **X2**), then LEVEL1 won't suffice and you'll have to use **LEVEL2**. Note that LEVEL2 is a FOCAL program that uses repeated iterations of the LEVEL1 function with different trial targets, moving the destination station **B** a long its main line. The transfer trails here will be: A – X1 – X2 – B

- Finally, if three transfers are required (**X1**, **X2**, **X3**) then you must use **LEVEL3** to get a valid connection. This case represents less than 5% of the cases on the network but because of the iterative process (calling LEVEL2 as a subroutine repeated times) it may take much longer to determine the solution. The transfer trails now are: A – X1 – X2 – X3 – B

Both **LEVEL0** and **LEVEL1** behave as test functions in a program, i.e. the execution will skip one program step if the connection isn't successfully established.

Note also that each of these functions can also be used in cases of less number of transfers, i.e. **LEVEL2** will also find a one-transfer path or direct connections; and **LEVEL1** can also be used for direct connections. Obviously **LEVEL3** is therefore the most powerful function since it'll find any connection involving up to three transfers.

Conversely, if a simpler solution is found (i.e. one with less number of transfers) then the functions won't continue the search for a higher-level solution. This means *the priority criteria is to minimize the number of transfers*. However, as any savvy metro user knows, sometimes an additional transfer reduces the total travel time and number of stations – but that level of expertise is not included in the algorithms at this point.

How Optimized are the Results?

Believe it or not, there's some intelligence programmed into the **LEVEL1** function – the real cornerstone of all connecting functions. Here's a short description of the planned-for contingencies.

Moving along the starting line.

If two given lines intersect twice (making a local loop) the branch taken may not be the one with fewer number of stations. In the initial version the algorithms operated on a first-found basis, whereby the first line intersection found was taken as the X-fer solution. This is good to reduce the execution time, but doesn't always result in the shortest path, as it depends on the relative position of the From: station with respect to the intersected line: we could miss a nearby x-fer "behind" as we start to move "ahead" along the line in the opposite direction.

In revision 1B the algorithm has been changed to always look for a second line intersection (moving backwards), even if a first one has been found in the forward direction, which is always tried first. When two paths are found a comparison of the branch lengths(measured in number of stations) between the two paths is made, and the shortest one is chosen.

Starting from a multiple Station.

If the From: station is multiple, the search will start moving along the lower-numbered line first. Then the above method will be applied looking for a connecting path, but if none is found then the algorithm will change lines and start scanning the next line (with the higher number). This is repeated for the third and the fourth lines if no path is found in the previous two or three.

Only when no direct connection can be established from all of the lines the FROM: station belongs to then the function will declare defeat. It's time to move up to LEVEL2 to try for other methods…

Moving the start station along the destination line.

LEVEL2 's operation is based on moving the destination station along its main line, positioning the "current" station in the transfer points X1, and using **LEVEL1** to try establish a 1-Xfer connection between the From: station and X1. If not possible, the process continues with the next transfer station on the line, X2, and checks again. This will continue until all X-fer points have been tried, in either direction of the destination line.

Move and Change until they connect.

If this process doesn't yield a positive connection then it's time for **LEVEL3**'s tactics, which involve a combination of moving to the next transfer point *and* changing lines to one still not tried. Then move back and forth along it, looking for hits – which at this points are bound to occur, but if not the algorithm will change lines again at the last transfer point found on the secondary line, getting into a third line and repeating itself until the 1-Xfer condition is eventually met.

Obviously here we need to keep track of the intermediate X-fer points to be able to reconstruct the complete path once the connection has been established. We use data registers R00 to R04 for this purpose in this case - but no data registers are used for all previous levels.

A few Examples.

The following examples illustrate the points made before on the branch selection and operation of the functions. Let's use **LEVEL1** to find connecting paths from the "Méndez Álvaro" and "Conde de Casal" Stations to "Atocha" in line#1.See the snapshot below as reference for this section of the network.

The input data are the Station Names using ^PATH, or the station parameters directly which can be obtained from the tables in appendix 2:Méndez Álvaro = 139; Conde de Casal = 141; Atocha = 18


```
Printer
 139.000 ENTER^
 18.000 LEVEL
LEVEL1
ME.NDZ_A.LVARO
PACI.FICO
PACI.FICO
MENDZ_PELAYO
ATOCHA_RENFE
ATOCHA
P=04 ST
```

```
Printer
 141.000 ENTER^
 18.000 LEVEL
LEVEL1
CONDE_CASAL
PACI.FICO
PACI.FICO
MENDZ_PELAYO
ATOCHA_RENFE
ATOCHA
P=04 ST
```

Armed with that information we can call **LEVEL1** directly for a faster operation, as seen in the screen dumps above on the right, taken from the ILPer window – and showing that the algorithms find the best approach in either forwards or backwards situation.

LEVEL2 Example.

Establish the connecting path between the stations "Pinar del Rey" (#190, in Line#8) and "Parque Avenidas" (#170 in line#7). The screens on the right show both the station listsand the transfer trails for details.

```
Printer
 190.000 ENTER^
 170.000 LEVEL
XROM "LEVEL2"
PINAR_DL_REY
COLOMBIA
COLOMBIA
CONCH_ESPINA
CRUZ_DL_RAYO
AV_AME.RICA
AV_AME.RICA
CARTAGENA
PQUE_AVNIDAS
P=06 ST
```

```
Printer
XTRAIL
L:08 #:06 X:
PINAR_DL_REY
L:08 #:07 X:9
COLOMBIA
L:09 #:09 X:8
COLOMBIA
L:09 #:12 X:4,6,7
AV_AME.RICA
L:07 #:18 X:4,6,9
AV_AME.RICA
L:07 #:16 X:
PQUE_AVNIDAS
```

Going for a longer ride now, let's find the connecting path between stations "Dos Castillas" (#305 on line 14) and "Ant. Machado" (#180 on line 7). The full result is listed below.

```
305.000 ENTER^
180.000 LEVEL
XROM "LEVEL2"
DOS_CASTLLAS
BE.LGICA
POZUELO_O
SOMOSAGUAS_C
SOMOSAGUAS_S
PRADO_DL_REY
COL_A.NGELES
PRADO_D_VEGA
COL_JARDI.N
COL_JARDI.N
CASA_D_CAMPO
BATA.N
LAGO
PRINCIPE_PI.O
PZA_D_ESPAN,A
TRIBUNAL
ALNS_MARTNEZ
GRIO_MARAN,O.N
GRIO_MARAN,O.N
ALONSO_CANO
CANAL
IS_FILIPINAS
GUZMA.N_BUENO
FR_RODRI.GUEZ
VALDEZARZA
ANT_MACHADO
P=23 ST
```

LEVEL3 Example.

As already mentioned these account for a small fraction of all possible combinations, yet show up in a few cases that may be relevant to the metro dweller. Mostly arise when going from and to isolated lines, such as line#12 (Metro Sur) and line #11 – which could be considered as a spur of the Circular line#6.

Find a path between stations "La Fortuna" (#259, at the end of line #11) and "El Bercial" (#281 on line#12). Using LEVEL3 for this yields the following list, with 43 stations and three transfer points:

```
259.000 ENTER^
 281.000 LEVEL
XROM "LEVEL3"
LA_FORTUNA
LA_PESETA
CRAB_ALTO
S_FRANCISCO
PAN_BENDITO
ABRANTES
PZA_ELI.PTICA
PZA_ELI.PTICA
OPAN,EL
OPORTO
CARPETANA
LAGUNA
LUCERO
ALTO_EXTRMDR
PTA_DL_A.NGEL
PRINCIPE_PI.O
PRINCIPE_PI.O
LAGO
BATA.N
CASA_D_CAMPO
COL_JARDI.N
AVIACIO.N_ESP
CUATRO_VNTOS
J_VILUMBRLES
PTA_DEL_SUR
PTA_DEL_SUR
PQUE_LISBOA
ALCORCO.N_C
PARQUE_OESTE
UREY_JCARLOS
MO.STOLES_C
PRADILLO
H_D_MO.STOLES
MLA_MALASAN,A
LORANCA
H_FUENLBRADA
PQUE_EUROPA
FUENLABRADA
PQUE_ESTADOS
ARYO_CULEBRO
CONSERVTORIO
ALNS_MENDOZA
GETAFE_CTL
J_DLA_CIERVA
EL_CASAR
L_ESPARTALES
EL_BERCIAL
P=43 ST
```

By now you have surely realized how difficult is to travel blind, and how much information is contained in your humble metro map folded up into your pocket –better hold on to it ;-)

Consolidated Launchers { LEVEL_ , XFER }

The module includes two function launchers that consolidate related functions into a single, convenient access point. Both launchers are interconnected and can be toggled using the [SHIFT] key.

First a **LEVEL** function launcher groups the five connecting cases into a selection prompt for your convenience. This prompt will also be presented after the station selection A, B is made using the input function **^PATH**.

The second launcher is **XFER**, which groups the station movements functions: One station up, one station down, End of line, Beginning of line, Change Line, and Map information.).

The table below shows all functions included in the launchers, and their hot-keys:

Key	LEVEL 0:1:2:3;X:^	Key	XFER +:-:*:C:H:T:?
[1]	LEVEL0	[+]	NEXTX
[2]	LEVEL1	[-]	PREVX
[3]	LEVEL2	[*]	NEARX
[4]	LEVEL3	[C]	CHANGE
[X]	XPLORE	[H]	LHEAD
[ENTER]	^PATH	[T]	LTAIL
-	-	[?]	?MAP
[SHIFT]	XFER _	[SHIFT]	LEVEL _
[ALPHA]	STNAME	[ALPHA]	STNAME
[PRGM]	STINFO	[PRGM]	STINFO

Note that the input for the functions in this launcher are the station handle, whilst for the former launcher it was the pair of handles for the From: and To: stations wanting to connect.

Other options at this point are provided by the ALPHA and PRGM keys, which will invoke the functions **STNAME** and **STINFO** respectively. With them you can retrieve the station name and station information, a single screen showing the main line, the position within it, and the other connecting lines for multiple stations. The user flag indicator will also change accordingly, showing the multiplicity order of the station. Note that this may alter the scope of the **FPATH** function in case you want to repeat the enumeration of a result.

Showing the Results, { FPATH, XTRAIL }

The output of the connecting functions is always the full path, i.e. enumerating all the stations between the from/to points, and not only the transfer points. This can be a lengthy list, so you can repeat the enumeration using **FPATH** as often as required after the solution has been found.

If all you need to know are the transfer points, then you can use **XTRAIL** to enumerate only these key stations. Note that each transfer station will be shown twice – as they really are two different stations, even if they're located at the same topological point. When this output type is used the enumeration also includes the station information for each transfer point - as provided by **STINFO**, minus the user flag selection.

From the above descriptions it follows that the first line acts as the main reference for the station. The station search routines will retrieve the position given by the first encounter within the LINES table.

To make it valid across all cases, the scope of **FPATH** and **XTRAIL** includes a variable number of registers. This is controlled by user flags UF 01 and UF 00 as follows:

UF 01	UF 00	Scope
Clear	Indistinctly	T:, Z:, Y:, and X:
Set	Clear	T:, Z:, Y:, X:, L:, and M:
Set	Set	T:, Z:, Y:, X:, L:, M:, N, and O:

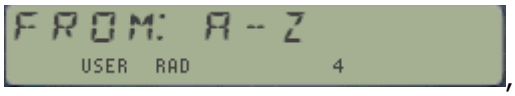
The connecting functions do this automatically, but you should be aware of it in cases you want to use the output functions separately. Also be aware that using non-connected station parameters as From/To data will "throw them for a loop" - so make sure that's never the case - for instance using a LEVEL0 instruction right before. The listing can be halted while any key other than R/S and ON are pressed. Once stopped press [<-] to clear the LCD.

Note: For LEVEL2 and LEVEL 3 cases it's not possible to resume the station enumeration after pressing R/S. Better let it run until the end and repeat the listing using FPATH or XTRAL.

Station parameters and data input/output.{ ^PATH_ }

Each station has a unique parameter, as determined by its position on the LINES table. For the Madrid network the parameters range between 1 and 325. Multi-line stations are repeated in that table, and thus have multiple parameters, one per each line they belong to.

The connecting functions always expect the From: and To: station parameters in the stack Y- and X-registers respectively. You can use the direct parameters if known, but these are automatically retrieved by the **^PATH** function, which will let you choose the From: and the To: stations by their names in a very convenient way.



, then:


Once you choose an initial letter the enumeration will proceed alphabetically from that point on. You can halt it at any moment with R/S, and navigate manually using the SST/BST keys. Pressing XEQ will jump one entire initial section forwards or backwards depending on the SHIFT status. Finally you press ENTER^ to select the station. The back-arrow aborts the function when the enumeration is halted.

