COL351: Assignment-1

Abhinav Jain
2019CS10322
Himanshu Gaurav Singh
2019CS10358

September 5, 2021

Problem 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.

a. Prove that G has a unique MST.
Proof We proceed by contradiction. Assume that G has two different minimum spanning trees T and T'. Since $|T| = |T'|$, $T \nsubseteq T'$ and $T' \nsubseteq T$. Thus, there must exist atleast an edge that lies in exactly one of $|T|$ or $|T'|$. Consider such edge $e = (u, v)$ with minimum cost(since costs are distinct, e is unique). WLOG, assume that $e \in T$. Since T' is spanning, u and v are connected in T'. Consider the (unique) path $P = u, x_1, x_2, ...x_k, v$ in T'. Observe that since T is a tree, not all edges of P lie in T, otherwise $C = P \cup (u, v)$ forms a cycle in T. Thus, there must exist edge $e' \in P$ not in T. Hence, e' lies in T' but not in T. Since e was chosen as the minimum such edge, $w(e') > w(e)$. Now consider $T'' = T'/e' \cup e$.

Claim: T'' is a spanning tree of G.
Proof Observe that the vertices u' and v'(endpoints of e') are connected in T'' because all the edges of C(to which e' belongs) except e' are in T'' . Consider any two vertices(say x,y) connected in T'. If the unique path from x to y in T' does not contain e' then x, y are connected in T'' by the same path. In the other case, if the path is(WLOG), $x...u', v'..y$, then x, y are connected through $x \sim u' \sim v' \sim y$. This implies that all vertices connected in T' are also connected in T''. Since, T' is spanning, T'' is also spanning.
Also we have $|T''| = |T'| = n - 1$ by construction. Since, T'' is a connected graph and has $n - 1$ edges, it must be a tree. Hence, T'' is a spanning tree.

Observe that, $w(T") = w(T') - w(e') + w(e) < w(T')$. Hence, T' cannot be a minimum spanning tree. Hence, contradiction.

b. 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.
The following algorithm gives the MST in $O(n)$ time. It assumes that G is connected beforehand. Consider $G.V$ and $G.E$ be the set of vertices an edges of G respectively.

Algorithm 1 $MST(G)$

if $G.E = G.V - 1$ **then**
return G
else
$T \leftarrow DFS\_TREE(G)$
Choose an edge $e = (u, v)$ such that $e \in G$ but $e \notin T$
Consider the path $P \in T$ from u to v.
$e' \leftarrow$ the most expensive edge e' in the cycle $C = P \cup e$
return $MST(G - \{e'\})$
end if

The following lemma will help us in proving the correctness of the algorithm.

Lemma The maximum weight-edge of a cycle in an undirected graph with distinct edge-weights does not belong to any minimum spanning tree.

Proof Consider an undirected graph G with distinct edge-weights. G has a unique MST T.Choose any arbitrary cylce C from G. WLoG, assume that maximum edge of C(say $e = (u, v)$) lies in T.
Consider $F = T - \{e\}$. Observe that F will be a forest with two trees(say T_1 and T_2). Since, T is spanning, all vertices of G lie in exactly one of T_1 or T_2. Also, u and v must belong to different subtrees.
WLoG, assume that $u \in T_1$ and $v \in T_2$. Consider the path $P = C/\{u, v\}$ in T. P will be of the form $x_0(= u), x_1, x_2, x_3.., x_k(= v)$. Consider the first edge e_0 in the path from u to v both of whose endpoints lie in different subtrees of F. Note that one such edge must exist since the ends of the path themselves lie in different subtrees.
Consider $T_0 = T/e \cup e_0(= T_1 + T_2 + e_0)$. Since T_1 and T_2 together span all the vertices , T_0 also spans all the vertices. Also, T_0 is a acyclic since T_1 , T_2 are subtrees and ednpoints of e_0 belong to different subtrees. Hence T_0 is a spanning tree.
Observe that $w(T_0) = w(T) - w(e) + w(e_0) < w(T)$, since e was maximum weight-edge in C. This contradicts the fact that T was the MST. Hence proved.

