Artifficial Intelligence Project

Mitre Flavia Antonia

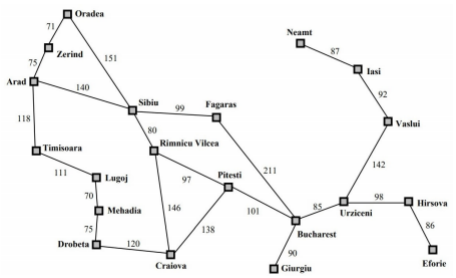
June 8, 2019

Second Year
Group: C.E.N. 2.1
Lecturer: Costin Bădică
Teaching assistant: Ionuţ Murareţu

1 Problem statement

P1 Suppose two friends live in different cities on a map, such as the Romania map shown in Figure 2. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance d(i, j) between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

An example of how the map for this problem should look:



Romania Map

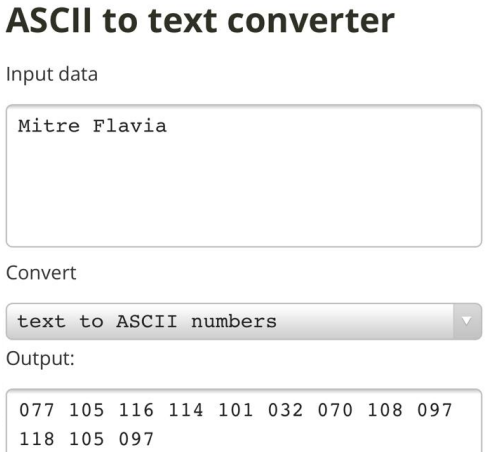
a. Write a detailed formulation for this search problem.
b. Identify a search algorithm for this task and explain your choice.

The input will be given as:

1. Two text files containing the number of cities that can be travelled, and the matrixes of costs each number on a line.

How I computed the hash value to obtain the problem's number:

Made the sum of the ASCII codes of all the characters in my last and 1st first name.



The sum is equal to 1140 (77 + 105 + 116 + 114 + 101 + 32 + 70 + 108 + 97 + 118 + 105 + 97). The final result that gives the number of the problem is: one plus the sum modulo 4 (1 + (1140 modulo 4) = 1 + 0 = 1).

2 Application design and algorithms

- The program is based on using algorithms. The correct approach for this problem is solving using Dynamic Programming.
- The application contains a P1.c file.

The main function is represented by :

- Initializing the maximum cost with 0.
- The call of the function that reads the input from a file, for the first travel: takeInput(0).
- Printing the path for the first travel.
- The call of the function that calculates the minimum cost for the travel: mincost(0,0).
- The minimum cost for the first friend is printed here.
- If the cost for the first friend is bigger than the maximum cost then the maximum cost takes its value. Then the variable "cost" becomes zero.
- The call of the function that reads the input from a file, for the second travel: takeInput(1).
- Printing the path for the second travel.
- The call of the function that calculates the minimum cost for the travel: mincost(0,1).
- The minimum cost for the second friend is printed here.
- If the cost for the second friend is bigger than the maximum cost then the maximum cost takes its value.

```
TAKEINPUT(number)
1. i ← 0
2. nrchr ← 0
3. if number = 0 do
4.    Open file1 and read the data from file (number of cities and cost matrix).
5. else
6.    Open file2 and read the data from file (number of cities and cost matrix).
7. for i ← 0, nrchr − 1, +1 do
8.    aux ← m[i]
9.    far[i] ← aux
10.   far[i] ← far[i] − 48
11. Close the file.
12. n ← far[0]
13. if r = 0 do
14.   Write: "The number of villages is n"
15. k ← 1
16. for i ← 0, n − 1, +1 do
17.   for j ← 0, n − 1, +1 do
18. ary[i][j] ← far[k]
19. k ← k + 1
20. completed[i] ← 0
21. Write "The cost list is:"
22. for i ← 0, n − 1, +1 do
23. Write "enter"
24.   for j ← 0, n − 1, +1 do
25. Write "tab ary[i][j]"
```

