6.046
PROBLEM SET 9

JOHN WANG

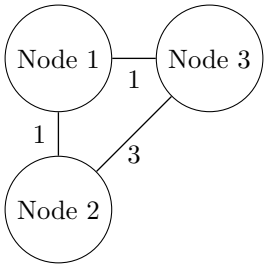
Collaborators: Sashko Stubailo, Brodrick Childs

1. Problem 9-1

1.1. Problem A.

Problem 1.1. *Provide an example of a room graph where finding the MPST for S would not give the piping of minimum overall length.*

Solution Consider the following graph, where $S = \{\text{Node 2}, \text{Node 3}\}$. It is clear that the only edge that connects nodes 2 and 3 is the edge of weight 3. Thus, this must be the MPST for S since it is the only edge that can possibly be in a minimum spanning tree. However, we see if if we had created an edge first from Node 2 to Node 1, then an edge from Node 1 to Node 3, the total cost of the piping system would have been 2, which is lower than the cost of the MPST.



□

1.2. Problem B.

Problem 1.2. *Assume any pair of roomsx, y can be connected using a pipe of length $d(x, y)$. Devise an efficient polynomial-time approximation algorithm for the plumbing problem with an approximation ratio $\alpha \leq 2$.*

Solution Create a graph of the problem where each node represents a room in S. Let the graph be completed connected so that each vertex has an edge to every other vertex. The weights of the vertices are simply the distances between the room. Thus, if vertex x corresponds to room x and vertex y corresponds to room y, then the weight of edge (x, y) is simply $d(x, y)$. Now the graph contains vertices to all rooms in S, but no other rooms. We will now run Prims to obtain a minimum spanning tree on the graph. We will then output the result of the minimum spanning tree as the rooms where pipes should go through. This will be a polynomial time 2-approximation of our problem.

For runtime, we note that the initialization of the graph takes $O(V + E) = O(n^2)$ time if there are n rooms. Moreover, we know that Prims runs in $O(E + V \log V) = O(n^2)$. This shows that the algorithm will run in polynomial time. All that is left is the proof of correctness.

For correctness, we shall prove the following lemmas:

Lemma 1.1. *Let S be the mission critical set of elements in G, and let T be an optimal connection of rooms. Then there is a tree T' which spans only the vertices in S such that $c(T') \leq 2c(T)$.*

Proof. Let us consider the graph of T and run depth first search on it. If we list the vertices that are visited (including repeats and backtracks), then we obtain a path $x_0, x_1, x_2, \ldots, x_k$, where $x_k = x_0$. Since the DFS visits each node in T exactly twice, we know that $\sum_{i=0}^{k} d(x_i, x_{i+1}) = 2c(T)$. Now, let $y_0, y_1, \ldots, y_l$ be the set of vertices from $\{x_i\}$ where $x_i \in S$ such that there are no repeated vertices.

Let us imagine that y_i corresponds to x_i and y_{i+1} corresponds to x_{j+1}. Then we see that $d(y_i, y_{i+1}) < d(x_i)+d(x_{i+1})+\ldots+d(x_{j+1})$ by the triangle inequality. This shows that $\sum_{i=0}^{l} d(y_i, y_{i+1}) < \sum_{i=0}^{k} d(x_i, x_{i+1}) = 2c(T)$. Therefore, we see that the total cost of the path $\{y_i\}$ is less than $2c(T)$. However, we see that $y_i \in S$ for all i. Moreover, since T originally spanned all the elements of S, we see that $\{y_i\}$ must also contain all of elements of S. Thus, we can take $\{y_i\}$ as a tree T' which spans only the vertices in S and has cost $c(T') \leq 2c(T)$. □

Using this lemma, we observe that a minimum spanning tree on S will always find a tree with total weight less than $2c(T)$ where T is the optimal connection of rooms. This is because there exists a tree T' which spans only the vertices in S which has this property. Thus, a minimum spanning tree on S will have a cost that is at most the cost of T'. Finding the minimum spanning tree using the algorithm given above will therefore provide a 2-approximation to the plumbing problem. □

1.3. **Problem C.**

Problem 1.3. *You are provided with an updated blueprint of possible direct pipe connections and effective pipe lengths for each of them (which may no longer satisfy the triangle inequality). Devise a polynomial-time approximation algorithm with approximation ratio $\alpha \leq 2$, using what you learned in part (b).*

Solution We will convert this problem into a problem which satisfies the triangle inequality, so that we can use the algorithm from problem b. The algorithm will proceed as follows: create a graph G where all rooms are vertices and each edge represents a possible connection between rooms (drop all room connections where pipes to go through). The weight of the edge will be given by the space-time distance (which may not satisfy the triangle inequality). Run Floyd-Warshall and find the shortest path between all vertices $x, y \in G$ and $d'(x, y)$ as the distance of the shortest path. We now perform the algorthim from part b on the new graph G with a distance metric of d'. After the algorithm completes, we take its output and construct a graph T' where we replace each edge (x, y) from the output with the shortest path from x to y given in Floyd-Warshall. Return this graph G' as the output of the algorithm.

First, we analyze the running time of the algorithm. Initialization of the graph will take time $O(V+E) = O(n^2)$. We know that Floyd-Warshall takes time $O(V^3) = O(n^3)$. Next, we must run the algorithm from problem (b), which will take time $O(n^2)$. Finally, converting the output of the algorithm from problem (b) into the final output will take time $O(E^2) = O(n^4)$ since each edge can have at most E new edges to add on for the shortest path. This implies that the entire algorithm runs in polynomial time.

