ASSIGNMENT 1: Solutions CMSC 451 (Section 0101, Fall 2020)

1. *Asymptotic growth.* [10 points]

   List the following 5 functions in ascending order of asymptotic growth rate. (In other words, if $f(n)$ precedes $h(n)$ in your list, then it should be that $f = O(h)$.) The base of log is always 2.

   $$g_1(n) = 2^{\sqrt{\log n}},\ \ g_2(n) = n^{5/2},\ \ g_3(n) = 5^{\log n},\ \ g_4(n) = \log^7 n,\ \ g_5(n) = n^6 + \log^2 n$$

   Solution: We have $g_1 = n^{1/\sqrt{\log n}}$ (which grows more slowly than any fixed power of n, but faster than any fixed power of $\log n$) and $g_3 = n^{\log 5} \approx n^{2.322}$, so the order is g_4, g_1 g_3, g_2, g_5.

2. *Gale-Shapley execution.* [25 points]

   Run the Gale-Shapley algorithm on the instance below, where each row indicates the ranked order of the indicated candidate/employer (highest ranked first).

Candidate preferences		Employer preferences	
candidate 1:	3, 7, 4, 2, 1, 5, 6	employer 1:	1, 6, 4, 3, 2, 5, 7
candidate 2:	4, 2, 7, 5, 1, 3, 6	employer 2:	4, 3, 7, 1, 6, 5, 2
candidate 3:	7, 2, 3, 4, 1, 6, 5	employer 3:	1, 6, 4, 2, 3, 5, 7
candidate 4:	4, 6, 2, 3, 1, 7, 5	employer 4:	7, 3, 6, 2, 4, 1, 5
candidate 5:	6, 7, 5, 2, 4, 3, 1	employer 5:	1, 6, 7, 2, 3, 5, 4
candidate 6:	5, 4, 1, 7, 3, 2, 6	employer 6:	4, 7, 6, 1, 5, 3, 2
candidate 7:	7, 6, 3, 4, 5, 1, 2	employer 7:	1, 5, 4, 3, 6, 7, 2

   List which employer each candidate is paired with. (*Note:* You should either run the algorithm by hand, or implement it in a programming language of your choice. You may *not* use someone else's implementation.)

   Solution: With the order (employer, candidate), the output of the Gale-Shapley algorithm is $\{(1,2),(2,3),(3,1),(4,7),(5,6),(6,4),(7,5)\}$.

3. *Stable matching for a class project.* [15 points]

   Suppose we want to assign partners in a class of $2n$ students. Each student provides a ranking of the other $2n-1$ students; a *stable matching* is an assignment of partners so that no two students who are not partners would each prefer to work with each other rather than with their assigned partner. Show that a stable matching need not exist by giving an input instance (set of rankings) with $n = 2$ and proving that no stable matching is possible for that instance.

   Solution: Suppose the four students Alice, Bob, Cathy, and David have the following preferences:

   Alice: Bob, Cathy, David
   Bob: Cathy, David, Alice
   Cathy: David, Bob, Alice
   David: Bob, Cathy, Alice

   Suppose $X \in \{\text{Bob}, \text{Cathy}, \text{David}\}$ is paired with Alice. Since Alice is ranked lowest by all her potential partners, X would prefer to be with either of the alternatives. Since each of $\{\text{Bob}, \text{Cathy}, \text{David}\}$ is ranked first by someone in that set, there is some $Y \in \{\text{Bob}, \text{Cathy}, \text{David}\}$ that ranks X first. Students X, Y are not paired (since X is paired with Alice) but they would prefer to be with each other than with their assigned partners. Hence no matching can be stable.

4. *Stable matching with incomplete rankings.*

Consider a variant of the stable matching problem where a candidate need not rank every employer but can instead mark some employers as *excluded*. (If a candidate c excludes an employer e, then c prefers not to be matched at all rather than to be matched to e.) We call a matching *acceptable* if no candidate is matched with an employer that they exclude. Assume all candidates rank at least one employer.

