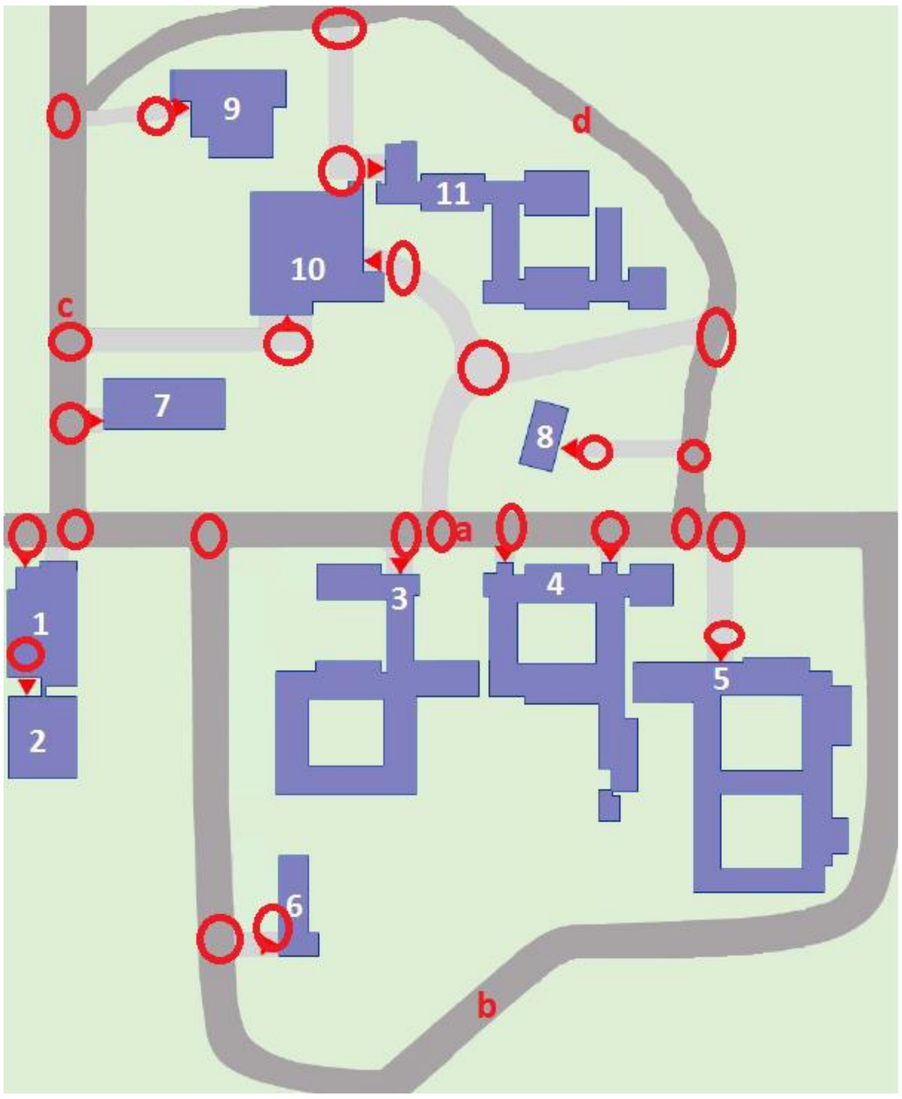
ARTIFICIAL INTELLIGENCE HOMEWORK 01 REPORT

