Solving Werewolf Problem
By Eakawat Tantamjarik 5929444

Werewolf problem

• We have villagers in a village

• Most of them are each other’s relative

• Werewolves are among those villagers

• Werewolves are never kill their ancestors and descendants
Werewolf problem (2)

- The problem provides
  - The set of death villagers killed by werewolves
  - The number of villagers in the village
  - All relationships between villagers

- The problem wants the set of villager who suspect to be werewolf
Problem Modelling

• Use graph to represent the entire village

2 villagers (A, B) while A is ancestors of B
Problem condition

- Werewolves never kill their ancestors and descendants

B is killed by werewolf

Z is suspect to be werewolf because Z is not ancestor or descendant of B
Idea to solve problem

• Find out all the ancestors and descendants of death villagers

• Other villagers not in the list of above are considered to be werewolf
Graph representation for werewolf problem

• Use modified **Adjacency List** representation for node

```python
class Villager(object):
    ancestors = list()
    descendants = list()
```

• Easier to directly locate all ancestors and descendants when the node (villager) is known
The algorithm

WEREWOLF(N)

\[ Q = \phi \]

for each death \( s \in N \)

\[ s.visited\_ancestor = true \]

ENQUEUE(Q, s)

while \( Q \neq \phi \)

\[ u = DEQUEUE(Q) \]

for each \( v \in u.ancestors \)

\[ \text{if } v.visited\_ancestor = false \]

\[ v.visited\_ancestor = true \]

ENQUEUE(Q, v)

for each \( s \in N \)

ENQUEUE(Q, s)

\[ s.visited\_descendant = true \]

while \( Q \neq \phi \)

\[ u = DEQUEUE(Q) \]

for each \( v \in u.descendants \)

\[ \text{if } v.visited\_descendant = false \]

\[ v.visited\_descendant = true \]

ENQUEUE(Q, v)

for each \( s \in N \)

\[ \text{if } v.visited\_ancestor = false \text{ and } v.visited\_descendant = false \]

ENQUEUE(Q, s)

\( Q \) is now contains the suspect villagers
The algorithm (2)

- It is a modified BFS (breath-first-search) for graph traversal

- Doing traversal 2 times
  - one for ancestor
  - another for descendant

- All nodes (villager) not visited by those 2 traversals are werewolf
Running time analysis

• Loop though all death villagers can be at most the number of all villagers- $O(V)$

• While loop only run for once for each villager because of the ancestor flag - $O(V)$

• Ancestor list is iterate once for each villager, at most equal to number of all edges - $O(E)$

• Do the same thing for descendant part - All above multiply by 2

• Lastly loop through all the villagers - $O(V)$

• $O(2V + 2V + 2E + V) -> O(V + E)$
Proof of correctness

• **Claim 1** - All normal villagers need to be visited at least once
  
  \( \text{ENQUEUE}(Q, v) \) will add node to be visited when the node is either ancestor or descendant of death villager by the flag \( \text{visited_ancestor} \) and \( \text{visited_descendant} \)

• **Claim 2** - All ancestors and descendants of death villager can be reached from death villager
  
  From claim 1 villager will be visited by either from ancestor or descendant relationship or both if the algorithm cannot find villager anymore to add to \( Q \) that mean all ancestors or descendants are already found because from claim 1 the villager will not repeat itself in each ancestor or descendant part which is the result of flag (\( \text{visited_ancestor} \), \( \text{visited_descendant} \))