COL351 Assignment 1

Aniket Gupta 2019CS10327
Aayush Goyal 2019CS10452

September 2, 2021

1 Minimum Spanning Tree

1.1

1.2

2 Huffman Encoding

2.1

2.2

Since these are 16 bit characters then there are a total of 2^{16} charcaters that are possible. We will denote 2^{16} by n. Let the frequencies of them be $f_1, f_2, ..., f_n$ and they are in increasing order. It is given that $f_n < 2f_1$. Let's denote the symbol with $a_1, a_2, ..., a_n$

Now let's say we consider any numbers from them. Let them be f_i and f_j.

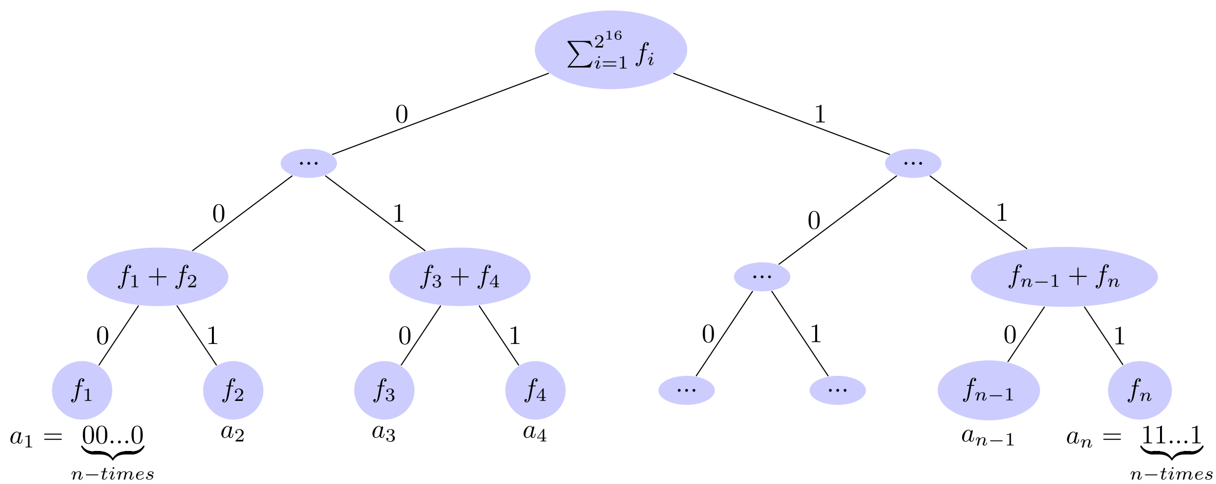
Claim 1: $f_i + f_j > f_n$ (and hence greater than every other frequency).
Proof of claim 1:
$f_i >= f_1$
$f_j >= f_1$
Thus $f_i + f_j >= 2f_1$
Also it is given that $f_n < 2f_1$, thus this directly proves that $f_i + f_j > f_n$.

Now in huffman encoding we choose the 2 vertices with minimum frequency (say f_1 and f_2) and combine them. Then place a node with value $f_1 + f_2$ and then recursively solve the problem further. The symbols that will be choosen in the next iteration will be f_3 and f_4, since $f_4 <= f_5 <= f_n < f_1 + f_2$. And hence we will join f_3 and f_4 from the set and replace with a node of value $f3 + f4$. This will go on and ultimately we will end with these frequencies in the set: $(f_1 + f_2), (f_3 + f_4), (f_5 + f_6), ..., (f_{n-1} + f_n)$, thus all of the initial $a_i's$ will be combined.

Claim 2: let f be a set of numbers of size n, here n is a power of 2. Let the numbers be $f_1, f_2, f_3, f_4, ..., f_{n-1}, f_n$. If we make another set ff from it such that $ff_i = f_{2i-1} + f_{2i}$, then it is of half the size and also follows the property that maximum element is less than twice the minimum element.
Proof of Claim 2: The minimum element of ff set is $f_1 + f_2$ and the maximum element is

$f_{n-1} + f_n$. Now we know that
$f_n < 2f_1$
$f_{n-1} < 2f_1$
and since $f_1 <= f_2$ we can also write that $f_{n-1} < 2f_2$.
Adding both the inequalities we get $f_n + f_{n-1} < 2(f_1 + f_2)$.
Thus for the set ff formed in the above mentioned way, the maximum element is less than twice of the smallest element.

Thus from Claim 2 it is evident that the same pattern will form here and the after combining 2 of them pairwise we will end up nodes of frequency $ff_1 + ff_2, ff_3 + ff_4, ..., ff_{\frac{n}{2}-1} + ff_{\frac{n}{2}}$. Since every successive level is formed after all the nodes from the previous level are exhausted it will take the shape of a perfectly balanced binary tree and every a_i will be at the same level.
Finally the bit encoding of a_i will be the 16 bit repersentation of $(i-1)$



Huffman encoding graph for 16-bit characters, $n = 2^{16}$

3 Graduation Party of Alice

3.1

The following problem can be represented as a graph. With the n people as the nodes of the graph and there is an edge between 2 nodes if the two people know each other. Once the graph is ready, we can make an adjacency list for the same in $O(m)$ time. $n->$ no. of people that are invited to the party and $m->$ no. of pairs who know each other (hence the number of edges in the graph will be m). We can maintain an array which will store the degree of each vertex and this can be done in $O(n+m)$ time using the adjacency list we have created above. Degree here will denote the number of people a person knows.

