Artificial Intelligence – Programming Assignment 1

UNIZG FER, academic year 2014/15

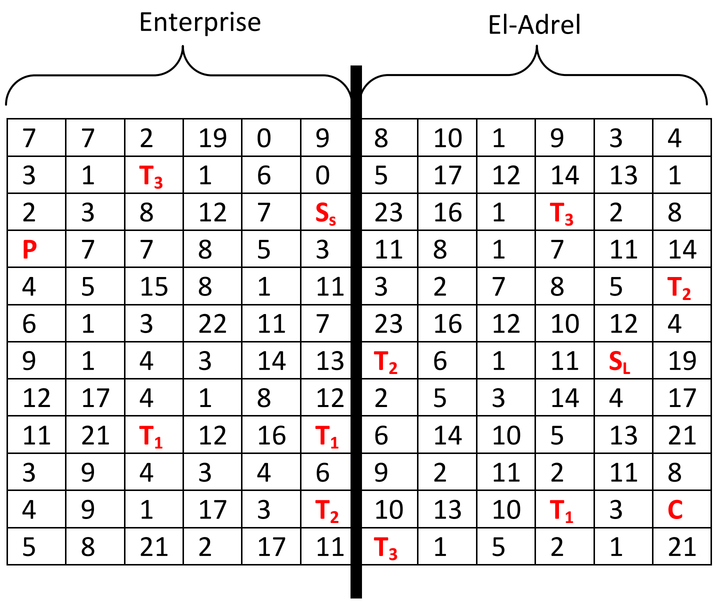
Handed out: 20.3.2015. Due: 5.4.2015. at 23.59.

Star Trek Shortest Path Problem

Star Fleet captain Jean-Luc Picard needs to arrive urgently from his starship Enterprise to the planet El-Adrel in order to prevent the potential conflict with the alien race of Tamarians. Picard first needs to reach one of the transporter on Enterprise to beam himself to the surface of the planet El-Adrel. Then, from the point of arrival on the planet, he needs to come to the meeting place. The magnetic field of the planet is interfering with the ship's transporter systems, so Picard cannot be be sure of the exact position at which he will be arriving on the planet. More precisely, each of the transporters on board of Enterprise will transport Picard to a different location on the planet. Instead of beaming himself on the planet, Picard may choose a slower but safer mean of transportation – a shuttle. If choosing to use the shuttle, Picard first needs to arrive to the shuttle launching bay.

The problem that Picard is facing can be visualized as the 2D grid of size N×N (see the picture on the next page). The starting position (i.e., the position of Enterprise's bridge) is denoted on the map by "P", whereas the meeting place on the planet El-Adrel (i.e., the destination) is denoted by "C". Positions of the transporters are given by marks "T_i", with possibly multiple transporters having the same mark. Using transporter marked "T_i", Picard can beam himself to any other position denoted by the transporter with the same mark "T_i". The shuttle launch bay is denoted by "S_S", whereas the only position on the planet El-Adrel where the shuttle can land is denoted by "S_L". Each position (except for the just mentioned special positions) on the map is marked with a number denoting the height of that position. The time Picard needs to cross from one position to the other equals the absolute difference of the heights of those two positions. Crossing is only allowed between positions on the map that share an edge (i.e., max. four neighbouring positions for each given position). The time of transporting using the transporter equals to the Manhattan distance between the starting and ending position. The time of travelling via shuttle is three times the Manhattan distance between the shuttle launching position ("S_S") and the shuttle landing position "S_L". Walking from Enterprise to El-Adrel **cannot be done** (there is space in between, denoted by the thick black vertical line in the picture), i.e., Picard must arrive to El-Adrel either by transporting himself or by flying the shuttle. All special positions ("P", "C", "T_i", "S_S", "S_L") have the height of 0. It is important to notice that, after arriving to the position marked with a transporter or shuttle launch position, Picard may simply continue walking to a neighbouring, non-special, position.