STUDENT NAME	HÜSEYİN ERDOĞAN
STUDENT NO	040100054
CRN	12338

BLG413E HW01 REPORT

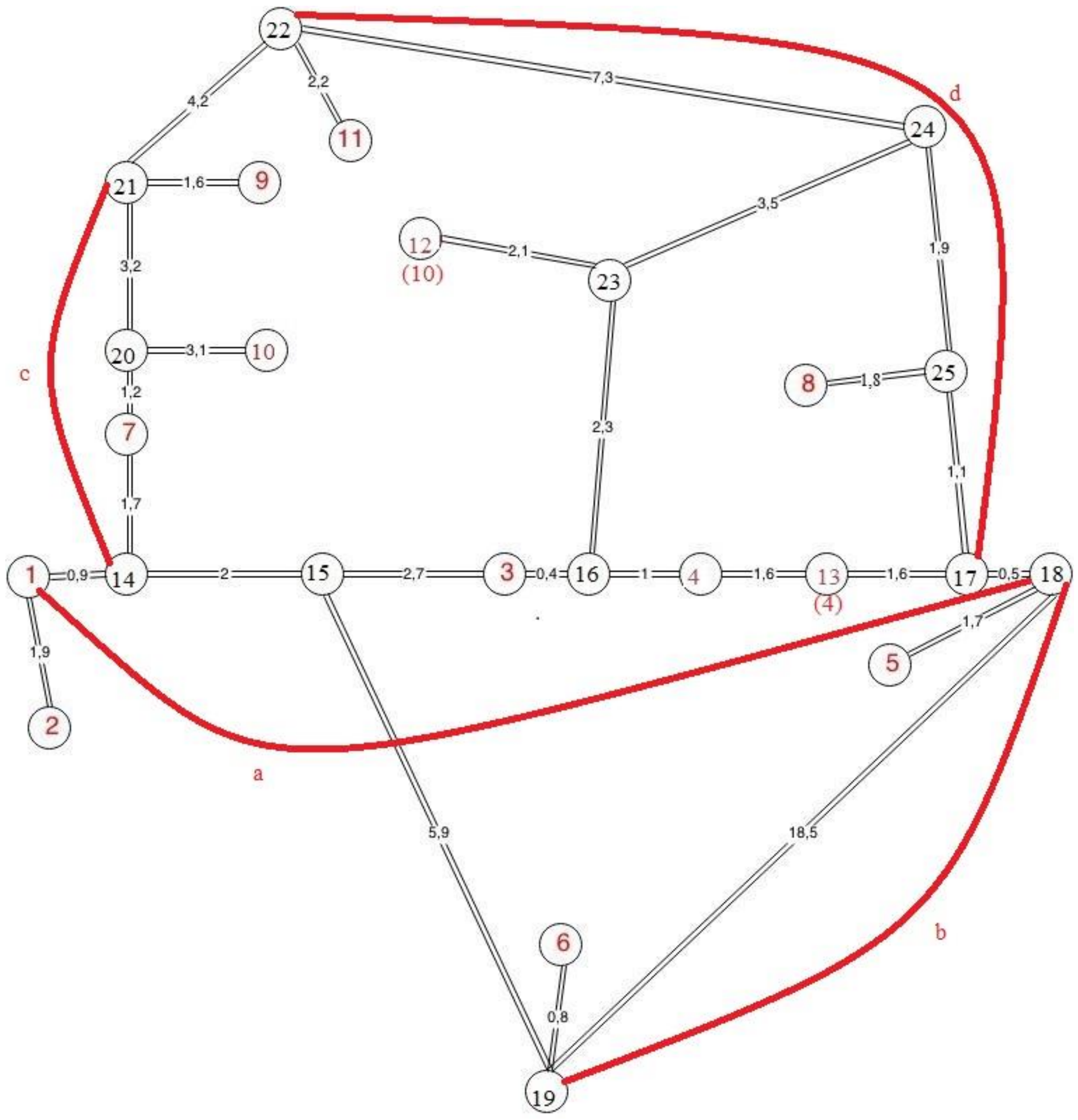
Questions

a) I add extra points for modelling the map. They are showen in below.



After adding extra points modelling was realy easy. I give every node an index value. There is two point named 4 and 10 so I give 13 index value to other point 4 and 12 index value to other point 10. There are total 25 points.

