ECS759P Artificial Intelligence: Coursework 1

Guanzhen Wu
161189284

1. Agenda-based Search

   a) Implement DFS, BFS, and UCS

   First, read the tube.csv file and regularize the data which include punctuation and extra spaces generated in the process of reading files. Maintain several lists to store "Starting Station", "Ending Station", "Tube Line", "Average Time Taken", "Main Zone" and "Secondary Zone". Use the add_nodes_from() function of networkx module to add the stations that appeared in the starting station and ending station as nodes to the graph. Add "Tube Line", "Main Zone" and "Secondary Zone" attributes for each node. Add an edge to every two points according to the tube file, and use the average token time between the two stations as the weight. So far, the graph data structure is established. For each of the search algorithm, it will use a visit list to save the visited node in order not to visit it again.

   i. DFS

   When a starting node is passed in, DFS will keep going along one of the edges of the node—by calling itself iteratively until it finds the target node or reach the end. If DFS find the target node, it will append the node to a route list and return. When it reaches the end of one route which is not the correct path, the algorithm will return to the previous layer, remove the last reached node from route list, and try to go along with the other edges. For the DFS algorithm, it only has one layer of vision, so it can only judge whether the reached node is the target node or end node to stop.

   ii. BFS

   The BFS algorithm uses the first-in-first-out (FIFO) feature of the queue to achieve layer-by-layer search. When entering the search algorithm, the first node will be pop out of the queue, and its child nodes will push in which can guarantee that the parent node is traversed first. If the target node is in the queue, then break the BFS loop and return the route.

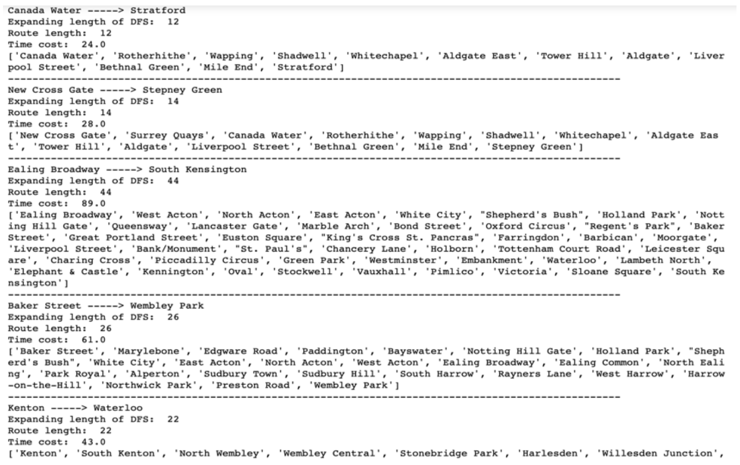
   iii. UCS

   UCS is a BFS based-algorithm. Similar to the BFS, but with more information in the queue structure. UCS uses the priority queue to process search algorithm. The average token time between the two stations is added as a weight to the queue along with its edge. Priority queue will sort the nodes by their weight. Thus, the first pop node will be the node with smallest weight in the UCS which means that it will explore the less cost, or nearest station from the current node.

   b) Compare DFS, BFS, and UCS

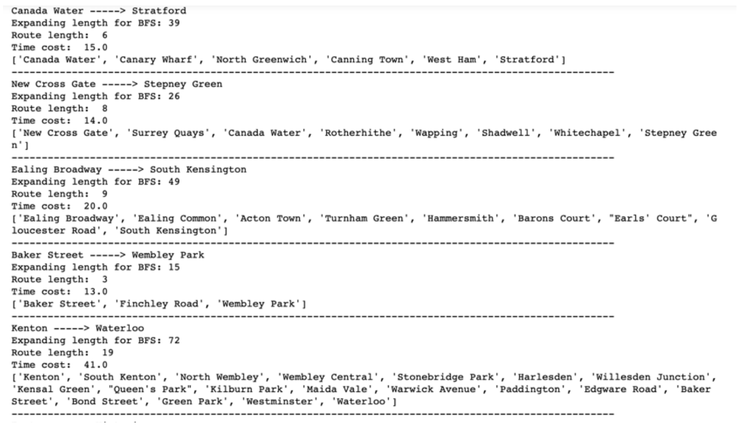
   By testing many different routes among the three search algorithms, the result shows as below.

