# CS170 Discussion Section 6: *3/1*

## 1 Minimum Spanning Trees

For each of the following statements, either prove or supply a counterexample. Always assume $G = (V, E)$ is undirected and connected. Do not assume the edge weights are distinct unless specifically stated.

(a) Let $e$ be any edge of minimum weight in $G$. Then $e$ must be part of some MST.

(b) If the lightest edge in a graph is unique, then it must be part of every MST.

(c) If $e$ is part of some MST of $G$, then it must be a lightest edge across some cut of $G$.

(d) If $G$ has a cycle with a unique lightest edge $e$, then $e$ must be part of every MST.

(e) The shortest path tree computed by Djikstra's algorithm is necessarily part of some MST.

(f) Prim's algorithm works correctly when there are negative edges.

(g) For any $r > 0$, define an $r$-path to be a path whose edges all have weight less than $r$. If $G$ contains an $r$-path from $s$ to $t$, then every MST of $G$ must also contain an $r$-path from $s$ to $t$.

## 2 Approximating Vertex Cover

Given an undirected graph $G = (V, E)$, nodes $S \subseteq V$ form a *vertex cover* of $G$ if every edge has at least one of its endpoints in $S$. In other words, $\forall (i, j) \in E$, we have $i \in S$ or $j \in S$.

It turns out that finding the smallest set of vertices that covers all the edges—the *minimum vertex cover*—is a very hard problem. Fortunately, it is easy find a vertex cover not too much larger than the true minimum. Consider the following greedy algorithm:

$S \leftarrow \emptyset$
**for** $(i, j) \in E$ **do**
  **if** $(i, j)$ is not covered **then** add both $i$ and $j$ to $S$
**return** $S$

Show that this algorithm is guaranteed to find a vertex cover of size at most $2 \cdot \textsc{opt}$, where $\textsc{opt}$ is the size of the minimum vertex cover.

## 3 Service scheduling

A server has $n$ customers waiting to be served. Customer $i$ requires $t_i$ minutes to be served. If, for example, the customers were served in the order $t_1, t_2, t_3, \ldots$, then the $i$th customer would wait for $t_1 + t_2 + \cdots + t_i$ minutes.

We want to minimize the total waiting time

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i)$$

Given the list of $t_i$, give an efficient algorithm for computing the optimal order in which to process the customers.

## 4 Huffman Encoding

1. Under a Huffman encoding of $n$ symbols with frequencies $f_1, f_2, \ldots, f_n$, what is the longest a codeword could possibly be? Give an example set of frequencies that would produce this case, and argue that it is the longest possible.

2. Prove that if all characters occur with frequency less than $1/3$, there is guaranteed to be no codeword of length 1.

3. Prove that if some character occurs with frequency more than $\frac{2}{5}$, there is guaranteed to be a codeword of length 1.