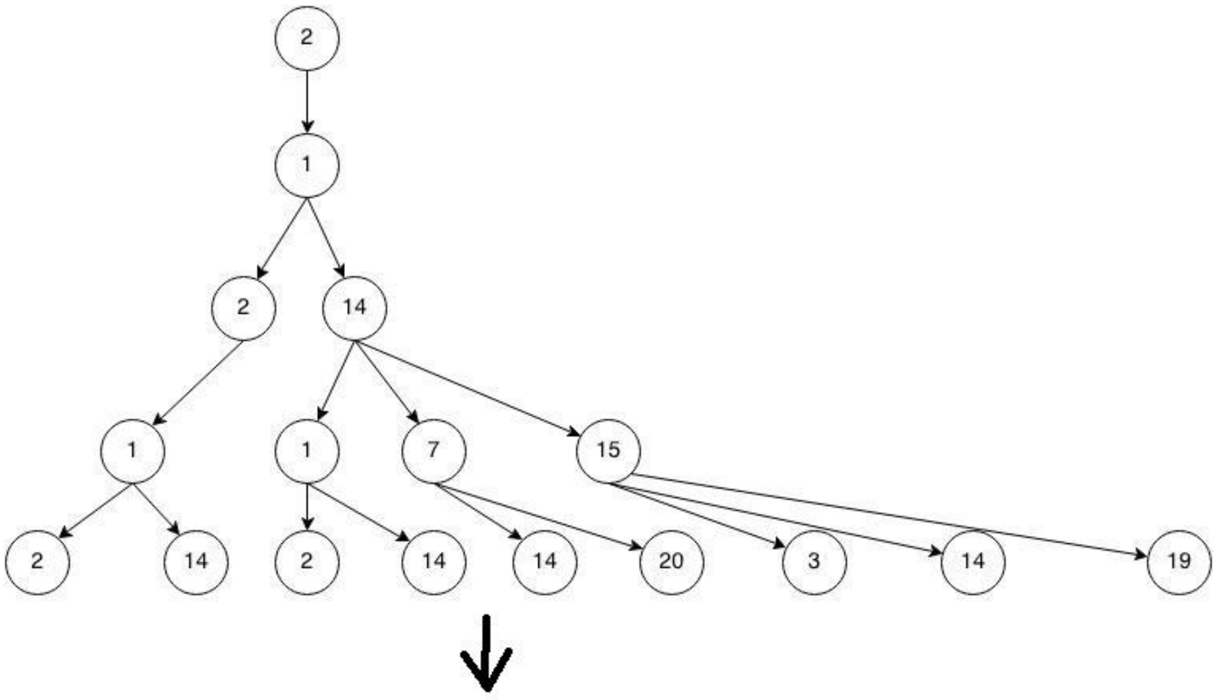
BLG413E HW01 REPORT



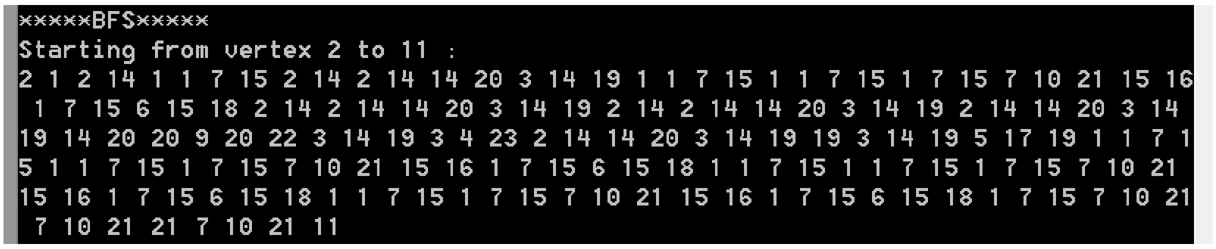
b) Conclusion for DFS and BFS

- In tree search version of BFS, algorithm finds point 11 in 7 layer. In this graph there may not be any big problems but if number of points in graph is increased, finding destination will be last very long. Also many nodes are repeated because of tree version of BFS. Tree version of graph is below.

