CSCI 570 - Summer 2021 - HW 1

1 Graded Problems

1. True.

2. False.

3. True.

4. A stable matching need not exist. Consider the following list of preferences. Let a, b, c all have d last on their list. Say a prefers b over c, b prefers c over a, and c prefers a over b. In every matching, one of a, b, c should be paired with d and the other two with each other. Now, d's roommate and the one for whom d's roommate is the first choice prefer to be with each other. Thus every matching is unstable no stable matching exists in this case.

5. In ascending order of growth, the list is $g_1(n), g_3(n), g_4(n), g_5(n), g_2(n), g_7(n), g_6(n)$.

6. Without loss of generality assume that G is connected. Otherwise, we can compute the connected components in $\mathcal{O}(m+n)$ time and deploy the below algorithm on each component.

   Starting from an arbitrary vertex s, run BFS and obtain a BFS tree (call it T). If $G = T$, then G is a tree and has no cycles. Otherwise, G has a cycle and hence there exists an edge $e = (u, v)$ such that e is in G but not in T. Find the least common ancestor of u and v in the tree. Call the least common ancestor w. There exist a unique path (call P_1) in T from u to w (and likewise a unique path P_2 in T from v to w). These paths can be constructed in $\mathcal{O}(m)$ time by starting from u (respectively from v) and going up the tree until w is reached. Output the cycle e concatenated with P_2 concatenated with $\bar{P}_1$. Here $\bar{P}_1$ denotes P_1 in the reverse order.

2 Practice Problems

1. We will use a variation of Gale and Shapley (GS) algorithm, then show that the solution returned by this algorithm is a stable matching.

   In the following algorithm (see next page), we use hospitals in the place of men; and students in the place of women, with respect to the earlier version of the GS algorithm given in Chapter 1.

```
while there exists a hospital h that has available positions do
  Offer position to the next highest ranked student s in the preference list of
  $h$
  if $s$ has not already matched to another hospital $h'$ then
    accept the offer of $h$
  else
    Let $s$ be matched with $h'$.
    if $s$ prefers $h'$ to $h$ then
      then $s$ does not accept the offer of $h$
    else
      $s$ accepts the offer from $h$
    end if
  end if
end while
```

   This algorithm terminates in $O(mn)$ steps because each hospital offers a position to a student at most once, and in each iteration some hospital offers a position to some student.

   The algorithm terminates by producing a matching M for any given preference list. Suppose there are $p > 0$ positions available at hospital h. The algorithm terminates with all of the positions filled. Any hospital that did not fill all of its positions must have offered them to every student. But then every student must be committed to some hospital. But if h still has available positions, $p > n$, where n is the number of students. This contradicts the assumption that the number of students is greater than the number of available positions.

   The assignment is stable. Suppose, towards a contradiction, that the M produced by our adapted GS algorithm contains one or more instabilities. If the instability was of the first type (a preferred student was not admitted), then h must have considered s before s', which is a contradiction because h prefers s' to s. The instability was not of the first type. If the instability was of the second type (there is a mutually beneficial swap between hospitals and students), then h must not have admitted s when it considered it before s, which implies that s' prefers h' to h, a contradiction. The instability was not of the second type. If the contradiction was of neither type it must not have existed, thus the matching was stable.

Thus at least one stable matching always exists (and it is produced by the adapted GS algorithm).

2. We know from the text that polynomials grow slower than exponentials. Thus, we will consider f_1, f_2, f_3, f_6 as a group, and then put f_4 and f_5 after them. For polynomials f_i and f_j, we know that f_i and f_j can be ordered by comparing the highest exponent on any term in f_i to the highest exponent on any term in f_j. Thus, we can put f_2 before f_3 before f_1. Now, since f_6 grows faster than n^2, and we know that logarithms grow slower than polynomials, so f_6 grows slower than n^c for any $c > 2$. Thus we can insert f_6 between f_3 and f_1. Finally come f_4 and f_5, we know that exponential can be ordered by their bases, so put f_4 before f_5. Therefore, in ascending order of growth, the list is f_2, f_3, f_6, f_1, f_4, f_5.

3. Assume that functions $f(n)$ and $g(n)$ take nonnegative values.

   (a) False. Consider for example $f(n) = 2, \forall n$ and $g(n) = 1, \forall n$.

   Clearly, $f(n) = \mathcal{O}(g(n))$. Observe that $\log_2(f(n)) = 1, \forall n$ and $\log_2(g(n)) = 0, \forall n$. Hence $\log_2(f(n)) \neq \mathcal{O}(\log_2(g(n)))$.

   Note: If we further add the constraint that $\exists N$ such that $g(n) \geq 2, \forall n > N$, then the statement becomes true.

   (b) False. Consider for example $f(n) = 2n$ and $g(n) = n$. Clearly 4^n is not $\mathcal{O}(2^n)$.

   (c) True. Since $f(n) = \mathcal{O}(g(n))$, there exists positive constants c and n_0 such that $f(n) \leq cg(n), \forall n \geq n_0$. This implies $f(n)^2 \leq c^2 g(n)^2, \forall n \geq n_0$, which in turn implies that $f(n)^2 = \mathcal{O}(g(n)^2)$.

4. Assume that G contains an edge $e = (x, y)$ that does not belong to T. Since T is a DFS tree and (x, y) is an edge of G that is not an edge of T, one of x or y is ancestor of the other. On the other hand, since T is a BFS tree if x and y belong to layer L_i and L_j respectively, then i and j differ by at most 1. Notice that since one of x or y is an ancestor of the other, we have that $i \neq j$ and hence i and j differ by exactly 1. However, combining that one of x or y is ancestor of the other and that i and j differ by 1 implies that the edge (x, y) is in the tree T. It contradicts the assumption that $e = (x, y)$ that does not belong to T. Thus G cannot contain any edges that do not belong to T.