Proof of correctness. Consider an un-directed connected graph G with distinct weight-edges. Define $G' = G - e'$ where e' is the most expensive edge in one of the cycles in G if there exists some cycle in G. We will prove that $MST(G) = MST(G')$ if G is cyclic and $MST(G) = G$ itself otherwise, assuming it is connected. Call OPT_G to be the MST of G.
Observe that, if G is acyclic, then G is itself a tree(since it is connected) and cannot be reduced further, hence it is itself the MST.
If the G is cyclic then G' is a connected graph, because a cycle edge is removed the endpoints of which are still connected through the remaining portion of the original cycle. Thus, the MST of G' will also span G. Hence, $w(OPT_G)$ $\leq$w($\text{OPT}_{G'}$). $-$ **1**
Now, consider OPT_G. From the lemma proved above, $e' \notin OPT_G$. Hence all the edges in OPT_G also belong to G' which means OPT_G is a spanning tree of G'. Thus, $w(OPT_{G'}) \leq w(OPT_G)$. $-$ **2**.
From 1 and 2 above, $w(OPT_G) = w(OPT_{G'})$.
Hence, proved.

Time complexity Observe that G' **has exactly the same number of vertices as** G **and has exactly one less edge than** G. Also, the algorithm terminates when $G.E = G.V - 1$. Initially $G.E \geq G.V - 1$, since G is connected. If $G.E == G.V - 1 + k$ edges, $k \geq 0$, then there are k recursive calls before termination. Each call to DFS_TREE takes $O(n + m)$ time. If we compute the parent of each node in the DFS-tree during the DFS and mark tree edges, we can choose non-tree edges and iterate through the cycle(C) edges in θ(number of edges in the path(P)) $\sim O(n + m)$ time.
Thus the complexity will be $O(k * (m + n))$. Observe that since $m \leq n + 8$, $k \leq 9$. $O(k * (m + n)) \sim O(9 * (n + n + 8)$ $\sim O(18 * n + 72) \sim O(n)$.

Problem 2

a. 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?

Ans. First we state and prove the following lemma which will be useful in constructing an optimal Huffman coding.

Lemma

$$\sum_{k=1}^{n} F(k) = F(n+2) - 1$$

where $F(k)$ denotes the k^{th} Fibonacci number.

Proof *Base Case* $(n = 1)$

$$\sum_{k=1}^{1} F(k) = F(1) = 1 = 2 - 1 = F(3) - 1$$

Inductive Step $(n \longrightarrow n+1)$

$$\sum_{k=1}^{n+1} F(k) = \sum_{k=1}^{n} F(k) + F(n+1) = (F(n+2) - 1) + F(n+1) = F(n+3) - 1$$

Claim

In an optimal huffman encoding, the letter with frequency $F(k)$ $(1 < k \leq n)$ would be encoded as $0^{n-k}1$ and the letter with frequency $F(1)$ would be encoded as 0^{n-1} $(n > 1)$.

Proof

We will proceed by induction.

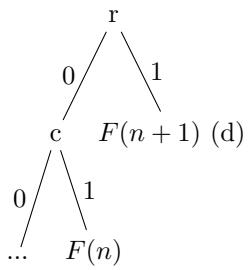
Base Case $(n = 2)$ In this case we have only two characters to encode and the frequency of both is 1, so we can encode one of them as 0 and other as 1. Hence the base case is true.

Inductive Step $(n \longrightarrow n+1)$ In order to construct the Huffman encoding, at each stage we greedily combine the two lowest character frequency nodes into a new parent node with character frequency as sum of the character frequency of the two children nodes.

By the above lemma we have $\sum_{k=1}^{n-1} F(k) = F(n+1) - 1 < F(n+1)$ and since $F(n) < F(n+1)$, during any stage but the final, there will always be two elements present with frequencies smaller than $F(n+1)$. Thus, the frequencies from $F(1)....F(n)$ combine together till the penultimate step to give a subtree of the final tree whose structure is defined by the induction hypothesis.

In the last stage of Huffman coding, we will combine the node with character frequency $F(n+1)$ (call this node d) with the node that is obtained by combining characters with first n frequencies (call this node c) resulting in the parent root node r. Suppose we assign label 0 to edge rc and label 1 to edge rd. So the character with frequency F(n+1) is encoded as 1. Now by inductive hypothesis we know the structure of subtree rooted at node c, hence the encoding of the character with frequecny $F(k)$ $(0 < k < n+1)$ can be obtained by prefixing a 0 to $0^{n-k}1$ i.e its resulting encoding is $0^{n+1-k}1$.