Now, to show correctness, we shall note that the metric d' satisfies the triangle inequality. This is because shortest paths follow the traingle inequality. Consider three points x, y, z. We must have $d'(x, y) + d'(y, z) \leq d'(x, z)$. If not, then we could choose $d'(x, z) = d'(x, y) + d'(y, z)$ as the shortest path and find a smaller cost for $d'(x, z)$, which would be a contradiction. This implies that we can apply the algorithm from problem (b) to the problem and obtain a valid 2-approximation for the graph G with metric d'. Let us say T is the output of the algorthim from problem (b), then we know that $c(T) \leq 2opt(G, d')$ by the reasoning from problem (b).

Next, we know that $d'(x, y) \leq d(x, y)$ because the shortest path from x to y will always be less than its absolute distance (or else we could just choose $d'(x, y) = d(x, y)$ to have the same distance). This implies that $opt(G, d') \leq opt(G, d)$. Since we replace T with T' where all of the edges between x, y are changed to shortest paths, we find that $c(T') = c(T) \leq 2opt(G, d') \leq 2opt(G, d)$. This shows that $c(T') \leq 2opt(G, d)$.

Moreover, we know that T' satisfies the requirements of the problem because we have shown that problem (b)'s algorithm is correct. Since we only change the edges from the output of the algorithm into their shortest paths, we still have a valid covering of all the vertices in S. This shows that we have found a 2-approximation to our problem. □

2. Problem 9-2

2.1. **Problem A.**

Problem 2.1. *You are working with a Huffman encoding scheme over n symbols. How long would a codeword be in the worst case? Provide an example set of frequencies $f_1, f_2, \ldots, f_n$ that would give rise to this longest codeword.*

Solution In the worst case, the Huffman encoding tree only has one node on each level. This occurs when every new node is incorporated on the right side of the graph. If one builds up recursively, we see that $f_1 + f_2 < f_3$, and that $f_3 + f_4 < f_5$, etc. This builds a tree which is stacked on the right side. There n levels in this tree because each codeword takes up its own level, so a codeword could be of length $O(n)$.

For an example of this, consider the frequencies $f_k = 2^{k-(n+1)}$. For instance, if $n = 5$, then the frequencies are $f_1 = 0.031, f_2 = 0.0625, f_3 = 0.125, f_4 = 0.25, f_5 = 0.5$. This sequence makes sure that $f_{k-2} + f_{k-1} < f_k$ so that the tree will always be stacked on the right side. Building the tree from above with $n = 5$ creates a right-sided tree with only one node per level. □

2.2. **Problem B.**

Problem 2.2. *Prove that if some character occurs with frequency more than* $2/5$*, then there is guaranteed to be a codeword of length 1.*

Solution Let n be the number of codewords in the tree. Clearly if $n = 3$, then there will always be a codeword of length 1 no matter what. Now, let us examine the case of $n = 4$. Let the frequencies of the codewords be $f_1 < f_2 < f_3 < f_4$. Now, suppose by contradiction that there is no codeword of length 1. Then the tree will be balanced, with a subtree of two leaves to the left and right of the root. This means that $f_1 + f_2 > f_4$. Moreover, since we know there is a character with frequency of more than $2/5$, we know that $f_1 + f_2 > f_4 > 2/5$. Next, we know that in order for f_3 to merge with f_4, we must have $f_1 < f_2 < f_3$. However, since $f_1 + f_2 > 2/5$, we must also have $f_3 > 1/5$. This follows because the smallest f_2 can possibly be in order for $f_1 < f_2$ and $f_1 + f_2 > 2/5$ to be satisfied is $f_2 = 1/5$. This implies that $f_3 > f_2 = 1/5$.

However, we see that the sum $f_1 + f_2 + f_3 + f_4 = (f_1 + f_2) + f_3 + f_4 < 2/5 + 1/5 + 2/5 = 1$. However, this is a contradiction because the frequencies must all add up precisely to 1. This shows that the graph must have a codeword of length 1. Now, if $n > 4$, we simply note that the tree will result in $f_x < f_y < f_3 < f_4$ where $f_4 > 2/5$, in which case the argument will hold again. This is because $f_3 > 1/5$ and $f_4 > 2/5$, which means that all the nodes before f_3 and f_4 can have a maximum cumulative frequency of $2/5$. This shows that the argument holds for all $n > 4$. □

2.3. **Problem C.**

Problem 2.3. *Prove that if all codewords occur with frequency less than* $1/3$*, then there is guaranteed to be no codeword of length 1.*

Solution First, note that if $n \leq 3$ is the number of codewords, then it is not possible to have all codewords with frequency less than $1/3$. Thus, we are confined to looking at $n \geq 4$. Now, suppose there is a codeword of length 1. Let us say its frequency is f_n. Now, let us say we have performed the Huffman algorithm for a number of merges and are left with a tree with f_n to the left of the root and a subtree to the right of the root with f_x and f_y stemming from the subtree's root.It is clear that $f_n > f_x$ and $f_n > f_y$ since it is a distance of 1 away from the root. We know that $f_n < 1/3$ by the hypothesis, which implies that $f_x + f_y + f_n < 1/3 + 1/3 + 1/3 = 1$. This is a contradiction because the frequencies of the tree at any point in the Huffman algorithm must sum to 1. This argument holds for arbitrary n, so we have completed the proof. □