BLG413E HW01 REPORT

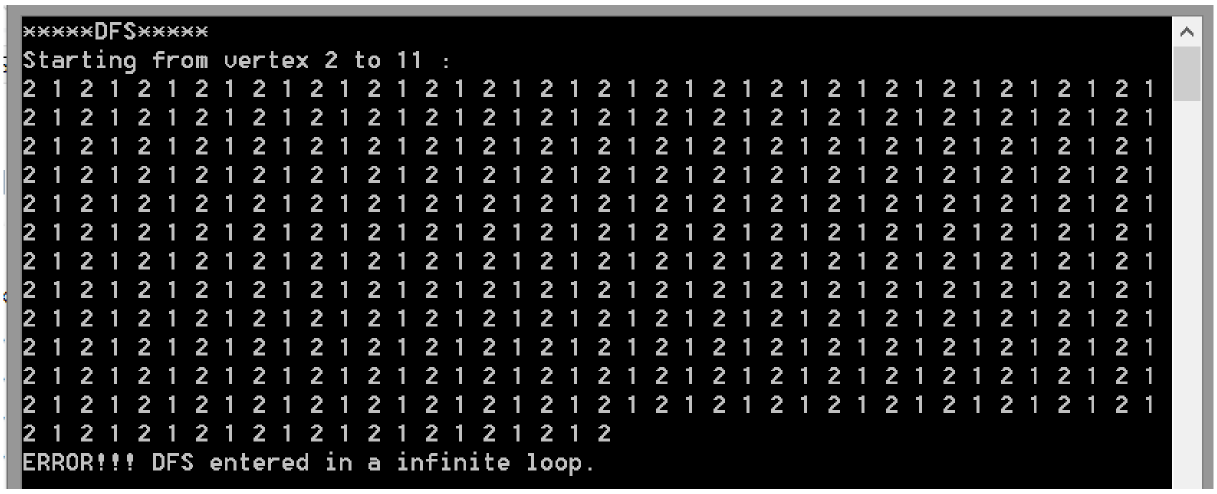


And it continuous to 7th layer. Screenshot of program is below.



```
*****BFS*****
Starting from vertex 2 to 11 :
2 1 2 14 1 1 7 15 2 14 2 14 14 20 3 14 19 1 1 7 15 1 1 7 15 1 7 15 7 10 21 15 16
 1 7 15 6 15 18 2 14 2 14 14 20 3 14 19 2 14 2 14 14 20 3 14 19 2 14 14 20 3 14
19 14 20 20 9 20 22 3 14 19 3 4 23 2 14 14 20 3 14 19 19 3 14 19 5 17 19 1 1 7 1
5 1 1 7 15 1 7 15 7 10 21 15 16 1 7 15 6 15 18 1 1 7 15 1 1 7 15 1 7 15 7 10 21
15 16 1 7 15 6 15 18 1 1 7 15 1 7 15 7 10 21 15 16 1 7 15 6 15 18 1 7 15 7 10 21
 7 10 21 21 7 10 21 11
```

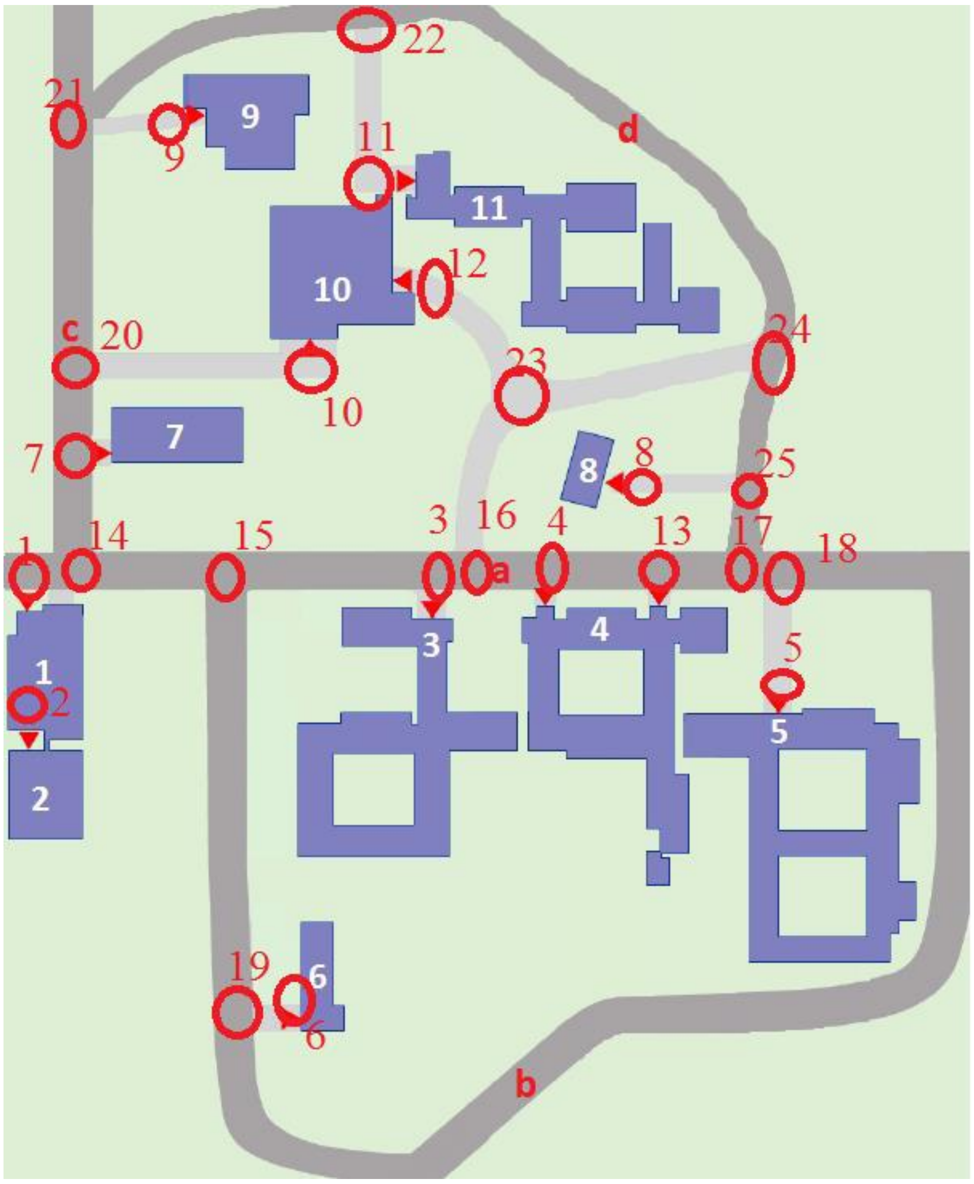
- In tree search version of DFS, algorithm enters an infinite loop and commuting beetween point 2 and point 4. Screenshot of program is below.



```
*****DFS*****
Starting from vertex 2 to 11 :
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
ERROR!!! DFS entered in a infinite loop.
```