Being Picard's science officer, your task is to determine the shortest path for Picard to arrive from the Enterprise's bridge to the meeting place on El-Adrel. You need to write



7	7	2	19	0	9	8	10	1	9	3	4
3	1	T_3	1	6	0	5	17	12	14	13	1
2	3	8	12	7	S_S	23	16	1	T_3	2	8
P	7	7	8	5	3	11	8	1	7	11	14
4	5	15	8	1	11	3	2	7	8	5	T_2
6	1	3	22	11	7	23	16	12	10	12	4
9	1	4	3	14	13	T_2	6	1	11	S_L	19
12	17	4	1	8	12	2	5	3	14	4	17
11	21	T_1	12	16	T_1	6	14	10	5	13	21
3	9	4	3	4	6	9	2	11	2	11	8
4	9	1	17	3	T_2	10	13	10	T_1	3	C
5	8	21	2	17	11	T_3	1	5	2	1	21

a program which identifies the shortest path and the optimal solution for the given input map. The optimal solution is the list of position transitions (walking, transportation(s), shuttle flight) that would leads Picard from the position "P" to the position "C" in the shortest time possible.

Each position on the map is uniquely determined by the row and column index (starting from 1, not 0). E.g., on the map from the picture, the position "P" has the coordinates (4,1), position "C" has the coordinates (11,12), and the position "S_L" has the coordinates (7,11). The optimal solution should be output as the list of state transitions, e.g., :

```
(4,1) -> (3,1) -> (3,2) -> (2,2) -> (2,3) -> (3,10) -> (4,10) -> (5,10)
(6,10) -> (7,10) -> (8,10) -> (8,11) -> (9,11) -> (10,11) -> (10,12)
 -> (11,12)
```

The problem needs to be solved using:

(a) Blind search, by implementing uniform cost search algorithm;

(b) Heuristic search, by implementing the A^* algorithm with the following heuristic functions:

(1) The time needed to arrive from the current position to the position "C" is larger than the Manhattan distance between the current position and the position "C";

(2) The time needed to arrive from the current position to the position "C" is larger than the time that would be needed to arrive from the current position to "C" on the map in which: (1) all transporters have the same mark (i.e., each transporter can be used to transport to any other transporter, not only to ones with the same mark), (2) only one transportation is allowed (and it must be from Enterprise to El-Adrel) and (3) heights of the positions are ignored and each walking transition costs 1.

You are encouraged to design your own heuristics that are potentially better than those provided above. In this case, however, you must be certain that your heuristics are optimistic – finding the optimal solution is the main goal of this programming assignment.

Input and Output Format

Your program needs to upload the map configuration from the file on the disk. The map will be given as N×N matrix, where N will always be an even number. The frontier between Enterprise and El-Adrel will always be in the middle of the map, i.e., between columns $N/2$ i $N/2 + 1$. The elements of the matrix given in the file are either numbers (heights for normal positions) or marks of special positions (for which height is 0). For the map given by the picture, the input file has the following format:

```
7 7 2 19 0 9 8 10 1 9 3 4
3 1 T3 1 6 0 5 17 12 14 13 1
2 3 8 12 7 SS 23 16 1 T3 2 8
P 7 7 8 5 3 11 8 1 7 11 14
4 5 15 8 1 11 3 2 7 8 5 T2
6 1 3 22 11 7 23 16 12 10 12 4
9 1 4 3 14 13 T2 6 1 11 SL 19
12 17 4 1 8 12 2 5 3 14 4 17
11 21 T1 12 16 T1 6 14 10 5 13 21
3 9 4 3 4 6 9 2 11 2 11 8
4 9 1 17 3 T2 10 13 10 T1 3 C
5 8 21 2 17 11 T3 1 5 2 1 21
```

The optimal solution (the shortest path) should we written on standard output as follows: (1) the time (i.e., cost) of the shortest path, (2) the total number of opened (i.e. expanded) states, and (3) the state transitions that lead to the goal state (one transition per line). Following are the examples of input maps with the corresponding outputs, which may serve to verify your programs (the given solutions are produced with the A* algorithms using the second heuristic; the number of opened states needs not exactly match the one shown in an example).