If the function doesn't find a connection (because the number of transfers is greater than what it can cope with), the display will show "NO" and the same From:/To: parameters will remain in the stack – so you can try the next level function if so desired.

If the function finds a connection, then the different transfer points are added (inserted really) to the stack and ALPHA registers as needed, according to the following pattern:

- **LEVEL1** adds the two parameters for the transfer station X1, one for each line it belongs to
- **LEVEL2** adds four parameters for the two transfers points, X1, and X2.
- **LEVEL3** adds 6 parameters, two per each of the three transfer points X1, X2, X3.

The table below shows the register layout of the output results for all cases:

Stack Reg.	LEVEL0	LEVEL1	LEVEL2	LEVEL3
T:	**0**	**A2**	**A2**	**A2**
Z:	**A1**	X1(a)	X1(1)	X1(1)
Y:	**A2**	X1(b)	X1(2)	X1(2)
X:	**B2**	**B2**	X2(1)	X2(1)
L:	**B1**	scratch	X2(2)	X2(2)
M:	scratch	scratch	**B2**	X3(1)
N:		scratch	scratch	X3(2)
O:		scratch	scratch	**B2**

During the enumeration of the results the transfer stations will be included twice, which feels like a longer showing time on the LCD during the enumeration.

Note that other registers are also used as scratch, like both registers L(4) and 5(M) are always used as scratch (even for LEVEL1). For the 2- and 3-transfer cases data registers R00 to R03 are also needed to hold the intermediate trials used by the algorithms.

Program listings for the multi-transfer Routines

See below the listings for the FOCAL programs. They're conspicuously short (but hard to figure out!) since they use the dedicated MCODE functions from the module.

Solving a Two X-fers situation. **LEVEL2**program,introducing functions **PREVX** and **NEXTX**.

1	**LBL "LEVEL2"**	
2	CF 06	Not Subroutine mode
3	CF 00	*excludes {N, O} from ROUTE*
4	*"SEARCHING…"*	
5	AVIEW	
6	GTO 06	
7	**LBL "LV2'**	
8	SF 06	Subroutine mode
9	LBL 06	
10	**LEVEL1**	*connected?*
11	GTO 05	*yes!*
12	STO M(5)	linitial position
13	LBL 00	
14	**PREVX**	*previous X-fer*
15	FC? 01	*exists?*
16	GTO 02	*no, try other direction*
17	**LEVEL1**	*connected?*
18	GTO 05	*yes!*
19	GTO 00	*no, keep trying*
20	LBL O2	
21	CLX	
22	RCL M	*restore initial TO:*
23	LBL 01	
24	**NEXTX**	*next X-fer*
25	FC? 01	*exists?*
26	GTO 02	*no, abandon ship*
27	**LEVEL1**	*yes, connected?*
28	GTO 05	*yes!*
29	GTO 01	*no, keep trying*
30	LBL 02	
31	CF 01	signal failure to connect
32	X<> M	*reinstate initial pos*
33	FS? 06	subroutine?
34	RTN	yes, return here
35	*"NO CIGAR"*	no, show error
36	AVIEW	
37	RTN	done.
38	LBL 05	
39	**FROUTE**	show route
40	END	done.

To remark the utilization of the "tentative" targets obtained by the successive movements of the TO: station along the main line it belongs to. This is done by functions **PREVX** and **NEXTX**. This ensures that one transfer point common to the intersecting lines will be reached, and then used to establish the full transfer trail.

Solving a Three X-fers situation: **LEVEL3** program, introducing function **CHANGE**

1	**LBL "LEVEL3"**	
2	*"SEARCHING..."*	
3	AVIEW	
4	STO N(6)	linitial position
5	***0,1***	
6	STO O(7)	
7	RDN	
8	LBL 02	TO: counts as X0
9	STO IND 0(7)	save this marker Xk
10	ISG O(7)	pointer to next REG
11	LBL 00	
12	**PREVX**	*previous X-fer*
13	FC? 01	*exists?*
14	GTO 01	*no, try other direction*
15	STO IND 0(7)	save this marker, X0
16	**XEQ "LV2"**	
17	FS? 01	connected?
18	GTO 05	*yes!*
19	GTO 00	*no, keep trying*
20	LBL 01	
21	**NEXTX**	*next X-fer*
22	FC? 01	*exists?*
23	GTO 02	*no, abandone line*
24	STO IND 0(7)	save this marker, X0
25	**XEQ "LV2"**	
26	FS? 01	connected?
27	GTO 05	*yes!*
28	GTO 01	*no, keep trying*
29	LBL 02	
30	CLX	
31	RCL IND 0(7)	recall last marker
32	ISG O(7)	pointer to next REG
33	**CHANGE**	*change line here*
34	GTO 02	
35	LBL 05	last marker in L, M
36	X<> 01	
37	X<> N(6)	N(6) <> 01
38	X<> 01	
39	SF 00	
40	X<> 00	
41	X<> 0(7)	O(7) <> 00
42	X<> 00	
43	**FROUTE**	
44	END	

Note the parallel with the previous case using "**LVL2**" instead of LEVEL1. Here the notable point is the use of the function **CHANGE** – to change lines at the intersections found by the "tentative" targets used in LEVEL2. This ensures that eventually a common transfer point between two lines will be encountered, and once that's done we'll need to stitch that segment with the previous movements made.

Note that there's no provision for a "not found" contingency – such is the beauty of the Madrid metro, never need to transfer more than three times to reach any point on the network.

I'm Feeling Lucky.{ ANYONE , XPLORE }

The module is equipped with a random number generator based on the Time Module (CX owners rejoice). It was written by JM Baillard, named **RNG** in the FAT.

This functionality is used by **ANYONE** to come up with a valid station parameter, i.e. an integer number between 1 and 325 – which will be placed in X (lifting the stack). The Station name corresponding to this parameter is displayed.

Using **ANYONE** twice instead of the more mindful **^PATH** is a great way to test the different functions in the module: just get two stations at random and see which level is needed to make a connection between them.

Let's see an example of the complete route output using these functions: -

```
ANYONE
        PPE_VERGARA
ANYONE
        UREY_JCARLOS
LEVEL
LEVEL1
        NO
LEVEL
XROM "LEVEL2"
        SEARCHING...
        PPE_VERGARA
        NUNEZ_BALBOA
        AV_AMERICA
        CRUZ_DL_RAYO
        CONCH_ESPINA
        COLOMBIA
        PIO_XII
        DUQ_PASTRANA
        PZA_CASTILLA
        PZA_CASTILLA
        CUZCO
        ST_BERNABEU
        NUEVOS_MINST
        GRIO_MARANON
        ALNS_MARTNEZ
        TRIBUNAL
        PZA_D_ESPANA
        PRINCIPE_PIO
        LAGO
        BATAN
        CASA_D_CAMPO
        COL_JARDIN
        AVIACION_ESP
        CUATRO_VNTOS
        J_VILUMBRLES
        PTA_DEL_SUR
        PTA_DEL_SUR
        PQUE_LISBOA
        ALCORCON_C
        PARQUE_OESTE
        UREY_JCARLOS
        P=28 ST
```

The Whimsical Explorer.

Another connecting function available is the module is **XPLORE** – which uses a rather risky approach to attempt the solution. The method used here is to always change lines in all possible opportunities, i.e. at every transfer station in the lines. This may result in a quick, multi-transfer route but it may also (and likely will) fall into an infinite loop of repetition – so as the name implies it's good to explore if you really like riding the subway.

Like it's often the case in real Metro networks, *this function is currently "under construction"*, so handle it with care.

Modeling the London Tube.

The second map available is for the London Tube network – in its basic configuration, i.e. excluding the Overground, Tram and LDR lines. Even with those restrictions the structure of the underground lines differs from the standard model in two aspects:

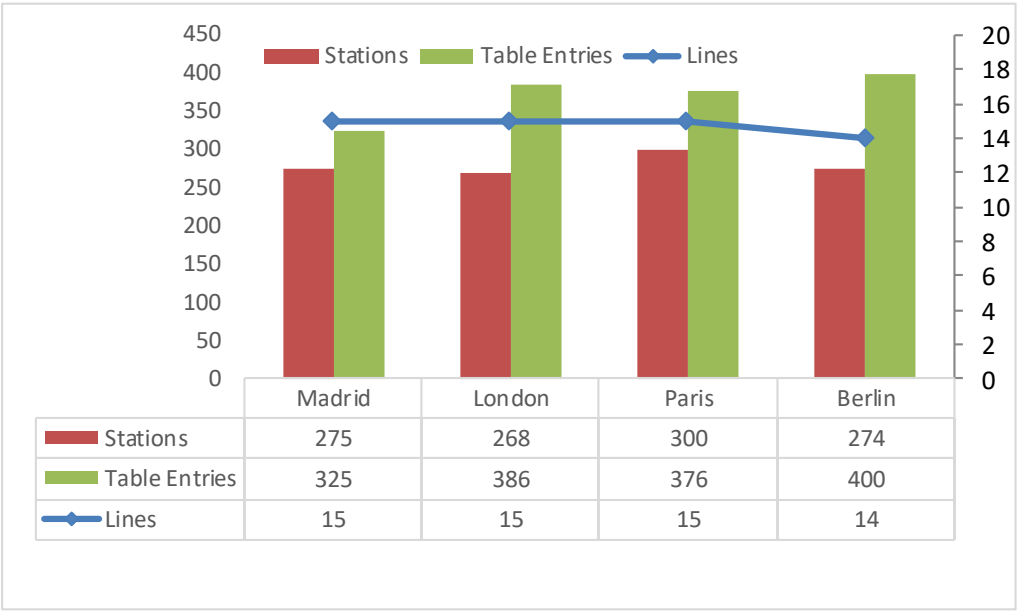
1. A few lines (Central, District, Northern & Metropolitan) have a forked termination, i.e. there's a bifurcation with two different line ends. Some even have that as a mid-line loop, like in the Central line.This contingency has been modeled by treating the forked branches as if they were independent lines, making for a total of 15 lines - the complete list of lines is as follows:

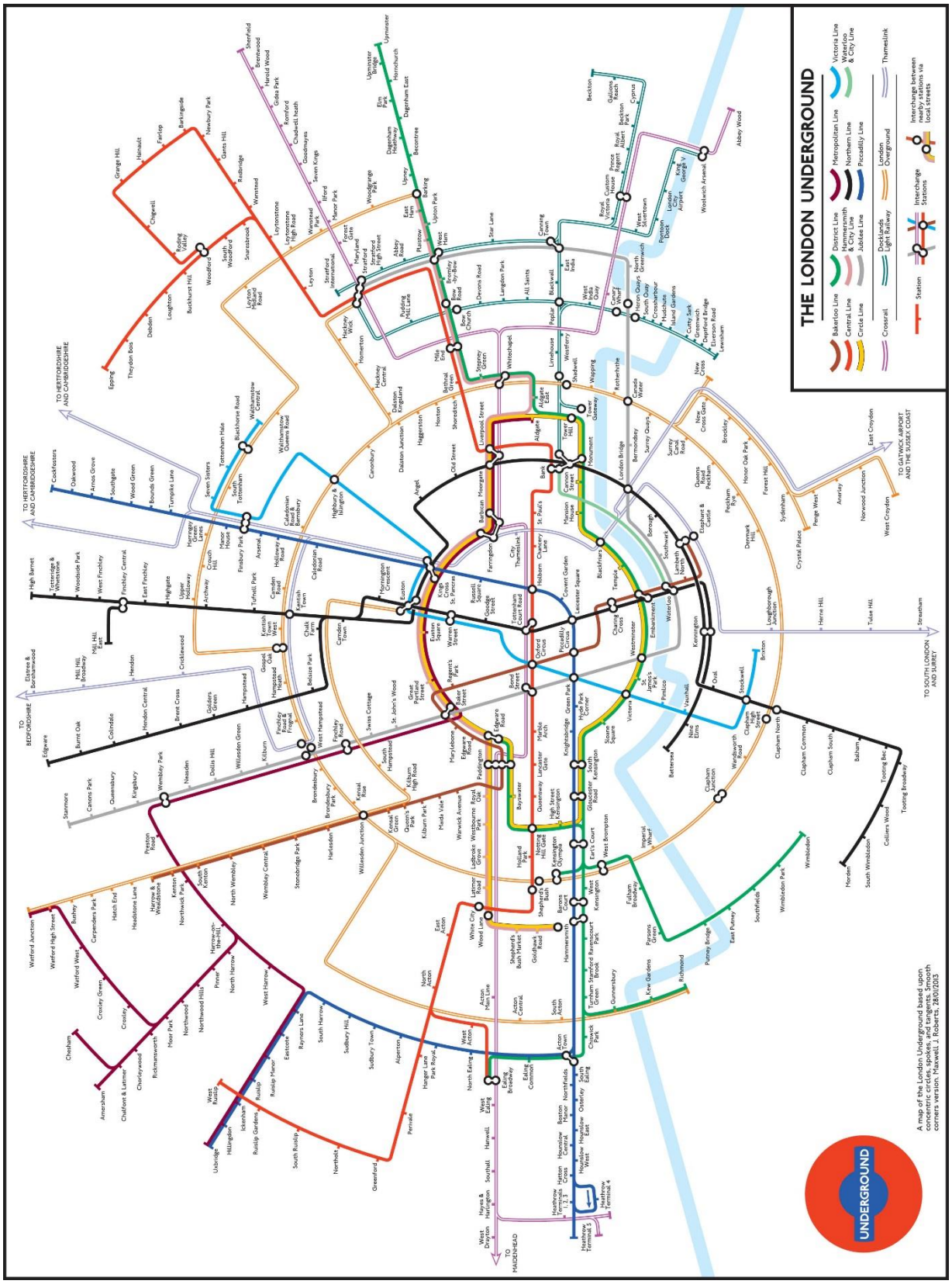
Line #	Name	Line #	Name	Line#	Name
1	Bakerloo	5	District	8	Northern
2	Central	C	District-branch	E	Northrn.-E
B	Central-branch	6	Jubilee	9	Piccadilly
3	Hammersmith	7	Metropolitan	F	P'dilly-branch
4	Circle	D	Metroplt.-Branch	10	Victoria