Hence by PMI our claim holds.



b. Suppose you aim to compress a file with 16-bit characters such that the maximum character frequency is strictly less than twice the minimum character frequency. Prove that the compression obtained by Huffman encoding, in this case, is same as that of the ordinary fixed-length encoding.

Ans. We will prove this claim for n-bit characters. Let $f(i)$ denote the i^{th} largest frequency. First we prove the following useful lemmas :

Lemma 1 $f(i) + f(j) > f(2^n)$ $\forall i, j$ such that $1 \leq i \leq 2^n$ and $1 \leq j \leq 2^n$.

Proof

Since we have been given that $f(1) + f(2) > f(2^n)$ and frequencies are sorted in increasing order so we have $f(i) + f(j) \geq f(1) + f(2) > f(2^n)$ $\forall i, j$ such that $1 \leq i \leq 2^n$ and $1 \leq j \leq 2^n$.

Lemma 2 After the first 2^{n-1} steps of huffman encoding there will be 2^{n-1} nodes such that the maximum character frequency is strictly less than twice the minimum character frequency.

Proof

Let x(i) denote the leaf node corresponding to character with frequency f(i). By the lemma 1, in the i^{th} iteration ($1 \leq i \leq 2^{n-1}$) of huffman encoding, the nodes with least character frequencies will be $x(2i-1)$ and $x(2i)$. We will merge them into a single node, say $y(i)$ which will have character frequency $f(2i-1) + f(2i)$. Then after 2^{n-1} steps all the x(i) nodes will be merged and we will be left with the nodes $y(1), y(2), .., y(2^{n-1})$, where the frequency corresponding to $y(i)$ is $f(2i-1) + f(2i)$.

Observe that the character frequencies of $y(i)$ are in sorted order. Let $l_1 = f(1) + f(2)$ be the character frequency of $y(1)$ and $l_2 = f(3) + f(4)$ be the character frequency of $y(2)$. By lemma 1 we have $l_1 > f(2^n)$ and $l_2 > f(2^n)$ implying that $l_1 + l_2 > 2f(2^n)$ and since $f(2^n - 1) \leq f(2^n)$ we have $l_1 + l_2 > f(2^n) + f(2^n - 1) =$ character frequency of $y(2^{n-1})$. Hence the nodes $y(1), y(2), .., y(2^{n-1})$ satisfy that the maximum character frequency is strictly less than twice the minimum character frequency.

Claim

Huffman encoding of n-bit characters such that the maximum character frequency is strictly less than twice the minimum character frequency is same as that of the ordinary fixed-length encoding i.e the binary tree representing the encoding is complete and has depth n.

Proof

We will proceed by induction.

Base Case ($n = 1$) In this case we have only two characters to encode so we can encode one of them as 0 and other as 1. Hence the base case is true.

Inductive Step ($n - 1 \longrightarrow n$) In order to construct the Huffman encoding, at each stage we greedily combine the two lowest character frequency nodes into a new parent node with character frequency as sum of the character frequency of the two children nodes.

Initially we have 2^n nodes say $x(1), x(2), .., x(2^n)$ with character frequencies in increasing order and satisfying the given constraint. Then by lemma 2 after the first 2^{n-1} steps of Huffman encoding there will be 2^{n-1} nodes say $y(1), y(2), .., y(2^{n-1})$ (where $y(i)$ is parent of $x(2i-1)$ and $x(2i)$) such that the maximum character frequency is strictly less than twice the minimum character frequency i.e they satisfy the same constraint as that by x(i)'s. By inductive hypothesis the encoding tree formed by $y(1), y(2), .., y(2^{n-1})$ is a complete one with depth $n-1$, hence the huffman encoding tree formed by $x(1), x(2), .., x(2^n)$ is a complete binary with depth n.

Hence by PMI our claim holds.

Problem 3

(a) Alice wants to throw a graduation party and is deciding whom to call. She has n people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick a largest subset of n people, subject to two constraints: at the party, each person should have at least five other people whom they know and five other people whom they don't know. Present an efficient algorithm that takes as input the list of n people along with the list of pairs who know each other and outputs the best choice of party invitees. Give the running time in terms of n.