DFS:



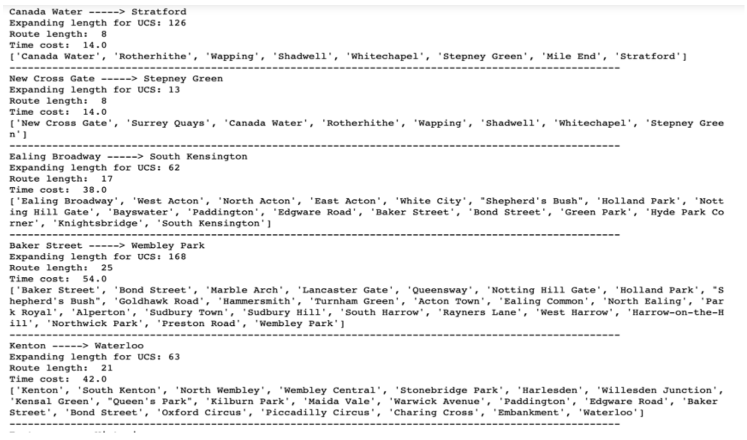
```
Canada Water -----> Stratford
Expanding length of DFS:  12
Route length:  12
Time cost:  24.0
['Canada Water', 'Rotherhithe', 'Wapping', 'Shadwell', 'Whitechapel', 'Aldgate East', 'Tower Hill', 'Aldgate', 'Liver
pool Street', 'Bethnal Green', 'Mile End', 'Stratford']
------------------------------------------------------------------------------------------------
New Cross Gate -----> Stepney Green
Expanding length of DFS:  14
Route length:  14
Time cost:  28.0
['New Cross Gate', 'Surrey Quays', 'Canada Water', 'Rotherhithe', 'Wapping', 'Shadwell', 'Whitechapel', 'Aldgate Eas
t', 'Tower Hill', 'Aldgate', 'Liverpool Street', 'Bethnal Green', 'Mile End', 'Stepney Green']
------------------------------------------------------------------------------------------------
Ealing Broadway -----> South Kensington
Expanding length of DFS:  44
Route length:  44
Time cost:  89.0
['Ealing Broadway', 'West Acton', 'North Acton', 'East Acton', 'White City', "Shepherd's Bush", 'Holland Park', 'Nott
ing Hill Gate', 'Queensway', 'Lancaster Gate', 'Marble Arch', 'Bond Street', 'Oxford Circus', "Regent's Park", 'Baker
Street', 'Great Portland Street', 'Euston Square', "King's Cross St. Pancras", 'Farringdon', 'Barbican', 'Moorgate',
'Liverpool Street', 'Bank/Monument', "St. Paul's", 'Chancery Lane', 'Holborn', 'Tottenham Court Road', 'Leicester Squ
are', 'Charing Cross', 'Piccadilly Circus', 'Green Park', 'Westminster', 'Embankment', 'Waterloo', 'Lambeth North',
'Elephant & Castle', 'Kennington', 'Oval', 'Stockwell', 'Vauxhall', 'Pimlico', 'Victoria', 'Sloane Square', 'South Ke
nsington']
------------------------------------------------------------------------------------------------
Baker Street -----> Wembley Park
Expanding length of DFS:  26
Route length:  26
Time cost:  61.0
['Baker Street', 'Marylebone', 'Edgware Road', 'Paddington', 'Bayswater', 'Notting Hill Gate', 'Holland Park', "Sheph
erd's Bush", 'White City', 'East Acton', 'North Acton', 'West Acton', 'Ealing Broadway', 'Ealing Common', 'North Eali
ng', 'Park Royal', 'Alperton', 'Sudbury Town', 'Sudbury Hill', 'South Harrow', 'Rayners Lane', 'West Harrow', 'Harrow
-on-the-Hill', 'Northwick Park', 'Preston Road', 'Wembley Park']
------------------------------------------------------------------------------------------------
Kenton -----> Waterloo
Expanding length of DFS:  22
Route length:  22
Time cost:  43.0
['Kenton', 'South Kenton', 'North Wembley', 'Wembley Central', 'Stonebridge Park', 'Harlesden', 'Willesden Junction',
```

