HW7

Xinhao Luo

Tuesday 12th May, 2020

1 DPV Problem 8.2

- Define f(G) return true when the graph G has a Rudrata Path, false otherwise.
- f(G) should take polynomial time.

Precedures

```
if f(G) == false:
    return false;
else:
    for edge E in Graph G:
        Remove edge E from Graph G
        if f(G_1) == false:
            Add edge E to Graph G
return G
```

Each time this algorithm takes one edge out of the graph to see if it can still form a Rudrata Path. If true, it is telling that that edge is not necessary for building the path and can be safely removed, or the edge must be in the final solution.
Runtime This algorithm loop all edge once and since f(g) takes polynomial time, the overall time complexity should still remain polynomial time

2 DPV Problem 8.4

a) Proving an NP problem means that the solution of the problem can be checked in polynomial time.

- Define solution Graph S has v vertices, and the goal g

Procedures

```
if v != g:
    return false
for vertice i in S:
    for vertice j in S:
        if i != j:
            if there is no edge between i and j:
                return false

return true
```

The CLIQUE means for every pair of vertices in graph, there will be an edge connecting them. Following this rule and also verify the goal g match with the graph will prove this solution is correct
Runtime
The double for-loop each loop through the graph's vertices v, so the overall time complexity will be $O(|V^2|)$, where v is the number of vertices in the solution graph S, which is a polynomial time algorithm.

b) To prove CLIQUE - 3 is an NP-Complete Problem, we need to show that other search problem can reduce to it. Even though we know the CLIQUE is an NP problem, the prove is actually reduce CLIQUE - 3 to CLIQUE, which does not prove the NP completeness of CLIQUE - 3.
In order to reduce CLIQUE to CLIQUE - 3 problem, we choose an graph G with vertice of degree ¡= 3. Showing that the solution S, satisfied CLIQUE iff S satisfied CLIQUE - 3. Clearly for graph G, CLIQUE and CLIQUE - 3 share the same solution.

3 DPV Problem 8.13

a) This problem can be solve in polynomial time

- Define spin(G) used to find a spinning tree (similar to Kruskal's, without the minimum constraint), return false if no found in the Graph G

Procedures

```
let G_1 = G without vertices in set L, alone with corresponded edges
let edges[] = removed edges from the previous statements

let G_2 = spin(G)

if G_2 == false:
    return false

let vertices[] = all vertices in G_2
for vertex v in U:
    edge e = an edge in edges[] that connects v and any vertex in vertices[]
    if e != None:
        add e to G_2
        remove e from edges[]
    else:
        return false

return G_2
```

First we remove all vertices in set L and find the spinning tree, then we try to add back edges and connect vertices in set L to the existing vertices of the spinning tree, which makes them definitely the leaf of the spinning tree.
Runtime Copying graph is $O(\|V\| + \|E\|)$, while spinning tree finding takes $O(\|E\| log\|V\|)$ which is about the same as the Kruskal's Algorithm. The last loop takes at most $O(\|V\|)$ for loop through U and find edge for it. Each edge can visited only once (find with map implementation). So the overall time complexity will be $O(\|V\| + \|E\|)$

b) This is a NP-Complete Problem

Check Procedures
(a) Run DFS and ensure every node can be reached and no loop in graph G, return false if not satisfied.
(b) Find all the leaf node in the graph during DFS and collect them into set M
(c) if set M is the same as the set L, return true, false otherwise

DFS takes $O(\|V\| + \|E\|)$, which is polynomial time

Reduction Here we will show that for graph G and two vertices x, y graph G builds a valid Rudrata Path iff it satisfied spanning tree with leaf x, y.
Procedures From any pair of vertices x, y in graph G, build the Rudrata Path from point x to y has the same result of setting x, y as the leaf set and getting the spanning set of graph G.
Since a spanning tree will touch all vertices exactly once, if at the same time it has only two leaves, a Rudrata Path will be satisfied. On the other hand, if a Rudrata Path of vertices x, y built, it satisfied the condition of Spanning tree with leaf x, y as well.
Runtime Polynomial time as the procedure only contains graph merging and splitting, which are constant.

c) This is a NP-Complete Problem

Check Procedures Similar to part b), but the set comparison changed from equals to L.containsAll(M). False if comparison failed.
The runtime should remain the same.

Reduction Similar to part b), using Graph G, vertices x, y as example.
With Rudrata Path start and end with x, y, vertices x, y is also part of the set L, satisfied the condition of Spanning tree using vertices in Set L.
With Set L contains x,y, the generated spanning tree graph also contains the Rudrata path due to the nature of undirected graph.
Runtime Similar to part b), spiltting and merging aare constant and is polynomial time.