CS 170 HW 6

Due on 2019-03-04, at 10:00 pm

1 Study Group

List the names and SIDs of the members in your study group.

2 Finding MSTs by Deleting Edges

Consider the following algorithm to find the minimum spanning tree of an undirected, weighted graph $G(V, E)$. For simplicity, you may assume that no two edges in G have the same weight.

procedure FINDMST($G(V, E)$)
  $E' \leftarrow E$
  for Each edge e in E in decreasing weight order **do**
    if $G(V, E' - e)$ is connected **then**
      $E' \leftarrow E' - e$
  return E'

Show that this algorithm outputs a minimum spanning tree of G.

3 Huffman Coding

In this question we will consider how much Huffman coding can compress a file F of m characters taken from an alphabet of $n = 2^k$ characters x_0, x_1, ... , x_{n-1} (each character appears at least once).

(a) Let $S(F)$ represent the number of bits it takes to store F without using Huffman coding (i.e., using the same number of bits for each character). Represent $S(F)$ in terms of m and n.

(b) Let $H(F)$ represent the number of bits used in the optimal Huffman coding of F. We define the *efficiency* $E(F)$ of a Huffman coding on F as $E(F) := S(F)/H(F)$. For each m and n describe a file F for which $E(F)$ is as small as possible.

(c) For each m and n describe a file F for which $E(F)$ is as large as possible. How does the largest possible efficiency increase as a function of n? Give you answer in big-O notation.

4 Graph Game

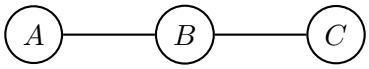
Given an undirected, unweighted graph G, with each node v having a value $\ell(v) \geq 0$, consider the following game.

1. All nodes are initially *unmarked* and your score is 0.

2. Choose an unmarked node u. Let $M(u)$ be the *marked* neighbours of u. Add $\sum_{v \in M(u)} \ell(v)$ to your score. Then mark u.

3. Repeat the last step for as many turns as you like, or until all the nodes are marked.

For instance, suppose we had the graph:



with $\ell(A) = 3$, $\ell(B) = 2$, $\ell(C) = 3$. Then, an optimal strategy is to mark A then C then B giving you a score of $0 + 0 + 6$. We can check that no other order will give us a better score.

(a) Is it ever better to leave a node unmarked? Briefly justify your answer.

(b) Give a greedy algorithm to find the order which gives the best score. Describe your algorithm, prove its optimality, and analyse its running time.

(c) Now suppose that $\ell(v)$ can be negative. Give an example where your algorithm fails.

(d) Your friend suggests the following modified algorithm: delete all v with $\ell(v) < 0$, then run your greedy algorithm on the resulting graph. Give an example where this algorithm fails.

5 Sum of Products

This question guides you through writing a proof of correctness for a greedy algorithm. You have n computing jobs to perform, with job i requiring t_i units of CPU time to complete. You also have access to n machines that you can assign these jobs to. Since the machines are in high demand, you can only assign one job to any machine. The jth machine costs c_j dollars for each unit of CPU time it spends running a job, so assigning job i to machine j will cost you $t_i \cdot c_j$ dollars (each job takes the same amount of CPU time to complete, regardless of which machine is used). Your goal is to find an assignment of jobs to machines that minimizes the total cost.

Assume the jobs and machines are sorted and have distinct runtimes/costs, i.e. $t_1 > t_2 > \ldots > t_n$ and $c_1 > c_2 > \ldots > c_n$.

(a) What assignment of jobs to machines minimizes the total cost? (Hint: What machine should we assign the longest job to? What machine should we assign the second longest job to? It might help to solve a small example by hand first.)

(b) Given an assignment of jobs to machines, consider the following modification: If there is a pair of jobs i, j such that job i is assigned to machine i', job j is assigned to machine j', and $t_i > t_j$ and $c_{i'} > c_{j'}$, instead assign job i to machine j' and job j to machine i'. Show that this modification decreases the total cost of an assignment.

(c) Use part b to show that the assignment you chose in part a has the minimum total cost (Hint: Show that for any assignment other than the one you chose in part a, you can apply the modification in part b. Conclude that the assignment you chose in part a is the optimal assignment.)

6 Preventing Conflict

A group of n guests shows up to a house for a party, but some pairs of guests are enemies. There are two rooms in the house, and the host wants to distribute guests among the rooms, breaking up as many pairs of enemies as possible. The guests are all waiting outside the house and are impatient to get in, so the host needs to assign them to the two rooms quickly, even if this means that it's not the best possible solution. Come up with a linear-time algorithm that breaks up at least half as many pairs of enemies as the best possible solution. Provide a proof of correctness and a runtime analysis as well.

You can treat the input as an undirected graph $G = (V, E)$ where each vertex in V represents a guest and (u, v) is in E if u and v are enemies.