BFS:



```
Canada Water -----> Stratford
Expanding length for BFS: 39
Route length:  6
Time cost:  15.0
['Canada Water', 'Canary Wharf', 'North Greenwich', 'Canning Town', 'West Ham', 'Stratford']
------------------------------------------------------------------------------------------------
New Cross Gate -----> Stepney Green
Expanding length for BFS: 26
Route length:  8
Time cost:  14.0
['New Cross Gate', 'Surrey Quays', 'Canada Water', 'Rotherhithe', 'Wapping', 'Shadwell', 'Whitechapel', 'Stepney Gree
n']
------------------------------------------------------------------------------------------------
Ealing Broadway -----> South Kensington
Expanding length for BFS: 49
Route length:  9
Time cost:  20.0
['Ealing Broadway', 'Ealing Common', 'Acton Town', 'Turnham Green', 'Hammersmith', 'Barons Court', "Earls' Court", 'G
loucester Road', 'South Kensington']
------------------------------------------------------------------------------------------------
Baker Street -----> Wembley Park
Expanding length for BFS: 15
Route length:  3
Time cost:  13.0
['Baker Street', 'Finchley Road', 'Wembley Park']
------------------------------------------------------------------------------------------------
Kenton -----> Waterloo
Expanding length for BFS: 72
Route length:  19
Time cost:  41.0
['Kenton', 'South Kenton', 'North Wembley', 'Wembley Central', 'Stonebridge Park', 'Harlesden', 'Willesden Junction',
'Kensal Green', "Queen's Park", 'Kilburn Park', 'Maida Vale', 'Warwick Avenue', 'Paddington', 'Edgware Road', 'Baker
Street', 'Bond Street', 'Green Park', 'Westminster', 'Waterloo']
------------------------------------------------------------------------------------------------
```

UCS:



```
Canada Water -----> Stratford
Expanding length for UCS: 126
Route length:  8
Time cost:  14.0
['Canada Water', 'Rotherhithe', 'Wapping', 'Shadwell', 'Whitechapel', 'Stepney Green', 'Mile End', 'Stratford']
------------------------------------------------------------------------------------------------
New Cross Gate -----> Stepney Green
Expanding length for UCS: 13
Route length:  8
Time cost:  14.0
['New Cross Gate', 'Surrey Quays', 'Canada Water', 'Rotherhithe', 'Wapping', 'Shadwell', 'Whitechapel', 'Stepney Gree
n']
------------------------------------------------------------------------------------------------
Ealing Broadway -----> South Kensington
Expanding length for UCS: 62
Route length:  17
Time cost:  38.0
['Ealing Broadway', 'West Acton', 'North Acton', 'East Acton', 'White City', "Shepherd's Bush", 'Holland Park', 'Nott
ing Hill Gate', 'Bayswater', 'Paddington', 'Edgware Road', 'Baker Street', 'Bond Street', 'Green Park', 'Hyde Park Co
rner', 'Knightsbridge', 'South Kensington']
------------------------------------------------------------------------------------------------
Baker Street -----> Wembley Park
Expanding length for UCS: 168
Route length:  25
Time cost:  54.0
['Baker Street', 'Bond Street', 'Marble Arch', 'Lancaster Gate', 'Queensway', 'Notting Hill Gate', 'Holland Park', "S
hepherd's Bush", 'Goldhawk Road', 'Hammersmith', 'Turnham Green', 'Acton Town', 'Ealing Common', 'North Ealing', 'Par
k Royal', 'Alperton', 'Sudbury Town', 'Sudbury Hill', 'South Harrow', 'Rayners Lane', 'West Harrow', 'Harrow-on-the-H
ill', 'Northwick Park', 'Preston Road', 'Wembley Park']
------------------------------------------------------------------------------------------------
Kenton -----> Waterloo
Expanding length for UCS: 63
Route length:  21
Time cost:  42.0
['Kenton', 'South Kenton', 'North Wembley', 'Wembley Central', 'Stonebridge Park', 'Harlesden', 'Willesden Junction',
'Kensal Green', "Queen's Park", 'Kilburn Park', 'Maida Vale', 'Warwick Avenue', 'Paddington', 'Edgware Road', 'Baker
Street', 'Bond Street', 'Oxford Circus', 'Piccadilly Circus', 'Charing Cross', 'Embankment', 'Waterloo']
------------------------------------------------------------------------------------------------
```