2. Two stations exceed the maximum multiplicity index supported by the model. This is the case for Baker Street (m=5) and King's Cross (m=6), and has been handled by splitting them in two different stations, each with a multiplicity less than five, and with a common line (acting as the "hinge" for both clones) to allow for the transfer options. With this arrangement only line#7 (Metropolitan) has the double-up stations configured, whereas the others have either one of the splintered versions. This is shown below:

Station	Line#1	Line#2	Line#3	Line#4
Baker St.-1	Bakerloo	Hammersmith	Jubilee	Metropolitan
Baker St.-2	Circle	Metropolitan		
King's Cross-1	Hammersmith	Metropolitan	Northern	Piccadilly
King's Cross-2	Circle	Metropolitan	Victoria	

Other than that, the two network maps share all the same characteristics and support the same functionality. The chart below shows some of their vital constants in comparison:




THE LONDON UNDERGROUND
Bakerloo Line
Central Line
Circle Line
Crossrail
District Line
Hammersmith & City Line
Jubilee Line
Docklands Light Railway
London Overground
Metropolitan Line
Northern Line
Piccadilly Line
Thameslink
Victoria Line
Waterloo & City Line
Station
Interchange Stations
Interchange between nearby stations via local streets
UNDERGROUND
A map of the London Underground based upon concentric circles, spokes, and tangents. Smooth corners version. Maxwell J. Roberts, 28/01/2013

Modeling the Paris Metro network.

The third map available is for the Paris Metro network – in its basic configuration, i.e. excluding the Trams and RER lines. Also very short lines "3bis"and "7bis"are not included. Even with those restrictions the structure of the underground lines differs from the standard model in a few points:

1. Branched lines – as it is the case for line #7 and line #13, and in less degree also line #10. In this implementation, the line#13 branch has been separated as an independent line #15, joining at the "La_Fourche" station. However, the other two branches are "absorbed" into the main line.

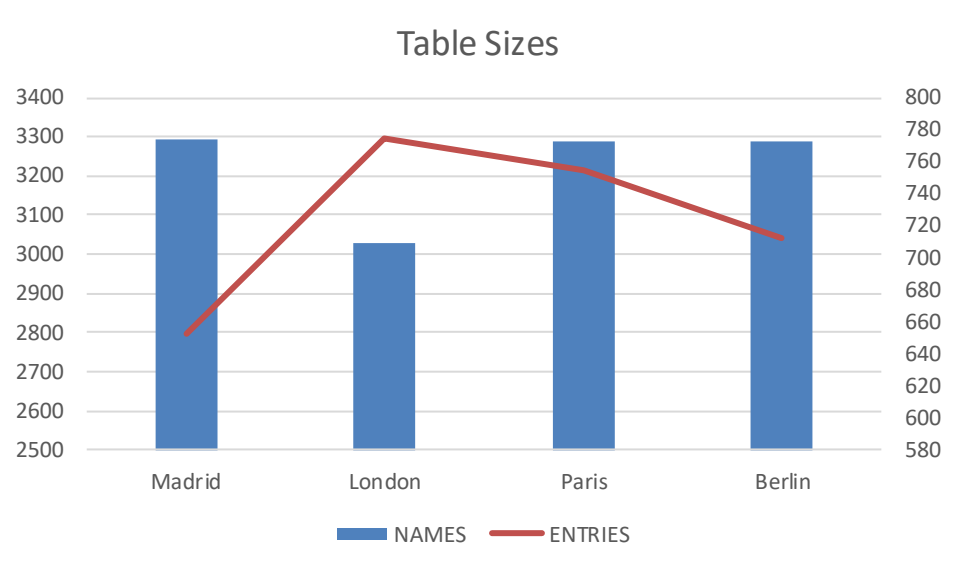
2. Two stations exceed the maximum multiplicity order – "Châtelet" and "République", both with m=5. Like in the London Tube's case, these have been dealt with by adding a second "shadow" station to reduce their multiplicities. The schema uses the following lines for the transfers:

Station	X-fer#1	X-fer#2	X-fer#3	X-fer#4
Châtelet	Line #1	Line #4	Line #7	Line #11
Châtelet-2	Line #11	Line #14		
République	Line #3	Line #5	Line #8	Line #9
République-2	Line #9	Line #11		

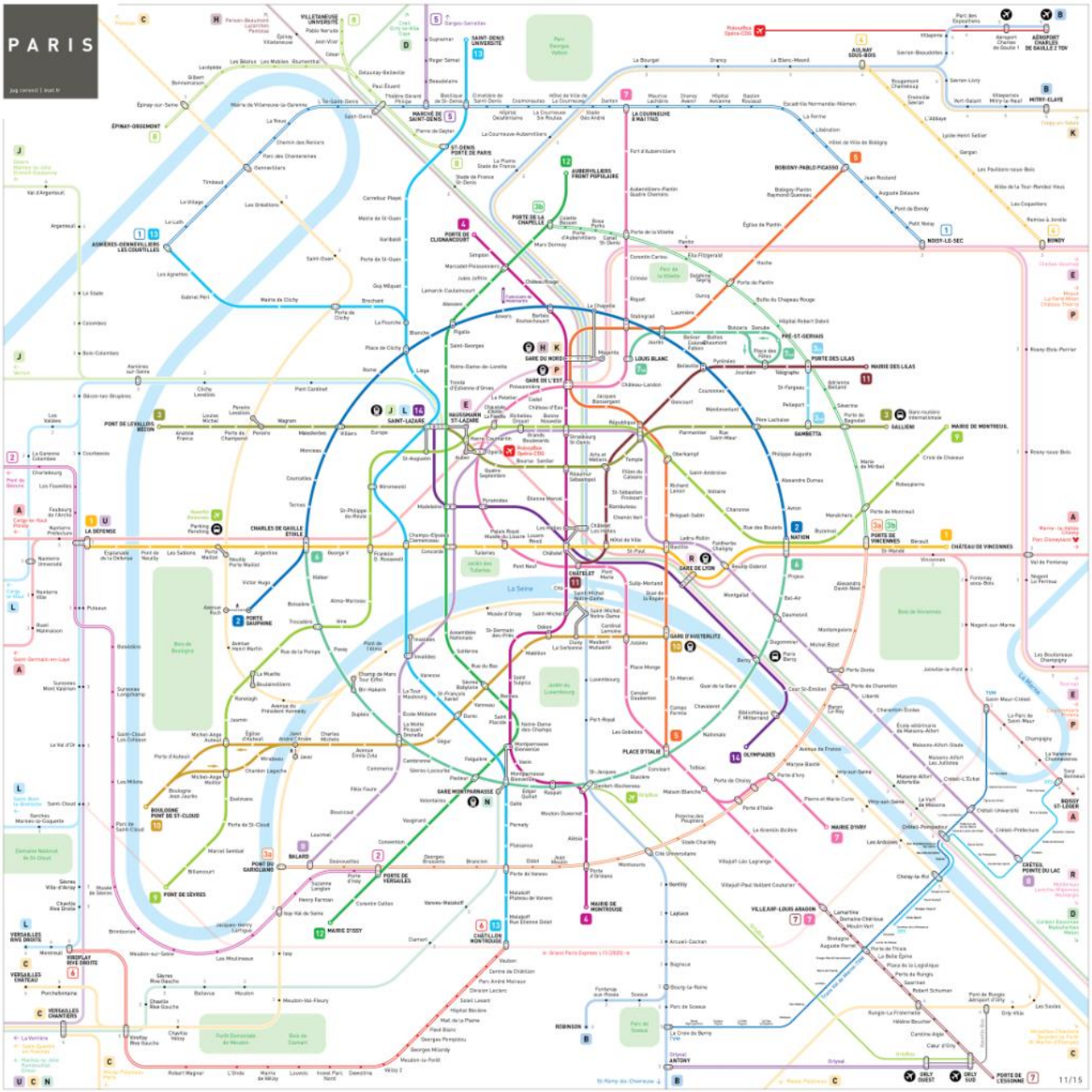
The common ("hinge") lines are line #11 for Châtelet and line #9 for République.

Other than that, the metro stations often used hyphened names with combined descriptions for the place. This makes it troublesome to fit the name in the 12-chars length of the LCD, and also contributes to a larger size for the station NAMES table. I've tried to reach a sensible compromise that works in most of the cases, even if a few instances it's an unusual text. The apostrophes are included but other special French characters (accents, etc.) are ignored.

The chart below shows the table sizes for all maps: the NAMES table in bar format (left axis), and the ENTRIES table in line format (right axis). As usual, English shows to be quite an economical language!



I find the official Paris metro map very hard to read so here's an alternate version that it's easier on the eyes and somehow clearer without the colored background. Note that there's no circular line, but two semi-circles formed by lines #2 and #6 together. Definitely a very dense network!



Modeling the Berlin U-Bahn network.

The fourth map available is for the Berlin U-Bahn, which also includes four of the famous S-Bahn lines so emblematic to that city. The assumptions made are similar to the previous three cases, and can be summarized as follows:

1. Underground lines U1 to U9 are called lines #1 to #9 in the model. They constitute the core section of the model.

2. Circular S-Bahn lines (Ring lines S41 and S42 in the original) have been merged into a common line #10.- The ramification lines S45, S46 are not included in the model.

3. S-Bahn lines S1, S2 (the north-south lines) are called lines #11 and #12 in the model.

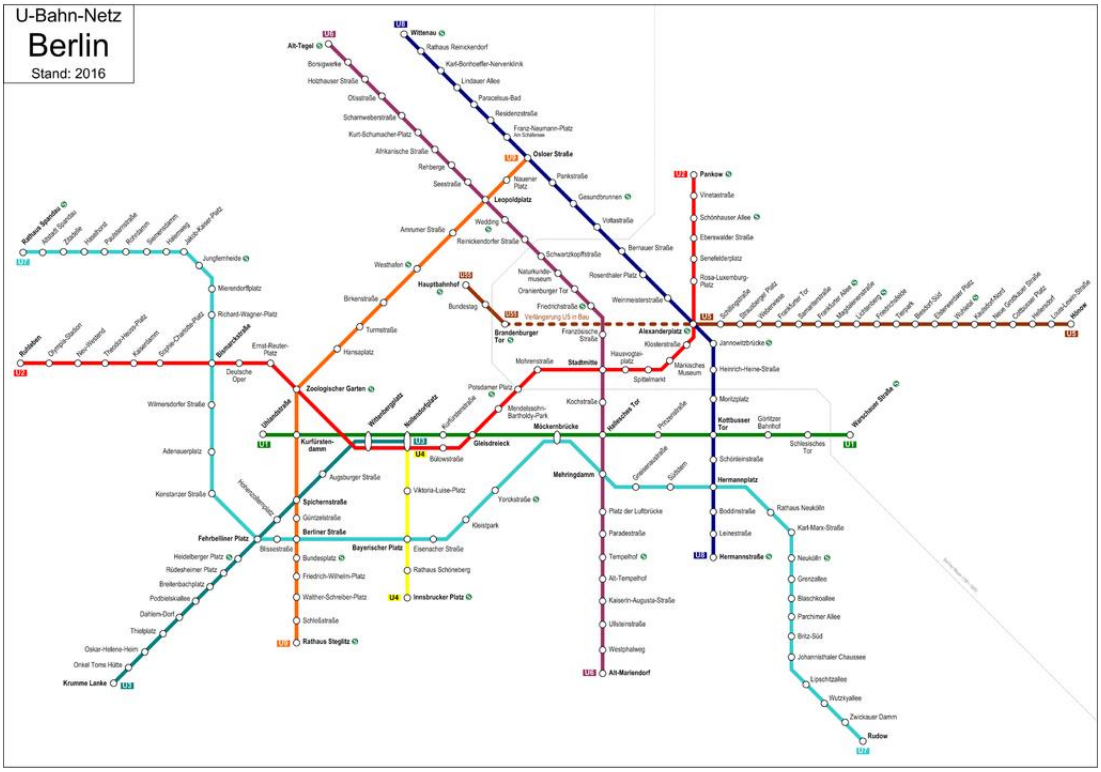
4. S-Bahn lines S3, S5 (the West-East lines) are called lines #13 and #14 in the model.