Solution Since "knowing" is a mutual, commutative relation, we can model the problem as a simple undirected graph G. The vertices of G are the set of people which Alice can chose from. Any two vertices(say u and v) are connected by an edge in G, if the corresponding persons know each other. **The problem reduces to finding a maximum size set of vertices such that in the induced subgraph corresponding to that set, the degree of every vertex is atleast 5 and for every vertex in the set, at least 5 other vertices in the subgraph are not adjacent to it**.The following algorithm solves the problem.

Algorithm 2 $GRAD\_PARTY(G)$

---

if $\forall\ v \in G.V$, $v.degree >= 5$ & $v.degree <= |G.V| - 5$ **then**
return $G.V$
else
Choose any v such that $v.degree < 5$ or $v.degree > |G.V| - 5$
$G' \leftarrow G[G.V/\{v\}]$ (*the induced subgraph of* $V/\{v\}$ *in* G, *or* G.E/{*all edges with one of the endpoints as* v})
return $GRAD\_PARTY(G')$
end if

Proof of correctness. The following lemma will help us in the proof.
Lemma A vertex in the graph which has degree less than 5 or degree greater than $n - 5$(where n is the number of vertices in the graph) will never be an element of any optimal set.

Proof If a vertex has degree less than 5, in any subgraph of the original graph the degree will be less than 5 too. Similarily, if a vertex is non-adjacent to less than 5 vertices, in a subgraph of the original graph too, the number of vertices it will not be adjacent to will be less than 5. Hence, it cannot be an element of any optimal set. **Hence, Proved**.

Consider $OPT(G)$ to be the the largest(in terms of vertex set) induced subgraph of G satisfying the constraints. We will prove that $|OPT(G).V| = |OPT(G').V|$, where G' is as chosen by the algorithm above.

Observe that $G'.V \subset G.V$ and $G'.E \subset G.E$, hence the optimal set for G' is also one of the valid(possibly not optimal) sets for G. Hence, $|OPT(G).V| \geq |OPT(G').V|$. $-$ **1**

Consider the optimal solution $OPT(G)$ for G. From the lemma proved above, the vertex chosen by the algorithm will not be a part of $OPT(G)$. Hence, $OPT(G).V \subset G'.V$. Thus, $OPT(G).V$ is one of the valid set(not necessarily optimal) for G'. Hence, $|OPT(G).V| \leq |OPT(G').V|$. $-$ **2**

From **1** and **2**, $|OPT(G).V| = |OPT(G').V|$
Also, Observe that if none of the vertex in the original graph violates the lower and upper bounds on the degree, all the vertices comprise the largest possible set which is optimal. **Hence, Proved**.

Termination Observe that in each tail recursive call, if the degree of all vertices is between 5 and and $|G.V| - 5$ than the recursive call ends in this step, otherwise a vertex is deleted from the graph and a recursive call is made on the obtained subgraph. So in each new recursive call number of vertices in the graph reduces by 1 and since empty graph is a valid solution, the algorithm must terminate.

Efficient Implementation We can implement this algorithm iteratively. The edges of the graph can be stored as an adjacency matrix. Along with it, we can maintain an indexed list of degrees of the vertices in the graph. In each iteration, we look for a vertex with degree less than 5 or greater than "*the current number of vertices in the graph*" $- 5$. If we get such a vertex, then we iterate through the row and column corresponding to this vertex and remove all the edges from the adjacency matrix. Both these operations take $O(n)$ time. Since, there are n vertices, there will be at

most n iterations since in each iteration except the last one, one vertex is removed. Thus, the total time complexity will be $O(n^2)$.

(b) Suppose finally Alice invited n_0 out of her n friends to the party. Her next task is to set a minimum number of dinner tables for her friends under the constraint that each table has a capacity of ten people and the age difference between members of each dining group should be at most ten years. Present a greedy algorithm to solve this problem in $O(n_0)$ time assuming the age of each person is an integer in the range [10, 99].

Solution The following greedy algorithm solve the problem in $O(n_0)$ time.