After comparison, it is found that due to the different search methods of the algorithm itself, the obtained path and the explored node are quite different.
For DFS, which is depth first as mentioned before, it always goes along with the edge. If the network graph has very many connections, and there are many paths to the target point, DFS can always find the target node along one line. That's why the expanding length of DFS is

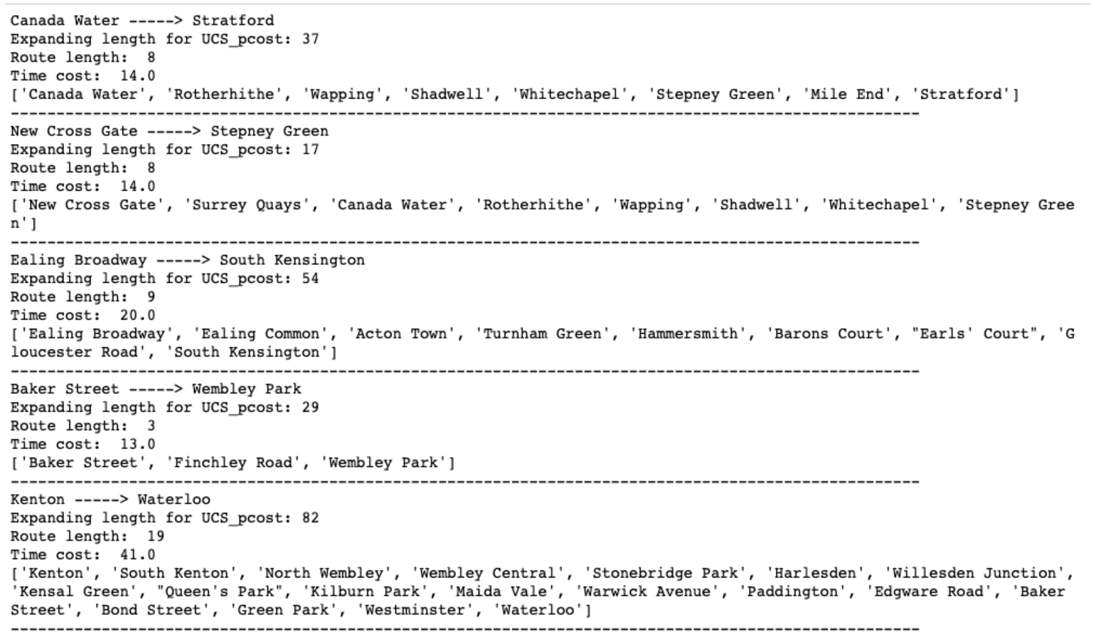
always equal to the Route length. But it may also lead to a high costed result. Take test case 4 which takes from “Baker Street” to “Wembley Park” as an example, the DFS result has expanded 26 nodes and cost an average time of 61. But the BFS only takes 15 nodes to expand and the final route length is 3, the token time of BFS result is 13 which much smaller than the result of DFS. There are only three stations between “Baker Street” and “Wembley Park”, but DFS select a wrong direction and finally came back after a long circle. UCS also has problems similar to DFS. It takes 168 nodes to expand even more than the nodes token by DFS. The reason why is that UCS always expand the least cost node in the priority queue which may drive the route to a wrong direction. To a certain extent, it is also a definition of ‘depth’.