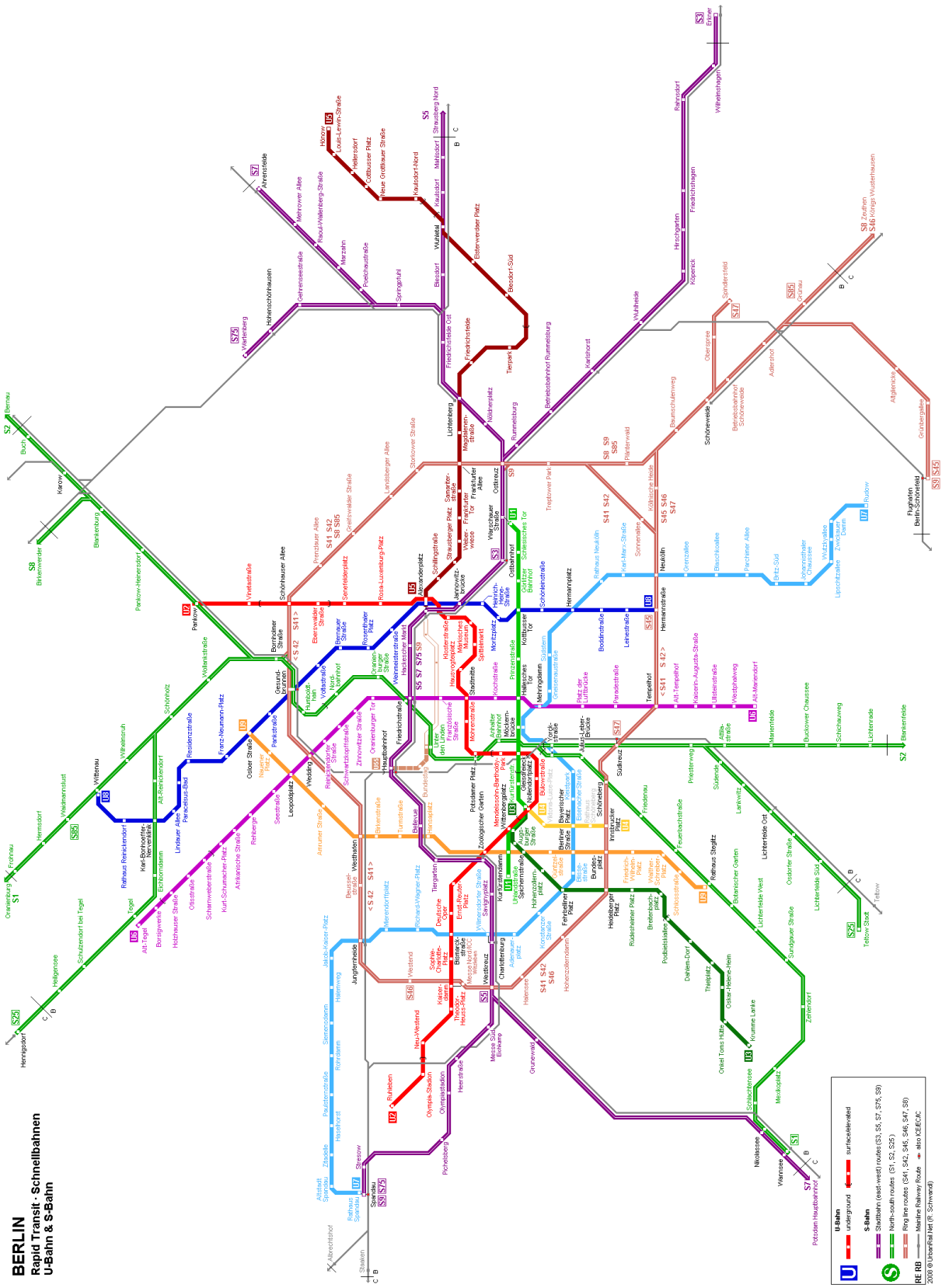
5. No "hinge" or duplicate stations are implemented, even if the Alexander Platz station has five transfers. This is justified by the similar topology of lines S3 and S5.

There is a total of 14 lines with 274 stations, which take up to 400 table entries due to their multiplicity – notably high in the S-Bahn lines.

Some station names are challenging due to the compounded syntax and length, thus frequent compromises have been made. Some common abbreviations include "ST." for Strasse, "PL." for Platz, "BK." for Brücke, and "MK." For Markt. The Umlaut sign is represented by a colon next to the letter in the names. The results should be intelligible even if you're not acquainted with the network – although being familiar with German words would definitely help.

The map below is for the U-Bahn alone, whilst the map in the next page also includes the S-Bahn lines.

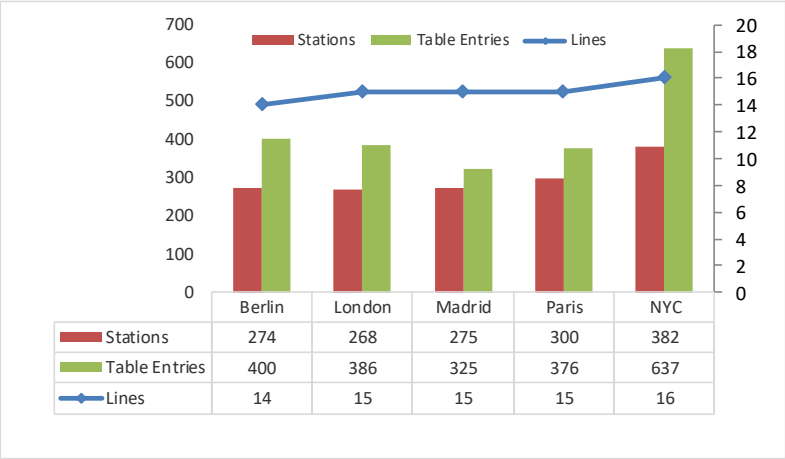



BERLIN
Rapid Transit · Schnellbahnen
U-Bahn & S-Bahn
U-Bahn
underground
surface/elevated
S-Bahn
Stadtbahn (east-west) routes (S3, S5, S7, S75, S9)
North-south routes (S1, S2, S25)
Ring line routes (S41, S42, S45, S46, S47, S8)
Mainline Railway Route
also ICE/EC/IC
RE RB
2008 © UrbanRail.Net (R. Schwandl)

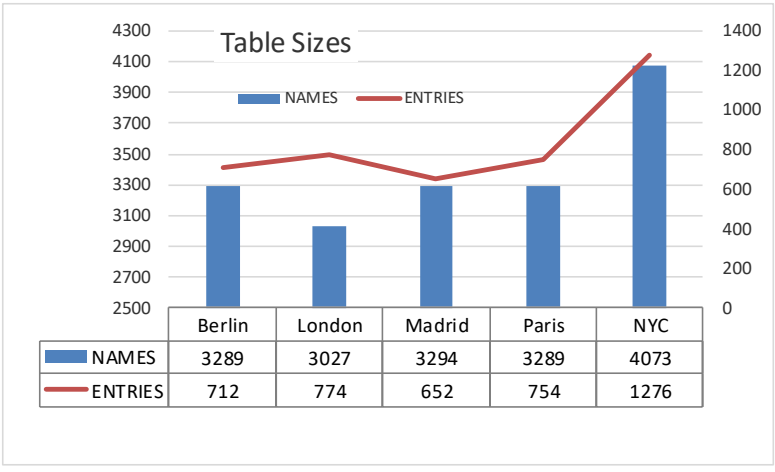
New York, New York….

The fifth and probably last map of this project is for the "great white shark" of underground systems: nothing less than the New York Subway – almost entirely covered, which is not a small feat considering its size and complexity. Yes, this baby is huge, so much so that the station names alone already fill up a complete 4k-page, displacing the lines tables from the standard arrangement. The solution was to move it to the main page, which for the NYC Subway is slightly different from the general-purpose implementation for reasons that will be discussed next. Refer to the table in page# 3 for configuration details.

The charts below show a quick comparison amongst the five networks; behold the NYC Subway case!



	Berlin	London	Madrid	Paris	NYC
Stations	274	268	275	300	382
Table Entries	400	386	325	376	637
Lines	14	15	15	15	16



	Berlin	London	Madrid	Paris	NYC
NAMES	3289	3027	3294	3289	4073
ENTRIES	712	774	652	754	1276

But size alone is not the only problem: with it comes the distinction between Local/Express trains within the same lines (or "tracks"); week-days and week-end schedules, rush-hour skips and late-nigh stops, etc. All that complexity makes it more difficult to plan for in a simple model like this, thus you should be prepared to find some compromises in the choices made, such as:

- The model doesn't distinguish between the day/night or weekend schedules,
- All stations are reported if the train stops there in any of the possible cases.
- Street transfers between stations with different names are implemented as "guest" stations on the associated line
- Stations with line multiplicity m>4 are implemented with the addition of a "shadow" station with the same name. This was also present in the London and Paris networks.

Several additional aspects made modelling the NYC Subway a challenge, not only its vast size.

For starters, there is a combination of numbers and letters for the lines and stations descriptions, and there are plenty of station names that are numeric strings. These two considerations alone required a modification of the station name and line number input routines, to allow for alphanumeric values: using the [SHIFT] key for numbers during the ALPHA mode, as it is standard for the OS.

Note that selecting non-available letters will briefly show the 'NO MATCH" message, and will repeat the same prompt – either for the source or destination depending on the current stage.

The next issue was the number of lines amply exceeding the limit of the model (with 15 max). This was managed by grouping similar lines into a combined one, based on the commonality between them and their shared tracks. If you're at all familiar with the NYC subway you'll no doubt relate to the Local/Express varieties of the same lines, which New Yorkers refer to "trains" (remember the "Take the A-Train" jazz chestnut?). This consolidation lead to the following lines:

- Line 2/3; combining Lines 2 and Line 3 - *minus final section of line #2*
- Line 4/3; combination of lines 4 and 5 - *minus the initial section of line #4*
- Line B/D; combining lines B and D - minus final section of line #B
- Line N/R, combining lines #N and R - *minus final section of line #R*

Even with these consolidations there was no possible to include lines #S and #W, a small omission considering that for the most part they overlap with lines #B, #D, #N, and #R which are included.

For line entries in the prompting functions (such **TLINE**, and **ALINE**) you're expected to *enter the actual line letter or number as in the real network*, i.e. a value included in the table below. For your convenience, the functions won't allow non-available choices. The input will be converted to an internal hex value used by the code as shown below, but you don't need to worry about that unless you want to look under the hood:

NYC Line	Hex token	NYC Line	Hex token	NYC Line	Hex token
1	1	**A**	A	**L**	D
2	2	**C**		**M**	3
3		**B**	B	**N**	5
4	4	**D**		**R**	
5		**E**	E	**Q**	8
6	6	**F**	F	**J**	9
7	7	**G**	C	**Z**	

Note: the output of function **STINFO** *will show the internal hex codes in the transfer field of the LCD.*

The last important difference worth mentioning is that when compared to the other cities there are multiple cases of different stations with the same name. This is likely because of the way the subway was built and the history behind the different companies involved (IND, IRT, and BMT), but it only makes it more confusing to the unaware rider; so novices be aware and ensure you have the right name before you start!

Ready for some examples? Let's ride!

Example 1. Intrepid subway dweller Jeremy Carr needs to find his way between the Green Point Avenue in line G and Central Avenue in line M. Can you help him?

Solution: either using **^PATH** to select by names, or checking the station table directly, the station handles for the begin-end stations are 191 and 298 respectively. Both **LEVEL0** and **LEVEL1** return no connection, so we embark in **LEVEL2** with the following result:

GREENPNT_AV	CLNTN.-WASH.”	**DELANCEY**
NASSAU_AV	FULTON ST.”	DELANCEY
METROP.-LRMR.	HOYT-SCHMR.	MARCY_AV
BDWY.-UNION	**BERGEN/F:**	HEWES_ST
FLUSH.-MARCY	BERGEN/F:	LORIMER_ST
MYRTL-WLGBY.	JAY_ST-METRO	FLUSHING_AV
BEDFD-NOSTR.	YORK_ST	MYRTLE_AV
CLASSON_AV	E._BROADWAY	CENTRAL_AV

The transfers used are BERGEN and DELANCEY, with a total path length of 21 stations.

