BLG336E – Analysis of Algorithms 2, Spring 2018

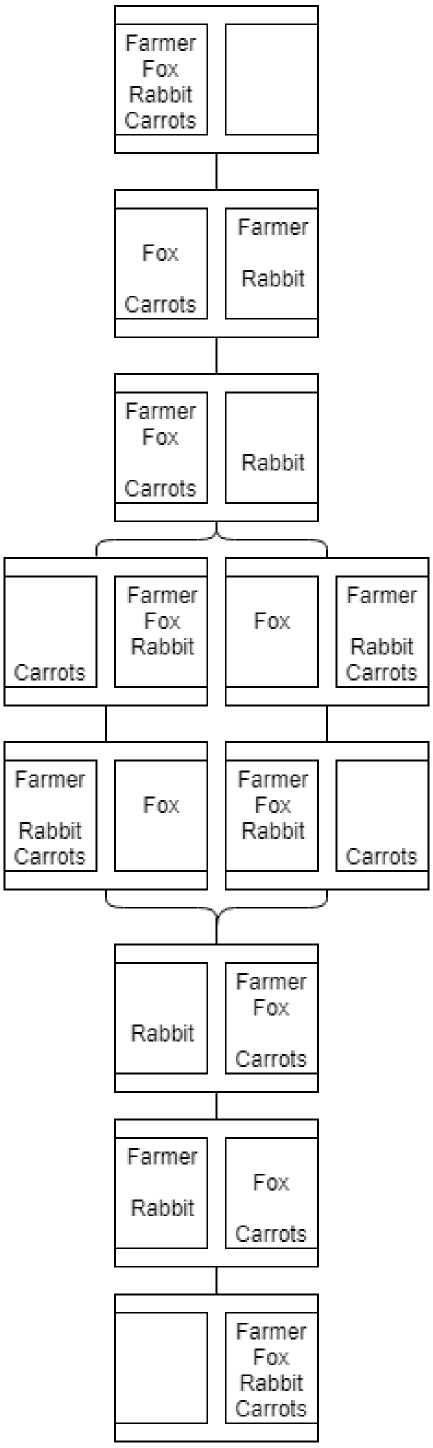
Project 1 Report

Yunus Güngör 150150701

1)Solution to problem

Each allowed step represented as a node in a graph. Each step includes location of the given entities, this graph can be found in figure 1.

Each node has left and right sides of the river, and entities listed in them. Constructed graph is an undirected graph that connects each step with possible steps. Graph can be undirected because, problem can be solved in reverse because it wouldn't differ depending to side of the river. Invalid steps is not represented and a search algorithm used to find sufficent solution. In this case, BFS and DFS algorithms used.

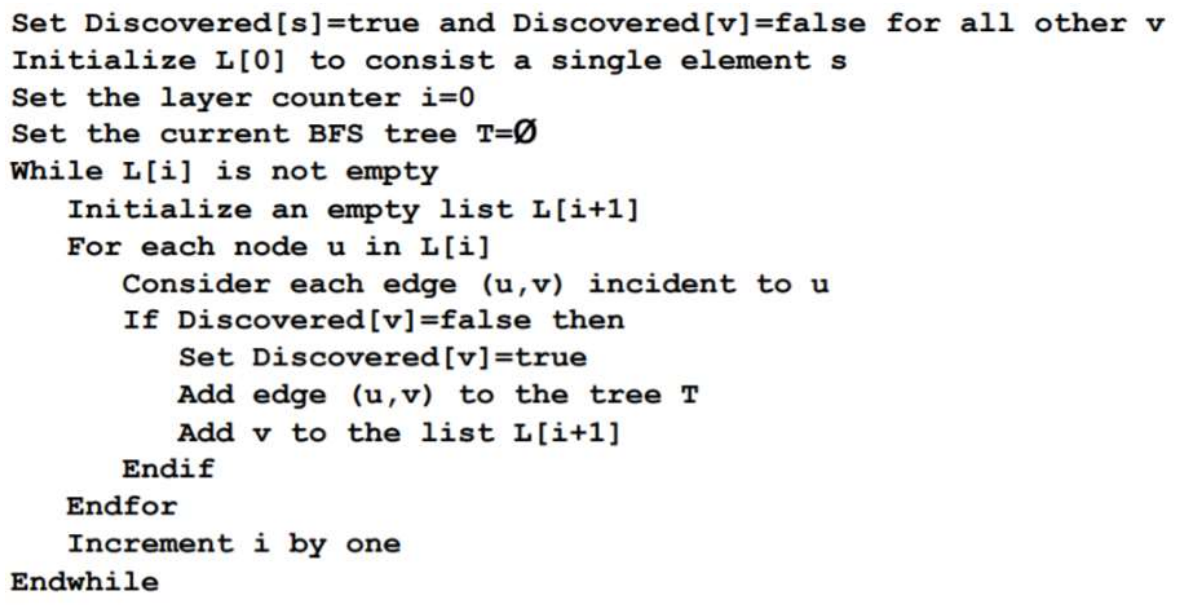


Farmer Fox Rabbit Carrots
Fox Carrots | Farmer Rabbit
Farmer Fox Carrots | Rabbit
Carrots | Farmer Fox Rabbit
Fox | Farmer Rabbit Carrots
Farmer Rabbit Carrots | Fox
Farmer Fox Rabbit | Carrots
Rabbit | Farmer Fox Carrots
Farmer Rabbit | Fox Carrots
Farmer Fox Rabbit Carrots

Figure 1 – Constructed Graph

2)Search Algorithm

I have used the algorithm from our course materials.