BLG413E HW01 REPORT

c) School map with grid for using in heuristic functions is below.



- First heuristic function is Manhattan distance.

The standard heuristic for a square grid is the Manhattan distance.

```
function heuristic(node) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return D * (dx + dy)
```

- Second heuristic function is Diagonal distance.

If your map allows diagonal movement you need a different heuristic.This function can handle diagonals.

```
function heuristic(node) =
```

BLG413E HW01 REPORT

```
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return D * max(dx, dy)
```

Heuristic values for function 1 and function 2. D is 1/60 for both function.

Point	Distance x	Distance y	Max	f1 (D=1/60)	f2 (D=1/60)
1	213,00	325,00	325,00	8,966666667	5,416666667
2	214,00	254,00	254,00	7,8	4,233333333
3	44,00	244,00	244,00	4,8	4,066666667
4	113,00	243,00	243,00	5,933333333	4,05
5	260,00	319,00	319,00	9,65	5,316666667
6	49,00	515,00	515,00	9,4	8,583333333
7	186,00	170,00	186,00	5,933333333	3,1
8	172,00	190,00	190,00	6,033333333	3,166666667
9	130,00	34,00	130,00	2,733333333	2,166666667
10	42,00	118,00	118,00	2,666666667	1,966666667
11	0,00	0,00	0,00	0	0
12	44,00	66,00	66,00	1,833333333	1,1
13	185,00	246,00	246,00	7,183333333	4,1
14	180,00	250,00	250,00	7,166666667	4,166666667
15	90,00	250,00	250,00	5,666666667	4,166666667
16	70,00	250,00	250,00	5,333333333	4,166666667
17	230,00	240,00	240,00	7,833333333	4
18	260,00	250,00	260,00	8,5	4,333333333
19	86,00	520,00	520,00	10,1	8,666666667
20	180,00	110,00	180,00	4,833333333	3
21	190,00	40,00	190,00	3,833333333	3,166666667
22	10,00	100,00	100,00	1,833333333	1,666666667
23	100,00	130,00	130,00	3,833333333	2,166666667
24	250,00	110,00	250,00	6	4,166666667
25	240,00	190,00	240,00	7,166666667	4

- Admissible

There is straigt line beetween point 22 and point 11 and its actual value is 2,2. In first heuristic function, distance beetween these points is 1,83. In second heuristic function, distance beetween these points is 1,66. Both values ,that are found by heuristic functions, are smaller than actual value.

- Consistent

I use this formula for proving functions’s consistence.