Examples

Example 1
Input:

```
7 7 2 19 0 9 8 10 1 9 3 4
3 1 T3 1 6 0 5 17 12 14 13 1
2 3 8 12 7 SS 23 16 1 T3 2 8
P 7 7 8 5 3 11 8 1 7 11 14
4 5 15 8 1 11 3 2 7 8 5 T2
6 1 3 22 11 7 23 16 12 10 12 4
9 1 4 3 14 13 T2 6 1 11 SL 19
12 17 4 1 8 12 2 5 3 14 4 17
11 21 T1 12 16 T1 6 14 10 5 13 21
3 9 4 3 4 6 9 2 11 2 11 8
```

```
4 9 1 17 3 T2 10 13 10 T1 3 C
5 8 21 2 17 11 T3 1 5 2 1 21
```

Output:

```
Minimal cost: 31
Opened nodes: 59
Path:
(4,1) ->
(5,1) ->
(5,2) ->
(6,2) ->
(7,2) ->
(7,3) ->
(8,3) ->
(9,3) ->
(11,10) ->
(11,11) ->
(11,12)
```

Examples 2
Input:

```
8 3 4 T2 9 1 3 4 7 10 1 6 2 4 3 7
P 1 1 7 3 4 6 SS 8 6 2 4 2 7 4 T1
4 T1 8 3 5 T3 8 5 3 8 1 1 2 9 2 3
8 3 T3 8 5 3 7 6 3 T1 8 3 6 9 T3 7
9 1 4 3 7 T1 8 3 6 4 T2 1 9 6 T1 4
8 5 7 4 T4 8 2 9 7 T1 7 2 6 4 9 3
T1 T4 8 2 1 9 8 7 4 8 12 2 SL 9 4 1
8 4 3 7 2 T4 8 9 4 3 2 1 4 5 T1 8
1 2 4 5 9 8 T2 4 5 6 1 2 8 T1 9 4
1 2 8 9 8 4 3 1 1 4 6 4 3 7 5 6
T2 9 8 4 7 T3 7 4 8 2 7 8 9 4 5 5
8 1 3 4 5 8 T4 7 3 8 2 7 4 7 C 5
1 2 9 4 7 8 T3 1 8 T1 8 4 3 5 1 2
9 5 4 8 3 5 2 4 7 5 T1 7 3 9 4 5
3 2 8 9 4 5 3 5 T2 8 4 5 6 8 3 9
1 2 3 4 5 6 7 8 9 T1 7 3 8 2 4 7
```

Output:

```
Minimal cost: 34
Opened nodes: 166
Path:
(2,1) ->
(2,2) ->
(3,2) ->
(9,14) ->
```

```
(10,14) ->
(10,15) ->
(11,15) ->
(12,15)
```

Example 3
Input:

```
7 3 9 8 2 T1 7 4 9 2
P 7 8 4 3 2 6 7 3 9
8 7 3 6 T2 8 6 7 4 8
8 3 6 4 8 2 T1 7 2 8
6 4 7 3 SS 9 7 1 8 2
9 7 T1 2 9 1 4 3 1 3
7 4 9 3 8 2 7 4 SL 9
8 4 3 1 9 3 T2 8 4 3
9 1 2 3 5 3 2 5 7 C
1 8 4 5 7 3 2 8 9 7
```

Output:

```
Minimal cost: 37
Opened nodes: 91
Path:
(2,1) ->
(2,2) ->
(2,3) ->
(2,4) ->
(2,5) ->
(3,5) ->
(8,7) ->
(9,7) ->
(9,8) ->
(9,9) ->
(9,10)
```