CS 170 HW 6

Due **2020-03-02, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

2 2-SAT

Please provide solutions to parts (d), (e) and (f) of Question 3.28 from `http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf`.

3 Perfect Matching on Trees

A *perfect matching* in an undirected graph $G = (V, E)$ is a set of edges $E' \subseteq E$ such that for every vertex $v \in V$, there is exactly one edge in E' which is incident to v.
Give an algorithm which finds a perfect matching *in a tree*, or reports that no such matching exists. Describe your algorithm, prove that it is correct and analyse its running time.

4 Encoding Emergencies

The basic intuition behind Huffman's algorithm (i.e. that frequent words have short encodings and infrequent words have long encodings) is at work in the English language. The words "I", "me", "you", "he", and "she" are all short, while the word "velociraptor" is quite long. However, words like "fire!","danger!", or "high-voltage!" are not short because they are frequent. Instead, time is precious in situations when these words are used.
To make things theoretical, suppose we have a file containing words numbered $1, \dots, m$ occurring with frequencies $f(1), \dots, f(m)$ and suppose that for each word i, the cost per bit of encoding of the word is $c(i)$, so that if we find a prefix code where word i has encoding length $l(i)$, the total cost of our encoding will be $\Sigma_i f(i) \cdot c(i) \cdot l(i)$.
Show how to modify Huffman's algorithm in order to compute the prefix code of minimum total cost.

5 Minimum Spanning k-Forest

Given a graph $G(V, E)$ with nonnegative weights, a spanning k-forest is a cycle-free collection of edges $F \subseteq E$ such that the graph with the same vertices as G but only the edges in F has k connected components. For example, consider the graph $G(V, E)$ with vertices $V = \{A, B, C, D, E\}$ and all possible edges. One spanning 2-forest of this graph is $F = \{(A, C), (B, D), (D, E)\}$, because the graph with vertices V and edges F has components $\{A, C\}, \{B, D, E\}$.

The minimum spanning k-forest is defined as the spanning k-forest with the minimum total edge weight. (Note that when $k = 1$, this is equivalent to the minimum spanning tree). In this problem, you will design an algorithm to find the minimum spanning k-forest. For simplicity, you may assume that all edges in G have distinct weights.

(a) Define a j-partition of a graph G to be a partition of the vertices V into j (non-empty) sets. That is, a j-partition is a list of j sets of vertices $\Pi = \{S_1, S_2 \ldots S_j\}$ such that every S_i includes at least one vertex, and every vertex in G appears in exactly one S_i. For example, if the vertices of the graph are $\{A, B, C, D, E\}$, one 3-partition is to split the vertices into the sets $\Pi = \{\{A, B\}, \{C\}, \{D, E\}\}$.

Define an edge (u, v) to be crossing a j-partition $\Pi = \{S_1, S_2 \ldots S_j\}$ if the set in Π containing u and the set in Π containing v are different sets. For example, for the 3-partition $\Pi = \{\{A, B\}, \{C\}, \{D, E\}\}$, an edge from A to C would cross Π.

Show that for any j-partition Π of a graph G, if $j > k$ then the lightest edge crossing Π must be in the minimum spanning k-forest of G.

(b) Give an efficient algorithm for finding the minimum spanning k-forest.

Please give a 3-part solution.

6 A Divide and Conquer Algorithm for MST

Is the following algorithm correct? If so, prove it. Otherwise, give a counterexample and explain why it doesn't work.

```
procedure FindMST($G$: graph on $n$ vertices)
    If $n = 1$ return the empty set
    $T_1 \leftarrow$ FindMST($G_1$: subgraph of $G$ induced on vertices $\{1, \ldots, n/2\}$)
    $T_2 \leftarrow$ FindMST($G_2$: subgraph of $G$ induced on vertices $\{n/2 + 1, \ldots, n\}$)
    $e \leftarrow$ cheapest edge across the cut $\{1, \ldots, \frac{n}{2}\}$ and $\{\frac{n}{2} + 1, \ldots, n\}$.
    return $T_1 \cup T_2 \cup \{e\}$.
```

7 Picking a Favorite MST

Consider an undirected, weighted graph for which multiple MSTs are possible (we know this means the edge weights cannot be unique). You have a favorite MST, F. Are you guaranteed that F is a possible output of Kruskal's algorithm on this graph? How about Prim's? In other words, is it always possible to "force" the MST algorithms to output F without changing the weights of the given graph? Justify your answer.