BLG413E HW01 REPORT

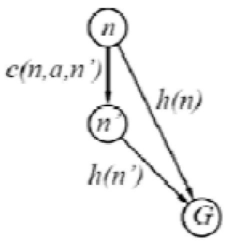
A heuristic is consistent if

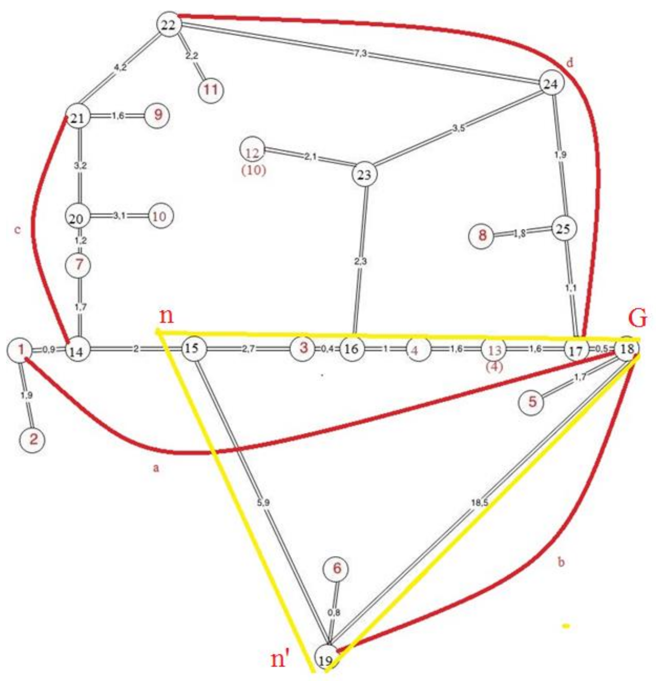
$$h(n) \leq c(n,a,n') + h(n')$$

If h is consistent, we have

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n,a,n') + h(n') \\ &\geq g(n) + h(n) \\ &= f(n) \end{aligned}$$

I.e., $f(n)$ is nondecreasing along any path.





For showing both heuristic functions are consistent, I use triangle with yellow lines.

Actual distance beetween point n and n' is 5,9

For function 1 heuristic value beetween n and G is : (1/60)*(354+2) = 5,93
For function 1 heuristic value beetween n' and G is : (1/60)*(345+274) = 10,4

c(n,a,n') =5,9 , h(n) =5,93 , h(n') = 10,4 so h(n)<h(n')+c(n,a,n')

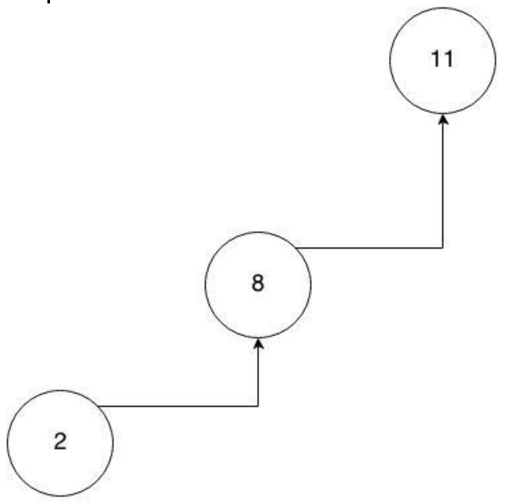
For function 2 heuristic value beetween n and G is : (1/60)*maximum(354,2) = 5,9
For function 2 heuristic value beetween n' and G is : (1/60)* maximum (345,274) = 5,75

c(n,a,n') =5,9 , h(n) =5,9 , h(n') = 10,4 so h(n)<h(n')+c(n,a,n')

d) A* Algorithm

e) Find a route from **2** to **11** by visiting **8.**

First we find a route beetween 2 and 8. Then we find a route beetween 8 and 11. In final step we combine two route.



f) Finding a route from **2** to **11** by passing from the road **b**.

We determine nodes for every road to check passing from that road.
For road **a,** I determine points 15,4.
For road **b,** I determine points 18,19.
For road **c,** I determine points 7, 20.
For road **d,** I determine points 24, 25

We can use this method for finding a route from **2** to **11** by passing from the road **b** like this:

First we find a route beetween 2 and 18. Then we find a route beetween 18 and 19. Then we find a route beetween 19 and 11. In final step we combine these tree routes.