Algorithm 3 Seating Arrangement

```
Bucket Sort the people in increasing order of there ages, let p_1, p_2, ...p_{n_0} denote the people in this order
i, Tables ← 1, {}
while i ≤ n_0 do
    cnt, table ← 0, {}
    while cnt ≤ 10 and i + cnt ≤ n_0 and age_{p_{i+cnt}} − age_{p_i} ≤ 10 do
        table ← table ∪ {p_{i+cnt}}
        cnt ← cnt + 1
    end while
    Tables ← Tables ∪ table
    i ← i + cnt
end while
return Tables
```

Proof Let $p_1, p_2, ..., p_{n_0}$ denote the people in increasing order of there ages. For the first table our algorithm greedily chooses $p_1, p_2, ..., p_i$ where i is the maximum integer such that $i \leq max(10, n_0)$ and $age(p_i) - age(p_1) \leq 10$. Then subsequently for the remaining people $p_{i+1}, p_{i+2}, ..., p_{n_0}$ same greedy procedure is applied.

To proof the correctness of our algorithm we will first show that there is an optimal solution making the above described greedy choice. Consider an optimal seating arrangement S. Consider the table on which the person with least age i.e p_1 sits. Suppose that $p_1, p_{i_2}, p_{i_2}..., p_{i_k}$ are the people seating on r_{th} table in increasing order of age as per the seating arrangement S.
Observe that $1 < i_2 < ... < i_k \Rightarrow i_j \geq j \ . \forall \ j$ such that $2 \leq j \leq k$.
Now there are 2 cases :
1.) $\{i_2, i_3, i_4, ..., i_k\} = \{2, 3, ..., k-1, k\} \Rightarrow$ In this case, we are done as done as S makes the greedy choice and since S is a valid seating arrangement we have $k \leq max(10, n_0)$ and $age(p_k) - age(p_1) \leq 10$.

2.) $\{i_2, i_3, i_4, ..., i_k\} \neq \{2, 3, ..., k-1, k\} \Rightarrow$ Consider the least u such that $u \in \{2, 3, ..., k-1, k\}$ and $u \notin \{i_2, i_3, i_4, ..., i_k\}$ and least v such that $v \in \{i_2, i_3, i_4, ..., i_k\}$ and $v \notin \{2, 3, ..., k-1, k\}$. Note that we have $v > u$. Suppose p_u sits on the s_{th} table and let $p_{j_1}, p_{j_2}., p_u, .., p_{j_l}$ be the people seating on this table in increasing order of age. Observe that we can swap the table of p_u and p_v without violating any conditions. This is because :
a.) $age(p_u) - age(p_1) \leq age(p_{i_k}) - age(p_1) \leq 10$ as $age(p_u) \leq age(p_k) \leq age(p_{i_k})$.
b.) $0 \leq age(p_v) - age(p_{j_1}) \leq age(p_v) - age(p_1) \leq 10$ and $|age(p_v) - age(p_{j_l})| \leq 10$ as $age(p_{j_1}) \geq age(p_1)$ and $age(p_u) \leq age(p_v)$.
We keep on applying this step until people seating on r_{th} table are $p_1, p_2., p_3, .., p_k$.

Hence by applying the exchange procedure we have shown that there exists an optimal solution making the greedy choice.

Let P be the set of people $p_1, p_2, ...p_{n_0}$ and $OPT(P)$ be minimum number of tables used in seating the people satisfying the given constraints. As per our greedy choice let $p_1, p_2, ..., p_i$ be the people seating on the first table where i is the maximum integer such that $i \leq max(10, n_0)$ and $age(p_i) - age(p_1) \leq 10$. Let $P' = P - \{p_1, p_2, ..., p_i\}$. In order to complete the proof of correctness of our greedy algorithm we will show that $OPT(P) = OPT(P') + 1$.

Since $p_1, p_2, ..., p_i$ can be seated on a single table, hence we have $OPT(P) \leq OPT(P') + 1$. $\quad - \mathbf{1}$

Since there exists an optimal solution in which $p_1, p_2, ..., p_i$ are seated on a single table, we have $OPT(P') \leq OPT(P) - 1$. $\quad - \mathbf{2}$

From **1** and **2**, we have $OPT(P) = OPT(P') + 1$. **Hence proved** .

Time complexity Since we are given that age of people invited in the party falls in the range [10,99] we can use bucket sort to sort the people according to there age. This step will take $O(n_0 + 90) \sim O(n_0)$ time. Observe that in each iteration of inner while loop the value of cnt variable increases by 1 and after the inner while has ended the value of i variable is increased to $i + cnt$, which effectively means that we are doing a linear scan over the sorted array of people. Hence the time complexity of the while loop is $O(n_0)$. Therefor the total time complexity is $O(n_0)$.