BFS always find the shortest path to the destination, however, when the path searched is very long or the network graph is huge, the expanding nodes will increase dramatically. In case 5, there is like an end-to-end route need to be found, DFS perform well which just keep one direction and expand 22 nodes while BFS expand 72 nodes to find the path.

The avg expanding nodes are 23.6, 40.2 and 86.4, the avg token time are 49, 20.6, 32.4. Among them, BFS has the best performance, with less expansion of nodes (than UCS) to find the least time-consuming path.

c) Extending the cost function

Instead of adding point-to-point token time directly, I use the current path token time as the cost function. Each expanding node will store a path token time from the original node, the lowest cost will be pop first according to the priority queue. With the extended cost function, the avg expanding nodes is 43.8 and the avg time cost is 20.4. Compared with the original UCS, the expansion node of UCS_pcost() is reduced by half, and the time consumption of the final path is also the least of all methods so far.



```
Canada Water -----> Stratford
Expanding length for UCS_pcost: 37
Route length:  8
Time cost:  14.0
['Canada Water', 'Rotherhithe', 'Wapping', 'Shadwell', 'Whitechapel', 'Stepney Green', 'Mile End', 'Stratford']
----------------------------------------------------------------------------------------------------
New Cross Gate -----> Stepney Green
Expanding length for UCS_pcost: 17
Route length:  8
Time cost:  14.0
['New Cross Gate', 'Surrey Quays', 'Canada Water', 'Rotherhithe', 'Wapping', 'Shadwell', 'Whitechapel', 'Stepney Gree
n']
----------------------------------------------------------------------------------------------------
Ealing Broadway -----> South Kensington
Expanding length for UCS_pcost: 54
Route length:  9
Time cost:  20.0
['Ealing Broadway', 'Ealing Common', 'Acton Town', 'Turnham Green', 'Hammersmith', 'Barons Court', "Earls' Court", 'G
loucester Road', 'South Kensington']
----------------------------------------------------------------------------------------------------
Baker Street -----> Wembley Park
Expanding length for UCS_pcost: 29
Route length:  3
Time cost:  13.0
['Baker Street', 'Finchley Road', 'Wembley Park']
----------------------------------------------------------------------------------------------------
Kenton -----> Waterloo
Expanding length for UCS_pcost: 82
Route length:  19
Time cost:  41.0
['Kenton', 'South Kenton', 'North Wembley', 'Wembley Central', 'Stonebridge Park', 'Harlesden', 'Willesden Junction',
'Kensal Green', "Queen's Park", 'Kilburn Park', 'Maida Vale', 'Warwick Avenue', 'Paddington', 'Edgware Road', 'Baker
Street', 'Bond Street', 'Green Park', 'Westminster', 'Waterloo']
----------------------------------------------------------------------------------------------------
```

d) Heuristic algorithm

```python
def heuristic(curp, endp, cost):
    '''
        add the information of Zone, and if the two stations are in the same Line
        the final cost = original time cost + sameline cost + difZone cost*5
    '''
    sameline = 3
    for line in nx_net.nodes[curp]['Line']:
        if line in nx_net.nodes[endp]['Line']:
            sameline = 0

    curZone = int(nx_net.nodes[curp]['Zone1'][0])
    endZone = int(nx_net.nodes[endp]['Zone1'][0])
    difZone = abs(curZone - endZone)

    cost = cost + sameline + difZone*5

    return cost
```