Claim 1: Any node in the graph which has a degree less than 5 cannot be invited to the party.
Proof of Claim 1: Let v_0 be the vertex with degree less than 5. Now since this node has a degree less than 5, it means that he knows less than 5 peope out of all the people who can be possibly

invited to the party. Their is no such way by which he can know more people and hence the only option that is left with us is to remove him from the list of possible people who can be invited the party.

Now we will keep removing the nodes of the graph which have a degree of less than 5. Notice that as we remove a vertex the degree of it's neighboring vertices will also change and we will have to update their degrees. Removing a node can cause reduction in degree of other nodes. If degree of those vertices fall below 5 then we can't invite them to the party either. We will have to remove them as well. Now this process will continue untill every vertex in the graph has degree of atleast 5.

Let the current number of nodes in the graph be n'. Now consider a node whose degree is more than $n'-6$. Then that person doesn't know less than 5 people and we must do something in this case.

Claim 2: Let V be the current set of nodes and n' be the size of V that is $n' = |V|$. Any vertex whose degree is more than $n' - 6$ cannot be the part of our final optimal solution.
Proof of Claim 2: We will show that their is no final optimal solution in which it doesn't know less than 5 people of all the invited people. Consider the current state of node set V. We know that the final set which will be invited to the pary will be a subset of the current V. Let the node with degree more than $n' - 6$ be v_0. Now if any vertex is removed from V (which is not v_0) then there are 2 possible cases. Either it is a neighbour of v_0, that is it knows v_0, or it is not a neighbour of v_0, it doesn't know v_0. If it doesn't know v_0 then removing it from V only reduces the no. of people v_0 doesn't know. If we remove any node which is the neighbor of v_0, then removing them doesn't change the number of people v_0 doesn't know. Hence the only possibility is we have to remove v_0.

Thus any vertex with degree more than $n' - 6$ cannot be the part of our final solution and it must be removed from V. Removing that person might decrease the degree of other vertices as well. From Claim 1 and Claim 2 we know that all the vertices with degree less than 5 must be removed and all the vertices with degree more than $n' - 6$ should also be removed and this process must continue untill all the nodes left in the final V has a degree atleast 5 and atmost $n' - 6$.

Once such a V is achieved then we can show that all of these people in V can be invited to the party. Consider any node from the set V, let it be v_1. Now v_1 has a degree of more than 4 and thus it has atleast 5 neighbors and thus knows atleast 5 people from will come to the party (because inviting all of the people present in graph). Also it's degree is atmost $n' - 6$ which means there are atleast 5 vertices that are not connceted to v_1. Hence there are atleast 5 people whom v_1 doesn't know.
Hence all of them can be invited to the party.

The problem can be solves in $O(m * log(m))$ time using priority queue, m is the number of edges. The Pseudo code for it is wriiten below. We can initially insert all the nodes ina a Binary min heap. Whenver any node is found to have a degree of less than 5 or more than n'-6, then it is removed and marked removed. The process continues untill the Heap is empty.

Algorthm:

3.2

Consider all of them are standing in a sorted order of their ages. Let us call them a1, a2, a3, a4 ... , an0. Thus we know that age[a1] ¡= age[a2] ¡= age[a3] ... ¡= age[an0]. Now consider the table on which a1 is sitting and let this table be T.

Claim 1: If ak is sitting on T in optimal arrangement and there exists a person (say a1) with age less than ak not sitting on T (say they are aitting on table T1), then there also exists an optimal solution in which ak is sitting in table T1 and ai is sitting on table T.

Proof of claim: Let's say the person with age less than that of ak is ai, ak is a person sitting on the table and ai is not sitting on the table ak. Let's say ai was on the table T1. We can exchange the position of ai and ak in this case. Because: ai can be placed on table of a1 since age[ai] - age[a0] ¡= age[ak] - age[a0]. And we can also place ai in place of ak of ai since the leat age of a member in that is greater than or equal to age[a1] and thus the least upper bound possible is age[a1]+9 and we know that since ak was sitting on T so age[ak] ¡= age[a1] + 9. Thus we can always exchange ai and ak. Keeping ak in the other table can provide more flexibilty on the other table.

Now using the above we can place S = a1, a2, a3 ... , am people on the first table, here m ¡=10 and age[am] - age[a1] ¡10. age[a1] ¡= age[a2] ¡= age[a3] ... ¡= age[am]. If m¡ 10 then then either we don't have enough number of people or age[m+1] - age[a1] ¿=10.

Let J be the set of all the people and S be the set that we have placed. Now define J* = J S.

Claim 2: opt(J) = opt(J*) +1, opt(J) is the minimum number seats required to arrange the set of people J on tables under the required constraints.

Proof of Claim: We will show that both the inequalities: opt(J) ¡= opt(J*) +1 and ...(i) opt(J)-1 ¿= opt(J*) hold. ...(ii)

Proof of (i): After placing S on 1 table we are left with J§people and opt(J*) will place all of them on some table. And hence there exists one arrangement in which the arrangement can be done in opt(J*) +1 no. of tables. Thus opt(J) ¡= opt(J*)+1

Proof of (ii): Let A be the optimal arrangement of J in which all the people in set S are placed on a single table and only they can be seated on that table (say table T). Now none of the people from J§can be seated on that table T since either T is full or their age is more than age[a1]+9. Thus the $A-T$ can be used to place all of the members of J§. Thus J§members can be placed using opt(J)-1 tables. Hence one solution of size opt(J)-1 exists for J*. This hows that opt(J*) ¡= opt(J)-1.

hence from both (i) and (ii) we have, opt(J)= opt(J*)+1.

Now for calculating the final we can maintain a freq array. freq[i] denotes the number of people with age i+10. Now we can start placing them on tables in increasing order as we have discussed above, by recursively dividing the problem into a smaller sub-problem.