Figure 1: Function that reads the input data.

```
LEAST()
1. nc ← 999
2. min ← 999
3. for i ← 0, n − 1, +1 do
4. if ary[c][i]! = 0andcompleted[i] = 0 do
5. if ary[c][i] + ary[i][c] < min do
6. min ← ary[i][0] + ary[c][i]
7. kmin ← ary[c][i]
8. nc ← i
9. if min!=999 do
10. cost ← cost + kmin
11. return nc
```

Figure 2: Returning an int that is equal to the last i.

```
MINCOST()
1. completed[city] ← 1
2. if r = 0 do
3. if city = n − 1 do
4. fr ← 1
5. Write city + 1
6. return
7. else
8. Write city + 1" → "
9. ncity ← least(city)
10. if ncity = 999 do
11. ncity ← 0
12. Writencity + 1
13. if city! = n − 1 do
14. cost ← cost + ary[city][ncity]
15. return
16. else
if r = 0 do
17. if city = n-1 do
18. fr ← 1
19. Write n-city
20. return
21. else
22. Write n − city" → "
23. ncity ← least(city)
24. if ncity = 999 do
25. ncity ← 0
26. Write ncity+1
27. if city !=n-1 do
28. cost ← cost + ary[city][ncity]
29. return
30. mincost(ncity, r)
```

Figure 3: This function computes the minimum cost.

MAIN()
1. $maxcost \leftarrow 0$
2. takeInput(0)
3. Write "The Path from 1 to n is:"
4. mincost(0,0)
5. Write "Minimum cost for the first friend is cost"
6. $maxcost \leftarrow cost$
7. $cost \leftarrow 0$
8. takeInput(1)
9. Write "The Path from n to 1 is:"
10. mincost(0,1)
11. Write:"Minimum cost for the second friend is cost"
12. **if** maxcost ¡ cost **do**
13. $maxcost \leftarrow cost$
14. Write "The minimum cost for the two friends to meet is maxcost"

Figure 4: The main function.

3 Experimental Data

In this section are a few non-trivial input tests and their outputs, and, also the execution time of every one of them.

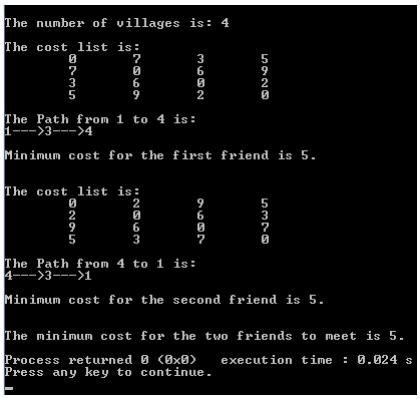


Figure 5: Output for the first set of data.

- In this case the minimum costs for the two friends are the same. The time of execution is 0.024 seconds.

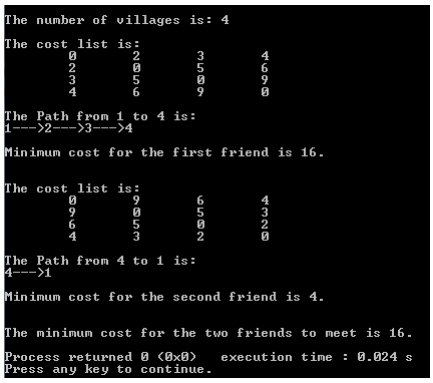


Figure 6: Output for the second set of data.

- In this case the minimum costs for the two friends are different, we observe that the shortest path depends on the starting point as the minimum cost for the first friend is 4 and for the second friend is 16 . The time of execution is 0.024 seconds.

4 Conclusions

This problem is an optimisation for the travelling salesman problem(TSP). The travelling salesman problem asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

The greedy algorithm can be used to give a good but not optimal solution (it is an approximation to the optimal answer) in a reasonably short amount of time. The greedy algorithm heuristic says to pick whatever is currently the best next step regardless of whether that prevents (or even makes impossible) good steps later. It is a heuristic in that practice says it is a good enough solution, theory says there are better solutions (and even can tell how much better in some cases).

References

[1] https://www.programiz.com/c-programming/examples/read-file

[2] https://math.stackexchange.com/questions/2489528/traveling-salesman-problem-with-two-salesmen

[3] https://en.wikipedia.org/wiki/Heuristic_(computer_science)

[4] https://www.overleaf.com/learn/latex/Inserting_Images