COL351
Assignment 1

Mallika Prabhakar, 2019CS50440
Sayam Sethi, 2019CS10399

August 2021

Contents

1 Question 1

Let G be an edge-weighted graph with n vertices and m edges satisfying the condition that all the edge weights in G are distinct.

1.a Unique MST

Question 1.a

Question. *Prove that G has a unique MST.*

Proof. We will prove this by induction on the size of G using an idea similar to Kruskal's algorithm discussed in the class.
Hypothesis:

$$h(n) : \forall\ G = (V, E) :\ |V| = n \implies MST(G)\ \textit{is unique} \tag{1}$$

Base case: $n = 1$ is true since there is no edge and $MST(G) = (V, \phi)$ is unique.

Induction Step: Assume $h(n-1)$ is true for $n \geq 2$, now for $h(n)$:
(Note: This proof assumes each edge to be an unordered pair of vertices)

Consider Kruskal's algorithm,

Algorithm 1 Recursive MST Routine – Kruskal's algorithm

```
1: procedure MST(G)
2:     e_0 ← (x, y) be edge with least weight
3:     H ← G
4:     remove x, y from H and add new vertex z
5:     for all v such that v is neighbour of x or y do
6:         add (v, z) to H
7:         wt(v, z) ← min(wt(v, x), wt(v, y))
8:         if wt(v, x) < wt(v, y) then
9:             map(v, z) ← (v, x)
10:        else
11:            map(v, z) ← (v, y)
12:        end if
13:    end for
14:    T_H ← MST(H)
15:    T_G ← (V, {e_0})
16:    for all e ∈ T_H do
17:        if e is not incident on z then
18:            add e to T_G
19:        else
20:            add map(e) to T_G
21:        end if
22:    end for
23:    return T_G
24: end procedure
```

In the above algorithm, it is clear that H has $n-1$ vertices. Thus, by our assumption, $h(n-1)$ is true and hence T_H is unique. Also, we know that T_G is a valid MST, from the correctness of Kruskal's algorithm. Now, assume by contradiction that T_G is not unique. Then there exists an MST, say $T' \neq T_G$.

Claim 1.1. *e_0 cannot be in T'*

Proof. This is because, if e_0 were in T', then $T \setminus \{e_0\} \neq T' \setminus \{e_0\}$ and thus, there would be two different MSTs for H which would be a contradiction to our assumption. Thus, $e_0 \notin T'$. □

Consider the path from x to y in T'. Since $e_0 = (x, y)$ is not present in T', there exists a different path, say $P = (f_1, f_2 \cdots, f_k)$ where $f_i \in E(T'), 1 \leq i \leq k$. We know that

$wt(f_i) > wt(e_0), 1 \leq i \leq k$.
Swap any of the f_i with e_0 and let the subgraph formed be T'', i.e., $T'' = T' \setminus \{f_i\} \cup \{e_0\}$.
We know T'' is a spanning tree of G since $V(T'') = V(G)$ and there are no cycles formed on performing the swap operation (this can be proven using contradiction as discussed in the lecture).
Now, consider the weight of T'':

$$\begin{aligned} & wt(T'') = wt(T') - wt(f_i) + wt(e_0) \\ \implies & wt(T'') < wt(T') \end{aligned} \tag{2}$$

We have shown that the total weight of T'' is lesser than the weight of T'. However, this is a contradiction to the fact that T' is the MST of G. Thus our assumption that T_G is not the unique MST of G was wrong. Therefore, $h(n)$ is true.

This completes the induction and the proof that *if all edge weights in a graph are distinct, then its MST is unique.* □

1.b Algorithm Sketch

Question 1.b

Question. *If it is given that G has at most $n+8$ edges, then design an algorithm that returns a MST of G in $O(n)$ running time.*

Solution. The idea is to use the previous result along with the fact that the number of edges to be removed to form a spanning tree is atmost $(n+8)-(n-1) = 9$, assuming that G was initially connected (else no MST exists). The algorithm is as follows:

Algorithm 2 Compute MST for 1.b

```
1: procedure MST(G)
2:     if |E(G)| equals |V(G)| − 1 then
3:         return G                          ▷ since G is acyclic and hence a tree
4:     end if
5:     C ← findCycle(G)
6:     e ← edge with largest weight in C
7:     remove e from G
8:     T_G ← MST(G)
9:     return T_G
10: end procedure
```

The procedure *findCycle* calls a DFS function on G which uses graph colouring and returns the first cycle it finds:

Algorithm 3 *findCycle*

```
1: procedure FINDCYCLE(G)
2:     v ← any vertex of G
3:     colour ← map of vertices initialised to zero
4:     parent ← map of vertices initialised to null
5:     (u, v) ← dfs(G, v, colour, parent, null)
6:                          ▷ returns the bottommost and topmost vertex of the cycle
7:     C ← empty array of edges
8:     add (u, v) to C
9:     while u ≠ v do
10:        add (u, parent(u)) to C
11:        u ← parent(u)
12:    end while
13:    return C
14: end procedure
```

The *DFS* function looks as follows:

Algorithm 4 Identify cycle using colouring and DFS

```
1: procedure DFS(G, v, colour, parent, p)
2:     parent(v) ← p
3:     colour(v) ← 1
4:     for all u such that u is neighbour of v in G do
5:         if colour(u) is 2 then
6:             return (u, v)
7:         else if colour(u) is 0 then
8:             value ← dfs(G, u, colour, parent, v)
9:             if value is not null then
10:                return value
11:            end if
12:        end if
13:    end for
14:    colour(v) ← 2
15: end procedure
```

□

2 Question 2

2.a Optimal Huffman Encoding

Question 1.b

Question. *What is the optimal binary Huffman encoding for n letters whose frequencies are the first n Fibonacci numbers? What will be the encoding of the two letters with frequency* 1*, in the optimal binary Huffman encoding?*

Solution. We begin by observing the property of Fibonacci numbers:

$$\begin{aligned} f(n) &= f(n-1) + f(n-2)\ \forall n \geq 3 \\ \text{and, } f(1) &= f(2) = 1 \end{aligned} \tag{3}$$

Now, consider an alphabet $A = (a_1, a_2, \ldots, a_n)$ such that it has a frequency vector $V = (1, 1, \ldots, f(n))$. We have the following claim for such a situation:

Claim 2.1. *The Huffman tree is unique and a_n gets encoded using a single bit.*

Proof. We will prove the claim using induction on n. □

□

2.b two point two

file for 2b

3 Question 3

3.a three point one

file for 3a

3.b three point one

file for 3b