Not bad, although there’s a much shorter solution (only 12 stations) changing lines at the METROPLT.-LORM., getting on line #L until MYRTL-WYCKF, then changing to line #M for the remaining two stations to the destination. Oh well, at least we got him there even if using the scenic route…

Example 2. Find a path to travel to Coney Island from Far Rockaway (at the end of Line #A). The station handles are: From 58 to 142,

LEVEL1 quickly yields the suitable route with 47 stops, no less – with just a single transfer at “JAY_ST”. station between lines #1 and #F.

Example 3.- Jane Rider needs to go from Clark street to 21 St. – Jackson Ave. What solution does the model find? The station handles are 350 and 190, as selected by **^PATH**. Remember to use the SHIFT key to access the numeric characters in the stations, like in this case for the destination. Also make sure you select “21_ST/JA”, and not “21_ST/QB”.

The path found is 20 stations long, with two transfers at stations “TIMES_SQ” to get on line #7 and “COURT_ST” to end on line #G – and from there backtrack one stop.

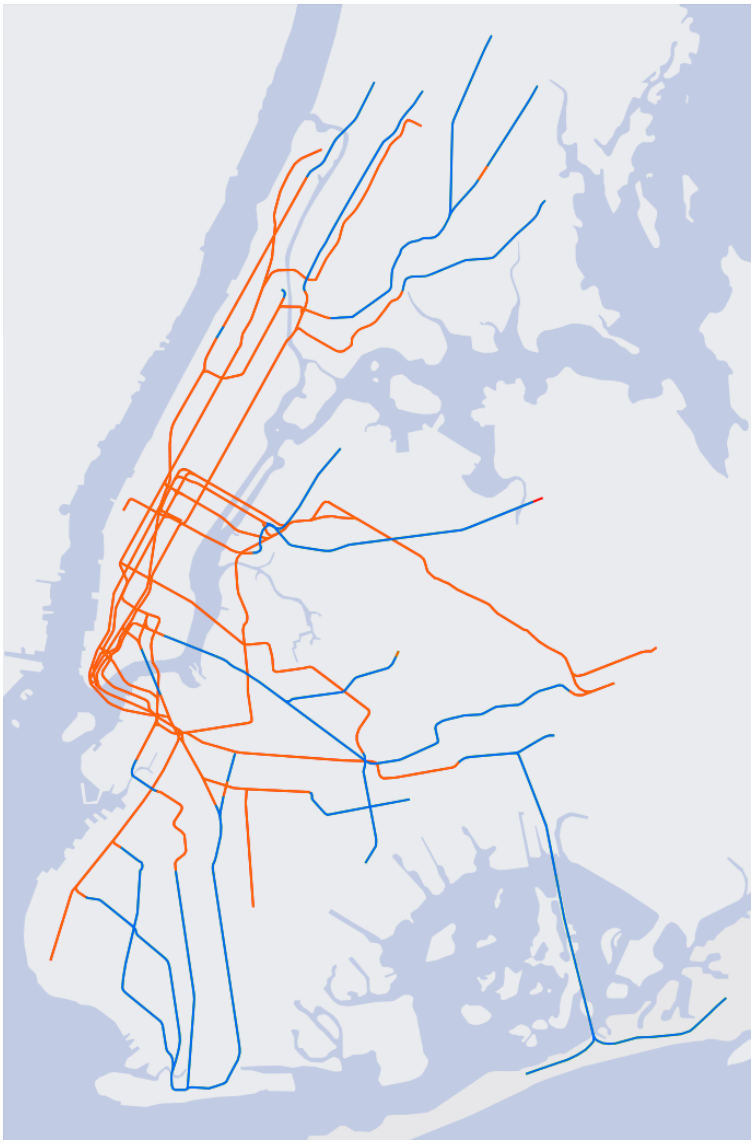

Stopping Patterns
Standard station stops: all trains stop here at all times
Certain trains skip these stops: see below
Stations with special stopping arrangements: see map for details
1 C E F G L M R S Call at all stops at all times, except where shown on the map
2 3 4 A Skip all at all times
5 Manhattan: Skips all Bronx – peak hours, peak direction only: Skips all
6 7 Call at all stops. Peak hours, peak direction only, additional 6 7 trains skip all
B Manhattan: Skips all Bronx/Brooklyn calls at all stops
D Manhattan/Brooklyn: Skips all Bronx – peak hours, peak direction only: Skips all
J Peak hours, peak direction only, skips all
N Manhattan/Queens: Calls at all stops. Brooklyn: skips all
Q Manhattan: Skips all Brooklyn calls at all stops
Z Peak hours, peak direction only, skips all
BRONX
MANHATTAN
QUEENS
BROOKLYN
STATEN ISLAND
New York City Subway Lines
Key
Restricted Services
Most services run seven days a week, daytime and late evenings.
The following services only run at certain times:
Concentric circles and spokes map of the New York Subway
© Maxwell J Roberts, 20/07/2013, www.tubemapcentral.com
Strictly no reproduction without permission

Final Comments.

The NYC Subway has a high level of "centrality", which most certainly will make unnecessary to transfer more than twice to get to any point from any initial station. Sometimes the transfer requires going to the street level, but that seems to be a small price to pay to shorten the route length and ride time if it weren't done.

That's not to say that astute riders won't resort to three-transfer paths for shorter and possibly less time-consuming solutions that those found by this model. This is also the case with the other cities, and is an inherent limitation of the implementation that searches for minimum number of transfers instead of distances or path lengths.

If you're interested in learning more about the NYC Subway there are a plethora of articles and videos online that are excellent references, from Wikipedia to the MTA official side.



Map of line elevation in relation to the ground; underground segments are in orange, and above ground segments are in blue, whether they are elevated, embanked, graded or open cut.

Appendix 1. Structure of the Network Tables.

For each city map, there are two tables that hold the network station information and line topology: the STATIONS table and the LINES table.

The LINES table lists all stations arranged in travelling order, starting with line-1 and ending with line-15. The entries on this table have the addresses in the STATIONS table for the station name and extended info. Each entry consists of two bytes as follows:
a b c - header byte #1
d e f - header byte #2

- **a :** signals the multiplicity of the station: 0 for single, 1 for double, 2 for triple, and 3 for quadruple stations (currently in the Madrid network there is only one station of this kind)
- **b :** signals the line this station belongs to. Note that stations are listed on all lines that include them, so there will be multiple entries on the LINES table under each of the different line sections. All of these entries will have the same address
- {**cef**}: is the hex address in the STATIONS table.
- **d :** denotes the beginning (value =1) or the end (value = 2) of line. This is utilized by the code to determine an "end-of-line" condition.

The STATIONS table consists of an alphabetical list of all stations, where each entry has a variable number of bytes per entry: featuring a two-byte header plus the station name, up to 12 characters max. The header has the following information coded in the first two bytes of the entry:
x y z - header byte #1
m n p - header byte #2

- {**x** } : indicates the multiplicity of the station, together with the "1" digit in { m }.
- {**z y p n**} : are the four lines this station can belong to. This format implies that there cannot be more than 15 lines on the network (from 1 to F) as only one digit is reserved for the line#. It also means that quintuple station(s) will need to be handled as special cases in the code.
- **m** : is always 1 – acts as a delimiting semaphore for the enumeration routines.

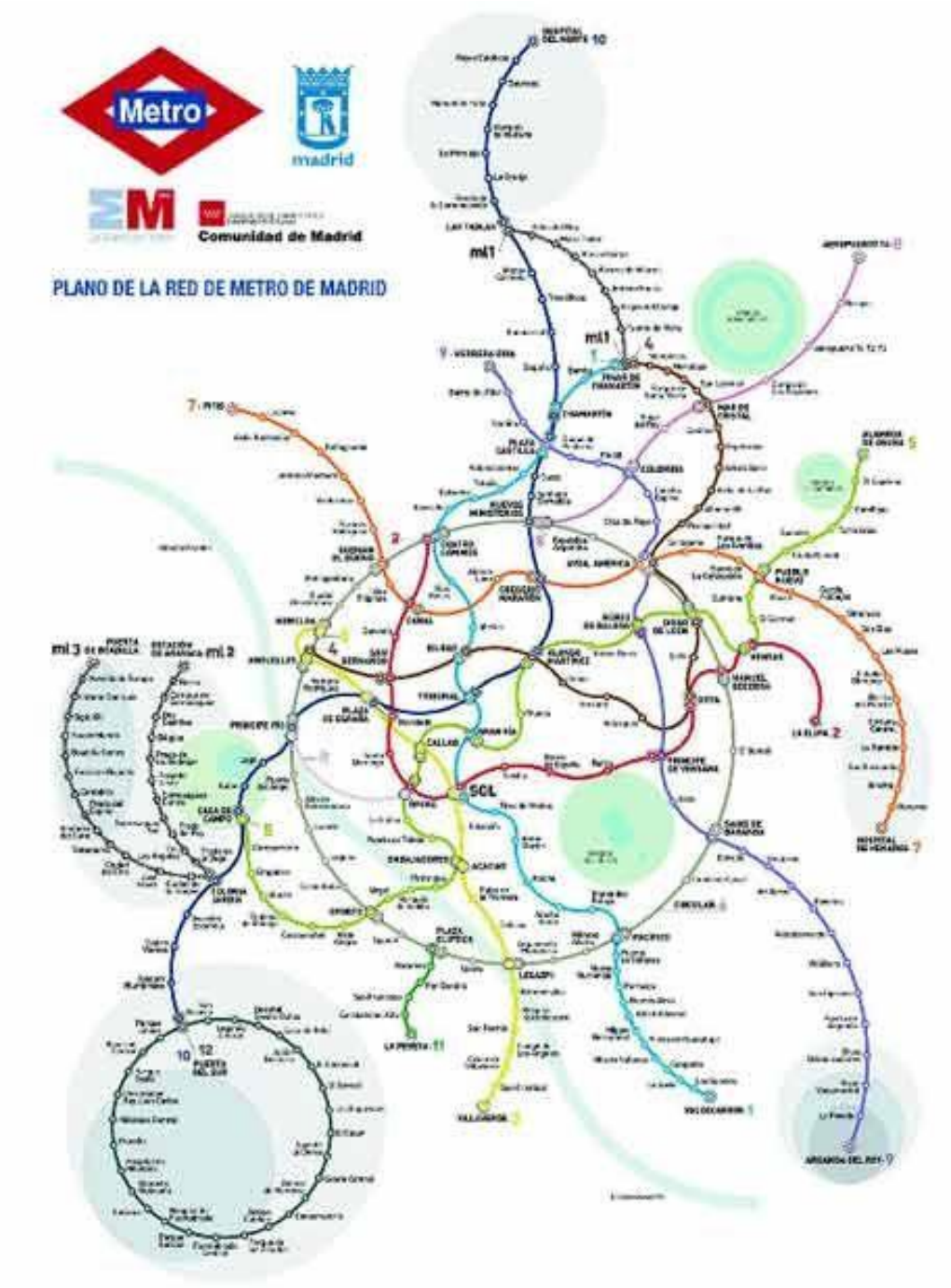
For the Madrid Metro, the STATIONS table is a *huge structure* with 275 entries occupying 3,294 bytes long; whilst the LINES table is 652 bytes long with 325 stations in it. Of these, 275 are unique whilst the remaining 50 are repeated stations as per their multiplicity in the network. These tables occupy the upper page of the module, which can be extended with other maps if you feel like keying in all that data to enter your very own city's underground ;-)

A few interesting tidbits: the London tube was the first metro in the world, and Boston had the first underground in the US - and therefore in the new world. The NYC Subway was thought to be too large so it wouldn't fit within the constraints of the design unless it is size is somehow reduced… to be continued.

Appendix 2. Madrid Metro Stations.

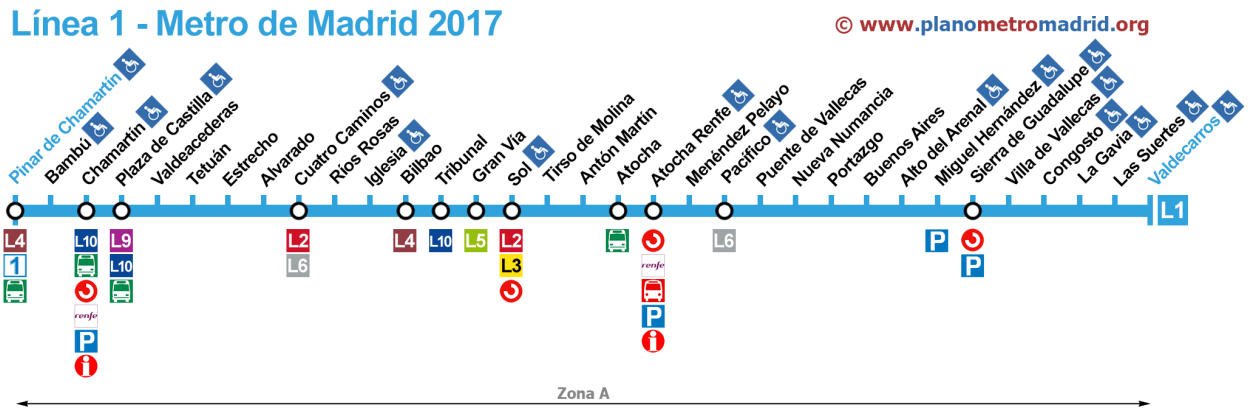
The following tables show all the stations listed in traveling order, with the different transfer points. The information is current as today's date, April 11, 2017. You're encouraged to check the official Metro website for a wealth of information, including a "route planner" that offers several options for optimal path selection at: https://www.metromadrid.es/en/index.html

Note that Line "R" (a single segment linking the Ópera&Ppe_Pío stations) is not included in this implementation. Neither are the "walking passageways" that exist between some stations (like Pza. De España&Noviciado; or Acacias &Embajadores).