By observing the problems of algorithms such as BFS, DFS and UCS, I implemented a heuristic function to solve the problem of expansion direction. The function cost is based on the path cost and added two other items: sameline and difZone. According to the advantages of DFS, stations on the same line will always be given priority. So, I set up a parameter sameline to represent. If the target station is in the same line of current station, the cost will set to be 0, otherwise will be 3. The reason why 3 is that the average of total edges token time is 2.3. I assume that the two stations on the same line will always take less time than transferring or staying at different line stations, at least less than an average time level.
By adding Zone information to enlighten the algorithm find the correct direction. I have observed that all zones are strictly adjacent to each other which means you can not jump two zone by one step. If you want to reach the zone 5 from the zone 1, you have to go through zone 2, 3, 4 at first. This is a kind of direction. So I calculate the difference of Zone and give it a big weight because the cost of wrong direction is much more than the cost of travel time.
After experiments the default sameline cost finally set to be 3 and difZone to times 5.

```
Canada Water -----> Stratford
Expanding length for UCS_Heuristic: 16
Route length:  6
Time cost:  15.0
['Canada Water', 'Canary Wharf', 'North Greenwich', 'Canning Town', 'West Ham', 'Stratford']
----------------------------------------------------------------------------------------------------
New Cross Gate -----> Stepney Green
Expanding length for UCS_Heuristic: 16
Route length:  8
Time cost:  14.0
['New Cross Gate', 'Surrey Quays', 'Canada Water', 'Rotherhithe', 'Wapping', 'Shadwell', 'Whitechapel', 'Stepney Gree
n']
----------------------------------------------------------------------------------------------------
Ealing Broadway -----> South Kensington
Expanding length for UCS_Heuristic: 35
Route length:  9
Time cost:  20.0
['Ealing Broadway', 'Ealing Common', 'Acton Town', 'Turnham Green', 'Hammersmith', 'Barons Court', "Earls' Court", 'G
loucester Road', 'South Kensington']
----------------------------------------------------------------------------------------------------
Baker Street -----> Wembley Park
Expanding length for UCS_Heuristic: 10
Route length:  3
Time cost:  13.0
['Baker Street', 'Finchley Road', 'Wembley Park']
----------------------------------------------------------------------------------------------------
Kenton -----> Waterloo
Expanding length for UCS_Heuristic: 45
Route length:  19
Time cost:  41.0
['Kenton', 'South Kenton', 'North Wembley', 'Wembley Central', 'Stonebridge Park', 'Harlesden', 'Willesden Junction',
'Kensal Green', "Queen's Park", 'Kilburn Park', 'Maida Vale', 'Warwick Avenue', 'Paddington', 'Edgware Road', 'Baker
Street', 'Bond Street', 'Green Park', 'Westminster', 'Waterloo']
----------------------------------------------------------------------------------------------------
```

The avg expanding nodes is 24.4 and the avg time cost is 20.6 which is the best performer of all algorithms.
Due to the limited time, the heuristic function cannot be optimized. It is only when the node to be explored is changing zones to know whether the direction is correct. In future research, zone-related attributes should be added to the edges of the graph. For example, if you follow this edge, the zone will increase. Then the direction will be known at first.