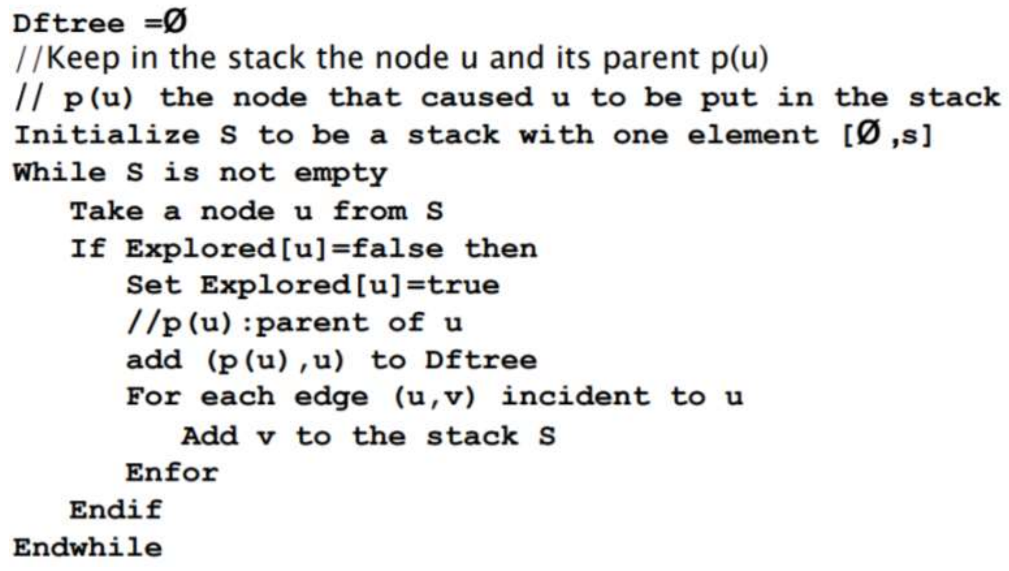


```
n  Set Discovered[s]=true and Discovered[v]=false for all other v
1  Initialize L[0] to consist a single element s
1  Set the layer counter i=0
1  Set the current BFS tree T=Ø
n  While L[i] is not empty
1     Initialize an empty list L[i+1]
n     For each node u in L[i]
1        Consider each edge (u,v) incident to u
1        If Discovered[v]=false then
1           Set Discovered[v]=true
1           Add edge (u,v) to the tree T
1           Add v to the list L[i+1]
         Endif
      Endfor
      Increment i by one
   Endwhile
```

Figure 2 - BFS Algorithm Pseudocode, taken from course slides

Since each node can be added to layers only once, and each node can have maximum n edges. Algorithm can be bounded by $O(n^2)$. From pseudocode, it can be bounded by $O(n+3+n(6+n))$ which is equal to $O(n^2)$.

If we consider edges, complexity is $O(m+n)$, m is the number of edges. Since each node added to the layers processed with their edges (for loop), complexity becomes $O(m)$. However, maintaining a discovered array costs $O(n)$. Which causes algorithm to have $O(m+n)$.



```
1  Dftree =Ø
   //Keep in the stack the node u and its parent p(u)
   // p(u) the node that caused u to be put in the stack
1  Initialize S to be a stack with one element [Ø,s]
n  While S is not empty
1     Take a node u from S
1     If Explored[u]=false then
1        Set Explored[u]=true
         //p(u):parent of u
1        add (p(u),u) to Dftree
n        For each edge (u,v) incident to u
1           Add v to the stack S
         Enfor
      Endif
   Endwhile
```

Figure 3 - DFS Algorithm Pseudocode, taken from course slides

Since each node and each node's neighbors added to the stack, and stack processed afterwards, complexity is $O(n^2)$. An explored array maintained with the cost of $O(n)$ but it is smaller than $O(n^2)$. From pseudocode algorithm has $O(2+n(4+n))$ which is equal to $O(n^2)$.

If we consider edges, complexity is $O(m+n)$, m is the number of edges. Each node will be added to the stack whenever an edge found. Since adjacency list has 2 of each edges. Complexity will be $O(2m)$. However, maintaining a discovered array costs $O(n)$. Which causes algorithm to have $O(m+n)$.

3)Discovered Nodes List

In depth first search, maintained stack has multiple entries of the same node, therefore it is necessary to maintain discovered nodes list to prevent multiple nodes in the same tree.

4)Bipartite Graph

Constructed graph is a bipartite graph. Because graph doesn’t have any cycles and there‘s no edges that joins two nodes on the same layers.

5)Solutions and Analysis

Only BFS or DFS algorithm’s resources used. The pre-build tree’s resources did not count.

BFS, visited 20 nodes and kept 10 nodes in memory (Layers). It took $2{,}6.\,10^{-7}$ seconds to complete on average.

DFS. visited 19 nodes and kept 8 nodes in memory (Stack). It took $5{,}1.\,10^{-7}$ seconds to complete on average.

Both algorithms find solution in 8 steps.