CSE 421 Midterm Review

1 Matchings

1. Consider the stable matching algorithm. Show how a woman w can be matched with all of the m's during the course of the Gale-Shapley algorithm by giving the preference lists and describing an execution of the algorithm where some w is matched with each m in turn.

2. Let $G = (V, E)$ be an undirected, bipartite graph. A matching M is said to be *maximal* if every edge of E shares at least one endpoint with an edge of M. Let M^* be a maximum cardinality matching for G.

   a) Give an example of a graph with a maximal matching M_g where $|M_g| = \frac{1}{2}|M^*|$.

   b) For a bipartite graph and maximal matching M_g, prove that $|M_g| \geq \frac{1}{2}|M^*|$.

3. The classical Stable Matching Problem assumes that all men and women have a fully ordered list of preferences. In this problem we will consider a version of the problem in which men and women can be *indifferent* between certain options. As before we have a set M of n men and a set W of n women. Assume each man and each woman ranks the members of the opposite gender, but now we allow ties in the ranking. For example (with $n = 4$), a woman could say that m_1 is ranked in first place; second place is a tie between m_2 and m_3 (she has no preference between them); and m_4 is in last place. We will say that w *prefers* m to m' if m is ranked higher than m' on her preference list (they are not tied). With indifferences in the rankings, there could be two natural notions for stability. And for each, we can ask about the existence of stable matchings, as follows.

   - A *strong instability* in a perfect matching S consists of a man m and a woman w, such that each of m and w prefers the other to their partner in S. Does there always exist a perfect matching with no strong instability? Either give an example of a set of men and women with preference lists for which every perfect matching has a strong instability; or give an algorithm that is guaranteed to find a perfect matching with no strong instability.
   - A *weak instability* in a perfect matching S consists of a man m and a woman w, such that their partners in S are w' and m', respectively, and one of the following holds:
     - m prefers w to w', and w either prefers m to m' or is indifferent between these two choices; or
     - w prefers m to m', and m either prefers w to w' or is indifferent between these two choices.

     In other words, the pairing between m and w is either preferred by both, or preferred by one while the other is indifferent. Does there always exist a perfect matching with no weak instability? Either give an example of a set of men and women with preference lists for which every perfect matching has a weak instability; or give an algorithm that is guaranteed to find a perfect matching with no weak instability.

2 Complexity

1. Arrange the following six functions in order of non-decreasing growth rate.
   - $f_1(n) = n^{2.1}$
   - $f_2(n) = \sqrt{7n}$
   - $f_3(n) = n^{2+\sin n}$
   - $f_4(n) = 10^n$
   - $f_5(n) = 100^n$
   - $f_6(n) = n^2 \log n$
   - $f_7(n) = \sum_{i=1}^{n} (i^2 + 7i)$
   - $f_8(n) = 2^{5 \log n}$
   - $f_9(n) = n^{\log n}$
   - $f_{10}(n) = n^{\log \log n}$
2. Suppose $f(n) = O(g(n))$. Prove or give a counterexample to the following:
   (a) $\log f(n) = O(\log g(n))$.
   (b) $\sum_{i=1}^{n} i^k \log i = \Theta(n^{k+1} \log n)$.

3 Graphs

1. Give an example of a directed graph with exactly two topological orderings and one node of in-degree 0.
2. Give an $O(n)$ algorithm to determine whether a given undirected graph on n nodes is a tree.
3. Let $G = (V, E)$ be an undirected graph. Let v be a vertex in G. Give an $O(n + m)$ time algorithm that finds the shortest cycle in G which contains the vertex v.
4. There is a new alien language which uses the latin alphabet. However, the order among letters is unknown to you. You receive a list of words from the dictionary, where words are sorted lexicographically by the rules of this new language. Design an algorithm to derive the order of letters in this language.

   For example, given the following words in dictionary,

   ```
   ["wrt", "wrf",  "er",  "ett",  "rftt"],
   ```

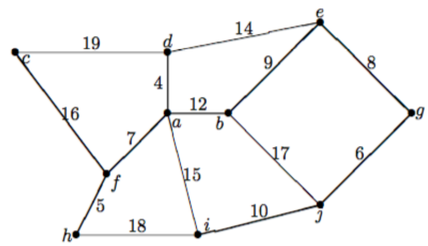
   your algorithm should return `"wertf"`.

   If there may be multiple valid order of letters, simply return any one of them.

4 Greedy Algorithms

1. Suppose that e is the maximum weight edge in an undirected, weighted graph G, where the weights on all edges are distinct. Give a necessary and sufficient condition for e to be in the minimum spanning tree of G.

2. Consider the following undirected graph G.



   (a) Highlight the edges of G that are in a minimum spanning tree.

   (b) Use the Edge Inclusion Lemma to argue that the edge (c, f) is in every Minimum Spanning Tree of G.

3. The *binpacking problem* is: Given a collection of items $I = \{i_1, \ldots, i_n\}$ where each item i_j has an s_j, an integer K, and a collection of bins $B = b_1, \ldots, b_m$ assign the items to bins such that the sum of the sizes of items assigned to each bin b_i is at most K. The goal is to minimize the number of bins that receive items, i.e., to pack the items into as few bins as possible.

   A greedy algorithm for the problem considers the items in order, and places each item in the first bin that has enough remaining space to hold the item.

   For this problem, assume that the bin size is $K = 3$, and the items have sizes 1, 2, or 3.

   a) Give an example that shows that the greedy algorithm does not necessary find the optimal solution (minimizes the number of bins).

   b) Describe an algorithm which minimizes the number of bins used.

4. Consider the following scheduling problem:

   Input: A set of n jobs, such that job i has weight w_i and processing time p_i.

   Output: A schedule (an order in which to process the jobs) that minimizes $\sum_j w_j C_j$, where C_j is the completion time of job j (i.e., $\sum_{i \in S_j} p_i$, where S_j is the set of jobs that are scheduled before job j, including job j itself.)

   Consider the greedy algorithm that schedules the jobs in nonincreasing order of w_j/p_j. Give a brief exchange argument to prove that this algorithm is optimal.

5 Recurrences

1. True or False: If the running time of an algorithm satisfies the recurrence $T(n) \leq T(n/10) + T(9n/10) + c(n)$, then $T(n) = O(n^2)$.

2. Give solutions to the following recurrences. Justify your answers.

   a)
   $$T(n) = \begin{cases} T(n/8) + 1 & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$

   b)
   $$T(n) = \begin{cases} T(9n/11) + n & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$

   c)
   $$T(n) = \begin{cases} 16T(n/4) + n^2 & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$