CS 170 HW 5

Due on 2017-02-26, at 11:59 pm

1 (★★) Minimum Spanning Trees (short answer)

(a) Given an undirected graph $G = (V, E)$ and a set $E' \subset E$ briefly describe how to update Kruskal's algorithm to find the minimum spanning tree that includes all edges from E'.

(b) Assume you are given a graph $G = (V, E)$ with positive and negative edge weights and an algorithm that can return a minimum spanning tree when given a graph with only positive edges. Describe a way to transform G into a new graph G' containing only positive edge weights so that the minimum spanning tree of G can be easily found from the minimum spanning tree of G'.

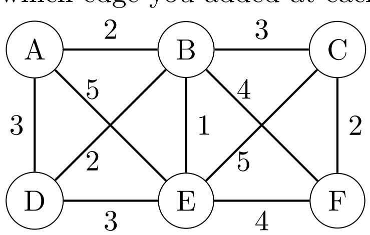
(c) Describe an algorithm to find a maximum spanning tree of a given graph.

2 (★) Prim's Algorithm

A popular alternative to Kruskal's algorithm is Prim's algorithm, in which the intermediate set of edges X always forms a subtree, and S is chosen to be the set of this tree's vertices. We can think of Prim's algorithm as greedily processing one vertex at a time, adding it to S. The pseudocode below gives the basic outline of Prim's algorithm. See the book for a detailed example of a run of the algorithm.

S = $\{v\}$
X = $\{\}$
While S $\neq$ V:
    Choose $t \in V \backslash S$, $s \in S$ such that $weight(s, t)$ is minimized
    X = X $\cup$ $\{(s, t)\}$
    S = S $\cup$ $\{t\}$
Return X

(a) Run Prim's algorithm on the following graph, stating which node you processed and which edge you added at each step .



(b) Prim's algorithm is very similar to Dijkstra's in that a vertex is processed at each step which minimizes some cost function. These algorithms also produce similar outputs: the union of all shortest paths produced by a run of Dijkstra's algorithm forms a tree. However, the trees they produce aren't optimizing for the same thing. To see this, give an example of a graph for which different trees are produced by running Prim's algorithm and Dijkstra's algorithm. In other words, give a graph where there is a shortest path from a start vertex A using edges that don't appear in any minimum spanning tree.

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_2, ... , x_{n-1}.

(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?

4 (★★) Horn Formulas

Describe a linear-time algorithm to find a satisfying assignment for a Horn formula, if it exists.

5 (★★) Money Changing Redux

During discussion section, we saw a simple greedy algorithm to try to find change that adds up to a given number. We saw that the greedy algorithm didn't find the optimal solution in all cases. In this problem, we will use our newly-found powers of computer science to fix this.

Recall that in the money-changing problem, we were given a fixed set of positive integers called *denominations* $x_1, x_2, \ldots, x_n$ (think of them as the integers 1, 5, 10, and 25). The problem you want to solve for these denominations is the following: Given an integer A, express it as

$$A = \sum_{i=1}^{n} a_i x_i$$

for some nonnegative integers $a_1, \ldots, a_n \geq 0$.

(a) You might remember that we can represent any integer k in *unary* form by repeating k consecutive 1s (e.g., 3 is represented by 111 in unary). Assume you are given a positive integer A and a set of denominations $x_1, x_2, \ldots, x_n$ in unary form. Give a fast algorithm to solve the money-changing problem.

(b) If you are given A and $x_1, x_2, \ldots, x_n$ in binary, does your algorithm still run in polynomial time? Why or why not?