2. Adversarial search

a) Play optimally

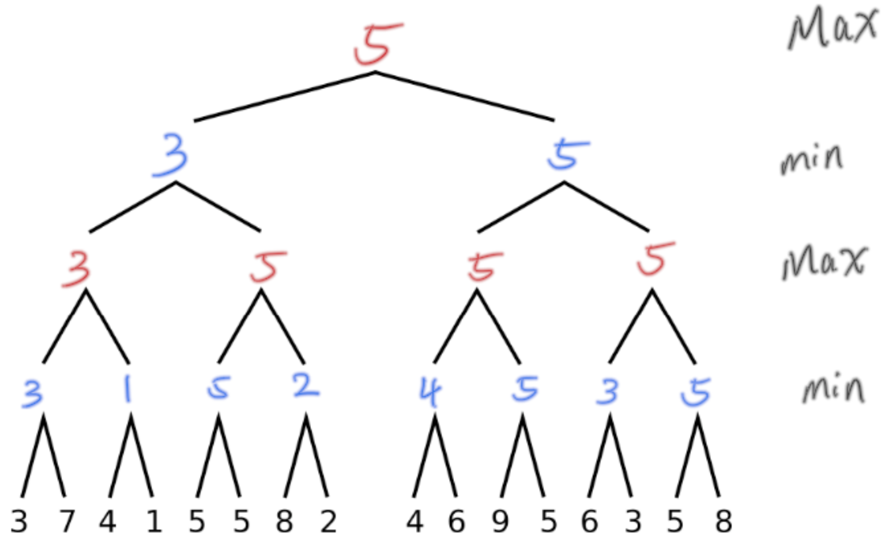


Figure 1: Tree of the game

b) Play optimally, quicker thinking (α-β pruning)

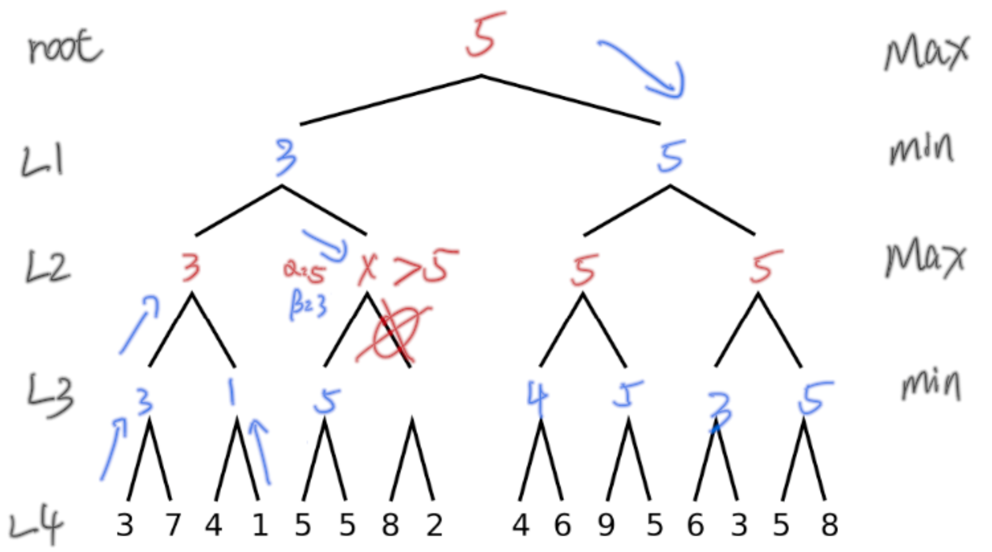


Figure 1: Tree of the game

First, without hesitation, enter the (3, 7) of the left subtree of L4, and then the min player in L3 will choose 3 and same process with the right subtree. Then the L2 max play would choose 3, and L1 player will compare it with the X node. It is found that L3 of X left subtree will be 5 and no matter what the X right subtree is, the X is at least 5 according to the max player. So L1 will not choose right subtree of X node which means it is pruned. Then perform the same operation on the subtree on the right and the final root value will equal to 5.

c) Entry fee to play

i. What are the ranges of values for x that will be still worth playing the game for the MAX player (to enter the game at all)?

According to the minimax tree above, if Max player has to pay x units, it means that the terminal leaf nodes are all need to minus x. Because the x is a constant value, it would not affect the tree decision. So, the subtree of root Max play would be (3-x, 5-x). Thus, when $x < 5$ the game will still worth to play for Max player.

ii. For a fixed value of x, do you prefer to be the first player (MAXimiser) or the second player (MINimiser)? Very briefly explain your answer.

The first-hand must-win theory is applicable in many decision-making games. For min player, the number of decisions he made is always one step behind max player or the same as max player. I tend to treat the entry fee as a compensation for the min player, which also means that the max player has an advantage in the decision-making process. So in general I am willing to be a max player. Specifically, if $x<5$ the Max player will be better, or if $x>9$ the min player will be better.