Line 1: Pinar de Chamartín - Valdecarros

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
1	1	Pinar de Chamartín	4	L1	0	1
1	2	Bambú	0	0	0	2
1	3	Chamartín	10	0	0	3
1	4	Plaza de Castilla	9	10	0	4
1	5	Valdeacederas	0	0	0	5
1	6	Tetuán	0	0	0	6
1	7	Estrecho	0	0	0	7
1	8	Alvarado	0	0	0	8
1	9	Cuatro Caminos	2	6	0	9
1	10	Ríos Rosas	0	0	0	10
1	11	Iglesia	0	0	0	11
1	12	Bilbao	4	0	0	12
1	13	Tribunal	10	0	0	13
1	14	Gran Vía	5	0	0	14
1	15	Sol	2	3	0	15
1	16	Tirso de Molina	0	0	0	16
1	17	Antón Martín	0	0	0	17
1	18	Atocha	0	0	0	18
1	19	AtochaRenfe	0	0	0	19
1	20	Menéndez Pelayo	0	0	0	20
1	21	Pacífico	6	0	0	21
1	22	Puente de Vallecas	0	0	0	22
1	23	Nueva Numancia	0	0	0	23
1	24	Portazgo	0	0	0	24
1	25	Buenos Aires	0	0	0	25
1	26	Alto del Arenal	0	0	0	26
1	27	Miguel Hernández	0	0	0	27
1	28	Sierra de Guadalupe	0	0	0	28
1	28	Villa de Vallecas	0	0	0	29
1	29	Congosto	0	0	0	30
1	30	La Gavia	0	0	0	31
1	31	Las Suertes	0	0	0	32
1	32	Valdecarros	0	0	0	33



Line 2: Las Rosas – Cuatro Caminos

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
2	1	Las Rosas	0	0	0	34
2	2	Avenida de Guadalajara	0	0	0	35
2	3	Alsacia	0	0	0	36
2	4	La Almudena	0	0	0	37
2	5	La Elipa	0	0	0	38
2	6	**Ventas**	**5**	0	0	39
2	7	**Manuel Becerra**	**6**	0	0	40
2	8	**Goya**	**4**	0	0	41
2	9	**Príncipe de Vergara**	**9**	0	0	42
2	10	Retiro	0	0	0	43
2	11	Banco de España	0	0	0	44
2	12	Sevilla	0	0	0	45
2	13	**Sol**	**1**	**3**	0	46
2	14	**Ópera**	**5**	0	0	47
2	15	Santo Domingo	0	0	0	48
2	16	Noviciado	0	0	0	49
2	17	**San Bernardo**	**4**	0	0	50
2	18	Quevedo	0	0	0	51
2	19	**Canal**	**7**	0	0	52
2	20	**Cuatro Caminos**	**1**	**6**	0	53

Line 3: Villaverde Alto - Moncloa

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
3	1	Villaverde Alto	0	0	0	54
3	2	San Cristóbal	0	0	0	55
3	3	Villaverde Bajo Cruce	0	0	0	56
3	4	Ciudad de losÁngeles	0	0	0	57
3	5	San Fermín - Orcasur	0	0	0	58
3	6	Hospital 12 de Octubre	0	0	0	59
3	7	Almendrales	0	0	0	60
3	8	**Legazpi**	**6**	0	0	61
3	9	Delicias	0	0	0	62
3	10	Palos de la Frontera	0	0	0	63
3	11	Embajadores	0	0	0	64
3	12	Lavapiés	0	0	0	65
3	13	**Sol**	**1**	**2**	0	66
3	14	**Callao**	**5**	0	0	67
3	15	**Plaza de España**	**10**	0	0	68
3	16	Ventura Rodríguez	0	0	0	69
3	17	**Argüelles**	**4**	**6**	0	70
3	18	**Moncloa**	**6**	0	0	71

Line 4: Argüelles – Pinar de Chamartín

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
4	1	**Argüelles**	**6**	**10**	0	72
4	2	**San Bernardo**	**2**	0	0	73
4	3	**Bilbao**	**1**	0	0	74
4	4	**Alonso Martínez**	**5**	**10**	0	75
4	5	Colón	0	0	0	76
4	6	Serrano	0	0	0	77
4	7	Velázquez	0	0	0	78
4	8	**Goya**	**2**	0	0	79
4	9	Lista	0	0	0	80
4	10	**Diego de León**	**5**	**6**	0	81
4	11	**Avenida de América**	**6**	**7**	**9**	82
4	12	Prosperidad	0	0	0	83
4	13	Alfonso XIII	0	0	0	84
4	14	Avenida de la Paz	0	0	0	85
4	15	Arturo Soria	0	0	0	86
4	16	Esperanza	0	0	0	87
4	17	Canillas	0	0	0	88
4	18	**Mar de Cristal**	**8**	0	0	89
4	18	Parque de Santa María	0	0	0	90
4	19	Hortaleza	0	0	0	91
4	19	San Lorenzo	0	0	0	92
4	20	Manoteras	0	0	0	93
4	21	**Pinar de Chamartín**	**1**	**L1**	0	94

Line 5: Alameda de Osuna – Casa de Campo

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
5	1	Alameda de Osuna	0	0	0	95
5	2	El Capricho	0	0	0	96
5	3	Canillejas	0	0	0	97
5	4	Torre Arias	0	0	0	98
5	5	Suanzes	0	0	0	99
5	6	Ciudad Lineal	0	0	0	100
5	7	**Pueblo Nuevo**	**7**	0	0	101
5	8	Quintana	0	0	0	102
5	9	El Carmen	0	0	0	103
5	10	**Ventas**	**2**	0	0	104
5	11	**Diego de León**	**6**	**4**	0	105
5	12	**Núñez de Balboa**	**9**	0	0	106
5	13	Rubén Darío	0	0	0	107
5	14	**Alonso Martínez**	**4**	**10**	0	108
5	15	Chueca	0	0	0	109
5	16	**Gran Vía**	**1**	0	0	110
5	17	**Callao**	**3**	0	0	111
5	18	**Ópera**	**2**	0	0	112
5	19	La Latina	0	0	0	113
5	20	Puerta de Toledo	0	0	0	114
5	21	Acacias	0	0	0	115
5	22	Pirámides	0	0	0	116
5	23	Marqués de Vadillo	0	0	0	117
5	24	Urgel	0	0	0	118
5	25	**Oporto**	**6**	0	0	119
5	26	Vista Alegre	0	0	0	120
5	27	Carabanchel	0	0	0	121
5	28	Eugenia de Montijo	0	0	0	122
5	29	Aluche	0	0	0	123
5	30	Empalme	0	0	0	124
5	31	Campamento	0	0	0	125
5	32	**Casa de Campo**	**10**	0	0	126

Line 6: Circular .

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
6	1	**Principe Pío**	**10**	**(R)**	0	127
6	2	Puerta del Ángel	0	0	0	128
6	3	Alto de Extremadura	0	0	0	129
6	4	Lucero	0	0	0	130
6	5	Laguna	0	0	0	131
6	6	Carpetana	0	0	0	132
6	7	**Oporto**	**3**	0	0	133
6	8	Opañel	0	0	0	134
6	9	**Plaza Elíptica**	**11**	0	0	135
6	10	Usera	0	0	0	136
6	11	**Legazpi**	**3**	0	0	137
6	12	Arganzuela - Planetario	0	0	0	138
6	13	Méndez Álvaro	0	0	0	139
6	14	**Pacífico**	**1**	0	0	140
6	15	Conde de Casal	0	0	0	141
6	16	**Sainz de Baranda**	**9**	0	0	**142**
6	17	O'Donnell	0	0	0	143
6	18	**Manuel Becerra**	**2**	0	0	144
6	19	**Diego de León**	**4**	**5**	0	145
6	20	**Avenida de América**	**4**	**7**	**9**	146
6	21	República Argentina	0	0	0	147
6	22	**NuevosMinisterios**	**8**	**10**	0	148
6	23	**Cuatro Caminos**	**1**	**2**	0	149
6	24	**Guzmán el Bueno**	**7**	0	0	150
6	25	Metropolitano	0	0	0	151
6	26	Ciudad Universitaria	0	0	0	152
6	27	**Moncloa**	**3**	0	0	153
6	28	**Argüelles**	**3**	**4**	0	154

Line 7: H. del Henares –Pitis .

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
7	1	Hospital del Henares	0	0	0	155
7	2	Henares	0	0	0	156
7	3	Jarama	0	0	0	157
7	4	San Fernando	0	0	0	158
7	5	La Rambla	0	0	0	159
L7	6	Coslada Central	0	0	0	160
7	7	Barrio del Puerto	0	0	0	161
7	8	Estadio Olímpico	0	0	0	162
7	9	Las Musas	0	0	0	163
7	10	San Blas	0	0	0	164
7	11	Simancas	0	0	0	165
7	12	García Noblejas	0	0	0	166
7	13	Ascao	0	0	0	167
7	14	**Pueblo Nuevo**	**5**	0	0	168
7	15	Barrio de la Concepción	0	0	0	169
7	16	**Guzmán el Bueno**	**6**	0	0	170
7	16	Parque de las Avenidas	0	0	0	171
7	17	Cartagena	0	0	0	172
7	18	**Avenida de América**	**4**	**6**	**9**	173
7	19	**Gregorio Marañón**	**10**	0	0	174
7	20	Alonso Cano	0	0	0	175
7	21	**Canal**	**2**	0	0	176
7	22	Islas Filipinas	0	0	0	177
7	25	Francos Rodríguez	0	0	0	178
7	25	Valdezarza	0	0	0	179
7	26	Antonio Machado	0	0	0	180
7	27	Peñagrande	0	0	0	181
7	28	Avenida de la Ilustración	0	0	0	182
7	29	Lacoma	0	0	0	183
7	30	Pitis	0	0	0	184

Line 9: Paco de Lucía – Arganda del Rey .

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
9	1	Paco de Lucía	0	0	0	193
9	2	Mirasierra	0	0	0	194
9	3	Herrera Oria	0	0	0	195
9	4	Barrio del Pilar	0	0	0	**196**
9	5	Ventilla	0	0	0	197
9	6	**Plaza de Castilla**	**1**	**10**	0	198
9	7	Duque de Pastrana	0	0	0	199
9	8	Pío XII	0	0	0	200
9	9	**Colombia**	**8**	0	0	201
9	9	Concha Espina	0	0	0	202
9	10	Cruz del Rayo	0	0	0	203
9	11	**Avenida de América**	**4**	**6**	**7**	204
9	12	**Núñez de Balboa**	**5**	0	0	205
9	14	Ibiza	0	0	0	206
9	14	**Príncipe de Vergara**	**2**	0	0	207
9	15	**Sainz de Baranda**	**6**	0	0	208
9	16	Estrella	0	0	0	209
9	17	Vinateros	0	0	0	210
9	18	Artilleros	0	0	0	211
9	19	Pavones	0	0	0	212
9	20	Valdebernardo	0	0	0	213
9	21	Vicálvaro	0	0	0	214
9	22	San Cipriano	0	0	0	215
9	23	Puerta de Arganda	0	0	0	216
9	24	Rivas Urbanizaciones	0	0	0	217
9	25	Rivas Futura	0	0	0	218
9	26	Rivas Vaciamadrid	0	0	0	219
9	27	La Poveda	0	0	0	220
9	28	Arganda del Rey	0	0	0	221

Line 10: H. Infanta Sofía – Puerta del Sur .

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
10	1	Hospital Infanta Sofía	0	0	0	222
10	2	Reyes Católicos	0	0	0	223
10	3	Baunatal	0	0	0	224
10	4	Manuel de Falla	0	0	0	225
10	5	Marqués de la Valdavia	0	0	0	226
10	6	La Moraleja	0	0	0	227
10	7	La Granja	0	0	0	228
10	8	Ronda de la Comunicación	0	0	0	229
10	9	**Las Tablas**	**L1**	0	0	2130
10	10	Montecarmelo	0	0	0	231
10	11	Tres Olivos	0	0	0	232
10	12	Fuencarral	0	0	0	233
10	13	Begoña	0	0	0	234
10	14	**Chamartín**	**1**	0	0	235
10	15	**Plaza de Castilla**	**1**	**9**	0	236
10	16	Cuzco	0	0	0	237
10	17	Santiago Bernabéu	0	0	0	238
10	18	**NuevosMinisterios**	**6**	**8**	0	239
10	19	**Gregorio Marañón**	**7**	0	0	240
10	20	**Alonso Martínez**	**4**	**5**	0	241
10	21	**Tribunal**	**1**	0	0	242
10	22	**Plaza de España**	**3**	0	0	243
10	23	**Principe Pío**	**6**	**(R)**	0	244
10	24	Lago	0	0	0	245
10	25	Batán	0	0	0	246
10	26	**Casa de Campo**	**5**	0	0	247
10	27	**Colonia Jardín**	**L2**	**L3**	0	248
10	28	Aviación Española	0	0	0	249
10	29	Cuatro Vientos	0	0	0	250
10	30	Joaquín Vilumbrales	0	0	0	251
10	31	**Puerta del Sur**	**12**	0	0	252

Line 7. NuevosMinisterios – Aeropuerto .

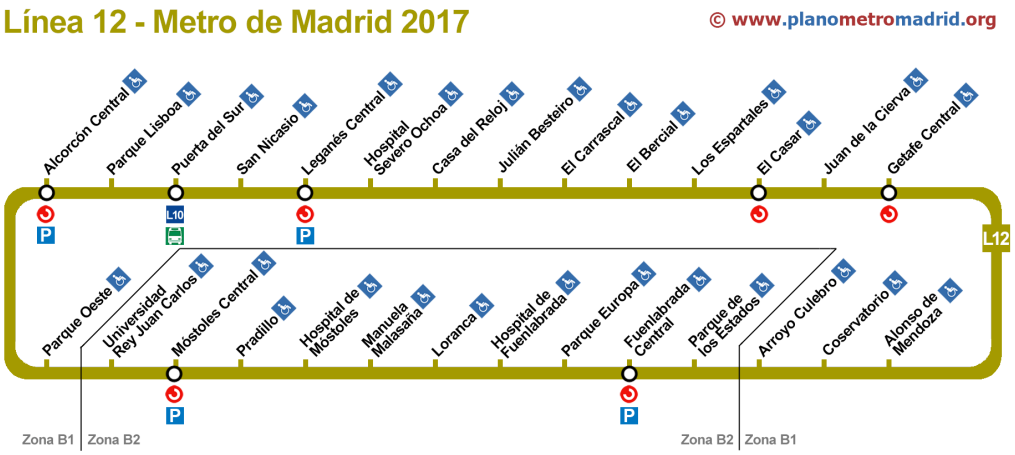
Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
8	1	**NuevosMinisterios**	**6**	**10**	0	185
8	2	**Colombia**	**9**	0	0	186
8	3	Pinar del Rey	0	0	0	187
8	4	**Mar de Cristal**	**4**	0	0	188
8	5	Campo de las Naciones	0	0	0	189
8	6	Aeropuerto T1 T2 T3	0	0	0	190
8	7	Barajas	0	0	0	191
8	8	Aeropuerto T4	0	0	0	192

Line 11. Plaza Elíptica – La Fortuna.

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
11	1	**Plaza Elíptica**	**6**	0	0	253
11	2	Abrantes	0	0	0	254
11	3	Pan Bendito	0	0	0	255
11	4	San Francisco	0	0	0	256
11	5	Carabanchel Alto	0	0	0	257
11	6	La Peseta	0	0	0	258
11	7	La Fortuna	0	0	0	259

Line 12 – Metrosur (Circular)

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
12	1	**Puerta del Sur**	**10**	0	0	260
12	2	Parque Lisboa	0	0	0	261
12	3	Alcorcón Central	0	0	0	262
12	4	Parque Oeste	0	0	0	263
12	5	Universidad Rey Juan Carlos	0	0	0	264
12	6	Móstoles Central	0	0	0	265
12	7	Pradillo	0	0	0	266
12	8	Hospital de Móstoles	0	0	0	267
12	9	Manuela Malasaña	0	0	0	268
12	10	Loranca	0	0	0	269
12	11	Hospital de Fuenlabrada	0	0	0	270
12	13	Parque Europa	0	0	0	271
12	14	Fuenlabrada Central	0	0	0	272
12	15	Parque de losEstados	0	0	0	273
12	16	Arroyo Culebro	0	0	0	274
12	17	Conservatorio	0	0	0	275
12	18	Alonso de Mendoza	0	0	0	276
12	19	Getafe Central	0	0	0	277
12	20	Juan de la Cierva	0	0	0	278
12	21	El Casar	0	0	0	279
12	22	Los Espartales	0	0	0	280
12	23	El Bercial	0	0	0	281
12	24	El Carrascal	0	0	0	282
12	25	Julián Besteiro	0	0	0	283
12	26	Casa del Reloj	0	0	0	284
12	27	Hospital Severo Ochoa	0	0	0	285
12	28	Leganés Central	0	0	0	286
12	29	San Nicasio	0	0	0	287



Lines 13, 14, 15: Metro Ligero L1, L2, L3

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
L1	1	**Pinar de Chamartín**	**1**	**4**	0	288
L1	2	Fuente de la Mora	0	0	0	289
L1	3	Virgen del Cortijo	0	0	0	290
L1	4	Antonio Saura	0	0	0	291
L1	5	Álvarez de Villaamil	0	0	0	292
L1	6	Blasco Ibáñez	0	0	0	293
L1	7	María Tudor	0	0	0	294
L1	8	Palas de Rey	0	0	0	295
L1	9	**Las Tablas**	**10**	0	0	296

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
L2	1	**Colonia Jardín**	**10**	**L3**	0	297
L2	2	Prado de la Vega	**0**	0	0	298
L2	3	Colonia de losÁngeles	**0**	0	0	299
L2	4	Prado del Rey	**0**	0	0	300
L2	5	Somosaguas Sur	**0**	0	0	301
L2	6	Somosaguas Centro	**0**	0	0	302
L2	7	Pozuelo Oeste	**0**	0	0	303
L2	8	Bélgica	**0**	0	0	304
L2	9	Dos Castillas	**0**	0	0	305
L2	10	Campus de Somosaguas	**0**	0	0	306
L2	11	Avenida de Europa	**0**	0	0	307
L2	12	Berna	**0**	0	0	308
L2	13	Estación de Aravaca	**0**	0	0	309

Line#	Stop#	Name	Xf-1	Xf-2	Xf-3	Param#
L3	1	**Colonia Jardín**	**10**	**L2**	0	310
L3	2	Ciudad de la Imagen	**0**	0	0	311
L3	3	José Isbert	**0**	0	0	312
L3	4	Ciudad del Cine	**0**	0	0	313
L3	5	Cocheras	**0**	0	0	314
L3	6	Retamares	**0**	0	0	315
L3	7	Montepríncipe	**0**	0	0	316
L3	8	Ventorro del Cano	**0**	0	0	317
L3	9	Prado del Espino	**0**	0	0	318
L3	10	Cantabria	**0**	0	0	319
L3	11	Ferial de Boadilla	**0**	0	0	320
L3	12	Boadilla Centro	**0**	0	0	321
L3	13	Nuevo Mundo	**0**	0	0	322
L3	14	Siglo XXI	**0**	0	0	323
L3	15	Infante Don Luís	**0**	0	0	324
L3	16	Puerta de Boadilla	**0**	0	0	325

Traveling Salesman Problem for the HP-41

By Jean-Marc Baillard

Overview : Three cases are available depending on the system dimension (2D & 3D), plus an spherical same given its direct applicability to the earth.

These programs employ the "nearest neighbor algorithm". It's likely not the best algorithm to solve this problem, but it may perhaps be useful.

For your convenience, the module includes versions of the programs with included data entry routines. They have a "+" sign in their names, such as "TS2+", "TS3+", and "TSS+"

Two-Dimensional Problem

Data Registers: *Registers R00 thru R2n are to be initialized before executing "TSP"*

- R00 = n = Nb of points
- R01 = x_1 • R03 = x_2 ..... • R2n-1 = x_n
- R02 = y_1 • R04 = y_2 ..... • R2n = y_n

Flags: none; Subroutines: none ; *(161 bytes / SIZE 2n+1)*

```
01 LBL "TSP"    21 X^2          41 CLX          61 RCL IND X
02 RCL 00       22 ISG M        42 SIGN         62 RCL 02
03 2            23 ISG N        43 ST- Z        63 -
04 -            24 RCL IND M    44 -            64 X^2
05 ST+ X        25 RCL IND N    45 RCL IND X    65 DSE N
06  E3          26 -            46 X<> IND Z    66 RCL IND N
07 /            27 X^2          47 STO IND Y    67 RCL 01
08 ISG X        28 +            48 CLX          68 -
09 STO M        29 RCL O        49 LBL 03       69 X^2
10 LBL 01       30 X<=Y?        50 SIGN         70 +
11 RCL M        31 GTO 03       51 ST- M        71 SQRT
12 2.004        32 X<>Y         52 ISG N        72 DSE N
13 +            33 STO O        53 GTO 02       73 LBL 04
14 STO N        34 RCL N        54 ST+ M        74 RCL IND M
15  E99         35 RCL M        55 ISG M        75 RCL IND N
16 STO O        36 2            56 GTO 01       76 -
17 LBL 02       37 +            57 RCL 00       77 X^2
18 RCL IND M    38 RCL IND X    58 ST+ X        78 DSE M
19 RCL IND N    39 X<> IND Z    59 STO M        79 DSE N
20 -            40 STO IND Y    60 STO N        80 RCL IND M
81 RCL IND N
82 -
83 X^2
84 +
85 SQRT
86 +
87 DSE M
88 DSE N
89 GTO 04
90 RCL 00
91 ST+ X
92  E3
93 /
94 ISG X
95 X<>Y
96 CLA
97 END
```

STACK	INPUTS	OUTPUTS
Y	/	1.2n
X	/	Length

Example: The coordinates of 8 points are given as follows:

```
   x |  1  2  6  7  9  8  3  4      Store these 16       R01  R03  R05  R07  R09  R11  R13  R15
   y |  1  7  9  6  4  2  5  8      numbers into:        R02  R04  R06  R08  R10  R12  R14  R16
respectively

   Number of points = 8  STO 00

   XEQ "TSP"   >>>>    L = 26.47818047          ---Execution time = 34s---
        X<>Y   1.016
```

The points have been permuted and we find in R01 thru R24:

```
   x |  1  3  2  4  6  7  9  8
   y |  1  5  7  8  9  6  4  2
```

So, the suggested solution is 1-7-2-8-3-4-5-6

Note: If there are 23 points, the execution time is about 4m00s

Tri-Dimensional Problem

Data Registers: *Registers R00 thru R3n are to be initialized before executing "TSP3"*

- R00 = n = number of points
- R01 = x_1 • R04 = x_2 ..... • R3n-2 = x_n
- R02 = y_1 • R05 = y_2 ..... • R3n-1 = y_n
- R03 = z_1 • R06 = z_2 ..... • R3n = z_n

Flags: none; Subroutines: none. *(202 bytes / SIZE 3n+1)*

```
01 LBL "TSP3"  22 X^2         43 RCL M      64 2          85 +          106 DSE M
02 RCL 00      23 ISG M       44 3          65 ST- M      86 DSE N      107 DSE N
03 2           24 ISG N       45 +          66 ISG N      87 RCL IND N  108 RCL IND M
04 -           25 RCL IND M   46 RCL IND X  67 GTO 02     88 RCL 01     109 RCL IND N
05 3           26 RCL IND N   47 X<> IND Z  68 ST+ M      89 -          110 -
06 *           27 -           48 STO IND Y  69 ISG M      90 X^2        111 X^2
07  E3         28 X^2         49 CLX        70 GTO 01     91 +          112 +
08 /           29 +           50 SIGN       71 RCL 00     92 SQRT       113 SQRT
09 ISG X       30 ISG M       51 ST- Z      72 3          93 DSE N      114 +
10 STO M       31 ISG N       52 -          73 *          94 LBL 04     115 DSE M
11 LBL 01      32 RCL IND M   53 RCL IND X  74 STO M      95 RCL IND M  116 DSE N
12 RCL M       33 RCL IND N   54 X<> IND Z  75 STO N      96 RCL IND N  117 GTO 04
13 3.006       34 -           55 STO IND Y  76 RCL IND X  97 -          118 RCL 00
14 +           35 X^2         56 CLX        77 RCL 03     98 X^2        119 3
15 STO N       36 +           57 SIGN       78 -          99 DSE M      120 *
16  E99        37 RCL O       58 ST- Z      79 X^2        100 DSE N     121  E3
17 STO O       38 X<=Y?       59 -          80 DSE N      101 RCL IND M 122 /
18 LBL 02      39 GTO 03      60 RCL IND X  81 RCL IND N  102 RCL IND N 123 ISG X
19 RCL IND M   40 X<>Y        61 X<> IND Z  82 RCL 02     103 -         124 X<>Y
20 RCL IND N   41 STO O       62 STO IND Y  83 -          104 X^2       125 CLA
21 -           42 RCL N       63 LBL 03     84 X^2        105 +         126 END
```

STACK	INPUTS	OUTPUTS
Y	/	1.3n
X	/	Length

Example: The coordinates of 8 points are as follows:

```
x |  1  2  10  7  9  8  3  4     Store these 24    R01  R04  R07  R10  R13  R16  R19  R22
y |  1  7   1  6  4  2  5  5     numbers into :  R02  R05  R08  R11  R14  R17  R20  R
z |  1  5   4  8  7  1  2  6     respectively      R03  R06  R09  R12  R15  R18  R21  R24
```

Number of points = 8 STO 00

XEQ "TSP3" >>>> L = 33.23751462 ; Execution time = 50s
X<>Y 1.024

The points have been permutated and we find in R01 thru R24: thus the suggested solution is 1-7-2-8-4-5-3-6

```
x |  1  3  2  4  7  9  10  8
y |  1  5  7  5  6  4   1   2
z |  1  2  5  6  8  7   4   1
```

Note: with 17 points the execution time is about 3m25s

Spherical Problem

Data Registers: *Registers R00 thru R2n are to be initialized before executing "TSPS"*

- R00 = n = number of points
- R01 = L_1 • R03 = L_2 ..... • R2n-1 = L_n **L & b expressed in ° ' "**
- R02 = b_1 • R04 = b_2 ..... • R2n = b_n

```
01 LBL "TSPS"
02 DEG
03 RCL 00
04 2
05 -
06 ST+ X
07  E3
08 /
09 ISG X
10 STO M
11 LBL 01
12 RCL M
13 2.004
14 +
15 STO N
16 SIGN
17 STO O
18 LBL 02
19 RCL IND M
20 RCL IND N
21 ISG M
22 ISG N
23 RCL IND M
24 X<>Y
25 RCL IND N
26 XEQ 04
27 RCL O
28 X<=Y?
29 GTO 03
30 X<>Y
31 STO O
32 RCL N
33 RCL M
34 2
35 +
36 RCL IND X
37 X<> IND Z
38 STO IND Y
39 CLX
40 SIGN
41 ST- Z
42 -
43 RCL IND X
44 X<> IND Z
45 STO IND Y
46 CLX
47 LBL 03
48 SIGN
49 ST- M
50 ISG N
51 GTO 02
52 ST+ M
53 ISG M
54 GTO 01
55 GTO 05
56 LBL 04
57 HR
58 X<> T
59 HMS-
60 HR
61 2
62 /
63 SIN
64 X^2
65 RDN
66 HR
67 +
68 2
69 /
70 ST- Y
71 COS
72 X^2
73 ST* T
74 RDN
75 SIN
76 X^2
77 ST* Y
78 -
79 -
80 RTN
81 LBL 05
82 RCL 00
83 ST+ X
84 STO M
85 STO N
86 DSE N
87 RCL IND N
88 RCL IND Y
89 RCL 01
90 RCL 02
91 XEQ 04
92 SQRT
93 ASIN
94 STO O
95 DSE N
96 LBL 06
97 RCL IND M
98 RCL IND N
99 DSE M
100 DSE N
101 RCL IND M
102 RCL IND N
103 X<> T
104 XEQ 04
105 SQRT
106 ASIN
107 ST+ O
108 DSE M
109 DSE N
110 GTO 06
111 RCL 00
112 ST+ X
113  E3
114 /
115 ISG X
116 RCL O
117 ST+ X
118 D-R
119 6371
120 *
121 CLA
122 END
```

Flags: none; Subroutines: none. *(204 bytes / SIZE 2n+1)*

STACK	INPUTS	OUTPUTS
Y	/	1.2n
X	/	Length

Example: The coordinates of 16 "points" are:

City	L	b	City	L	b
Paris	2°20'	48°50'	Ottawa	-75°43'	45°24'
Bombay	72°49'	18°53'	Oslo	10°43'	59°54'
Canberra	149°00	-35°19	Mt. Palomar	-116°52'	33°21'
Glasgow	-4°17'	55°52'	Pulkovo	30°20'	53°46'
Johannesburg	28°01'	-26°11'	Rio	-43°13'	-22°53'
Cape Town	18°29'	-33°56'	Sydney	151°12'	-33°52'
Madrid	-3°41'	40°24'	Washington DC	-77°04'	38°55'
Moscow	37°33'	55°42'	Zikawei	121°11'	31°11'

Store these 32 numbers into R01 to R32 (or use TSS+ directly)

```
16 , STO 00
XEQ "TSPS"  >>>>   L = 71589 km  ;     Execution time = 7m52s
      X<>Y   1.032
```

The coordinates of these 16 points have been modified in R01 thru R32 which suggests the following journey:

Paris-Glasgow-Oslo-Pulkovo-Moskow-Madrid-Ottawa-Washington-MountPalomar-Rio-CapeTown-Johannesburg-Bombay-Zikawei-Sydney-Canberra

Notes:

The length is calculated for the Earth, assuming the mean radius = 6371 km (line 119)
In fact, it's not the shortest path: if you swap the first 2 towns, you get L = 65181 km

So you can try to execute "TSPS" again after changing the order of the points.
Of course V41 and the 41-CL will give you the results much more quickly.

Mazes for the HP-41 , *by Jean-Marc Baillard*

Overview.- The program hereunder generates a pseudo-random rectangular maze of dimensions n x m .You place a random seed in register R00 , n in register Y and m in register X and XEQ "MAZE"

The algorithm uses backtracking (cf reference [1]):

Starting at register R01, the HP41 successively finds non-visited neighbors and deletes the walls between them. When it becomes impossible, it backtracks until it finds an non-visited cell. When that also becomes impossible, the HP41 returns to register R01 and we have our maze.

Registers R01 , R02 , ... , Rmm are the 1st raw.
Rm+1 , Rm+2 , ... , R2m are the 2nd raw and so on...

Program Listing

Data Registers: *(Register R00 is to be initialized before executing "MAZE")*

- R00 = alea R01 to Rmn = the different cells of the maze.

```
01 LBL"MAZE"
02 STO M
03 X<>Y
04 STO N
05 *
06 .12
07 LBL 00
08 STO IND Y
09 DSE Y
10 GTO 00
11 SIGN
12 STO O
13 ST+ 01
14 LBL 10
15 FS? 10
16 VIEW O
17 CF 01
18 CF 02
19 CF 03
20 CF 04
21 CLX
22 STO Q
23 RCL N
24 RCL O
25 RCL M
26 ST* Z
27 +
28 X>Y?
29 GTO 02
30 RCL IND X
31 INT
32 X#0?
33 GTO 02
34 SF 01
35 ISG Q
36 LBL 02
37 RCL O
38 RCL M
39 MOD
40 X=0?
41 GTO 02
42 SIGN
43 RCL O
44 +
45 RCL IND X
46 INT
47 X#0?
48 GTO 02
49 SF 02
50 ISG Q
51 LBL 02
52 RCL O
53 RCL M
54 -
55 X<=0?
56 GTO 02
57 RCL IND X
58 INT
59 X#0?
60 GTO 02
61 SF 03
62 ISG Q
63 LBL 02
64 CLX
65 SIGN
66 RCL O
67 RCL M
68 MOD
69 X=Y?
70 GTO 02
71 RCL O
72 DSE X
73 INT
74 RCL IND X
75 INT
76 X#0?
77 GTO 02
78 SF 04
79 ISG Q
80 LBL 02
81 X<> Q
82 X=0?
83 GTO 07
84 RCL 00
85 R-D
86 FRC
87 STO 00
88 *
89 INT
90 SIGN
91 ST+ L
92 FS? 04
93 4
94 FS? 03
95 3
96 FS? 02
97 2
98 FS? 01
99 1
100 LBL 05
101 RDN
102 DSE L
103 GTO 05
104 GTO IND T
105 LBL 01
106 RCL M
107 RCL O
108 ST+ Y
109 .1
110 GTO 06
111 LBL 02
112 SIGN
113 RCL O
114 ST+ Y
115 .02
116 GTO 06
117 LBL 03
118 RCL O
119 RCL M
120 -
121 STO Y
122 .1
123 GTO 06
124 LBL 04
125 RCL O
126 1
127 -
128 STO Y
129 .02
130 LBL 06
131 ST- IND Y
132 X<> Z
133 X<> O
134 E5
135 /
136 1
137 +
138 ST+ IND O
139 GTO 10
140 LBL 07
141 RCL IND O
142 E2
143 *
144 FRC
145 E3
146 *
147 STO O
148 X#0?
149 GTO 10
150 RCL M
151 RCL N
152 *
153 E3
154 /
155 ISG X
156 CLA
157 END
```

(261 bytes / SIZE n.m+1)

Flags: F01-F02-F03-F04. Set F10 if you want to follow the construction of the maze
Subroutines: none/
Lines 140 & 150 are synthetic three-byte GTO 10
Register Q is used. It can be replaced by another synthetic register like register P

STACK	INPUTS	OUTPUTS
Y	n	/
X	m	1.eee

Where n = number of rows , m = number of columns and 1.eee = cntrol number of the maze with eee = m.n

The driver program **MAZE+** is a more convenient way to use it, as it will prompt for the required data entry, so the user doesn't need to load all values beforehand.

Example: Let's try with n = 7 and m = 10

If we choose r = 1 as the random seed: 1 STO 00

```
 7  ENTER^
10  XEQ "MAZE"  >>>>  1.070                    ---Execution time = 5m11s---
```

Each register now contains a number of the form a.bcdef

a = 0 for an non-visited cell
a = 1 for a visited cell So, at the end, all the cells have been visited and a = 1

The walls are numbered this way:

```
      _3_
 4 |__| 2
      1
```

where walls n°3 and n°4 are in fact the walls 1 and 2 of other contiguous cells, so we only need to deal with walls 1 & 2 as an effective fact.

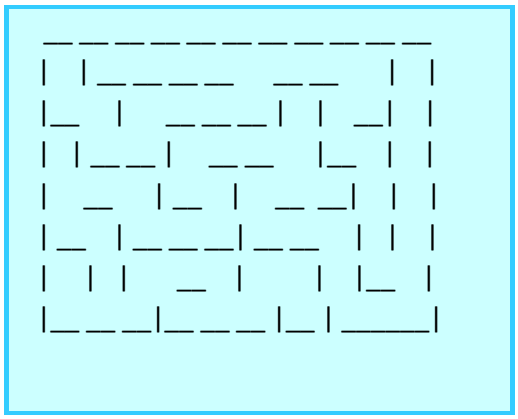
bc = 00 if the walls 1 & 2 are both deleted
bc = 10 if the wall 2 only is deleted
bc = 02 if the wall 1 only is deleted
bc = 12 if the walls 1 & 2 are not deleted

The decimals def indicate the previous visited cell.

So, we only have to take bc into account to draw the maze (We assume that the edges of the rectangle are already drawn)

For R01 bc = 02 , that gives the first cell on the left: |
For R02 bc = 10 , ---------- 2nd --- of the 1st raw: __ ... and so on ...

Then we get a maze that looks (approximately) as shown below:



Finally, choose the entry and the exit at random! - though the HP41 could do that too...

Note: If you have an extended functions module or an HP-41CX, adding the following lines after line 155 may help to visualize the walls more easily:

156 ENTER^	164 FRC	172 95	180 INT	188 LBL 08
157 FIX 0	165 10	173 XTOA	181 X=0?	189 RDN
158 CF 29	166 *	174 RDN	182 GTO 08	190 AVIEW
159 LBL 08	167 INT	175 CF 05	183 FS?C 05	191 ISG X
160 " C"	168 ST- L	176 LBL 08	184 "~ "	192 GTO 08
161 ARCL X	169 SF 05	177 X<> L	185 33	193 X<>Y
162 "~="	170 X=0?	178 10	186 XTOA	194 FIX 4
163 ARCL IND X	171 GTO 08	179 *	187 RDN	195 SF 29

Line 162 is "append = "
Line 184 is "append space"
Set flag F21 and each AVIEW will stop the HP-41, or replace line 190 by PROMPT.

Perhaps will you find better characters to display walls n°1 & n°2 in a cell ?

Reference:

[1] http://en.wikipedia.org/wiki/Maze_generation_algorithm