(a) [10 points] Show that a perfect, acceptable matching need not exist.

Solution: Consider a case where there are two students and two companies, and both students rank only the first company. In an acceptable matching, only one of the two students can be matched to the first company and the other student cannot be matched to the second company because that student did not rank the second company. Therefore, any acceptable matching will be imperfect because the student who is not matched to the first company cannot be matched to any company.

(b) [20 points] Propose an algorithm that outputs a nonempty, stable, acceptable matching. (Stability is defined as usual; a candidate prefers any employer they ranked to an employer they excluded.) Give pseudocode for your algorithm along with a proof of correctness and an analysis of its running time.

Solution: Consider an algorithm that behaves identically to the original Gale-Shapley (GS) algorithm, except that a candidate will only accept an engagement from a non-excluded employer. This algorithm clearly produces an acceptable matching since a candidate can never become engaged with an excluded employer.

We claim that once a candidate receives an offer from a non-excluded employer, they remain engaged until the end of the algorithm, and their job can only rise in their preference list over time. This follows by the same argument as for the GS algorithm, except that the candidate only becomes engaged if they receive an offer from a *non-excluded* employer.

Next, we claim that the algorithm terminates after at most n^2 iterations of the while loop. This follows identically as for the GS algorithm.

We also claim that the matching output by the algorithm is stable. To see this, we suppose (for a contradiction) that there is an employer-candidate pair (e, c) who would prefer to be with each other than to follow the assigned matching M (in particular, this means c must not exclude e). We consider two cases:

- If $(e, c') \in M$ for some c' that is lower in e's preferences than c, then e must have made an offer to c before c', since the algorithm has companies make offers in decreasing order.
- If e is unmatched in M, then e must have made an offer to c at some point during the algorithm, because the algorithm does not exit the while loop until all unengaged companies have made offers to every candidate.

In either case, this means that when e made an offer to c, either c already had an offer they preferred, so they rejected e's offer; or c accepted e's offer but later switched to a preferred employer. Either way, c ends up with an employer ranked at least as high as e, but instead M either assigns a lower-ranked employer or none at all, which is a contradiction. Therefore the matching must be stable.

Observe that the empty matching cannot be stable under the given assumption that every candidate ranks at least one employer, because if c ranks e then (e, c) would both rather be together than be unmatched. Thus the matching is nonempty.

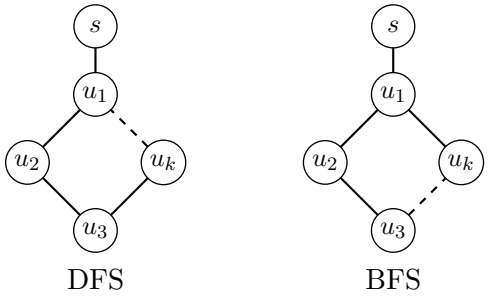
Finally, note that the above claim about termination shows that the algorithm runs in the same time as GS, namely $O(n^2)$.

5. *When are BFS and DFS the same?* [15 points]

   Fix a connected graph G and a specific vertex s in that graph. Suppose we obtain the same tree T when computing a breath-first search tree rooted at s and a depth-first search tree rooted at s. Prove that $G = T$. (*Hint:* you may use any results proved in Chapter 3 of the book, even if we did not cover them in class.)

   Solution: If G is a tree then clearly both BFS and DFS must output that tree (since this is the only possible spanning tree). Thus it remains to show that if BFS and DFS produce the same output, G must be a tree. Suppose for a contradiction that G contains a cycle C. Without loss of generality, let the cycle be $C = (u_1, u_2, \ldots, u_k, u_1)$ where u_1 is the first vertex explored by the DFS starting at s. By the definition of DFS, the algorithm will traverse the path $(u_1, u_2, \ldots, u_k)$ and then encounter $\{u_k, u_1\}$ as a non-tree edge.

   On the other hand, consider the BFS tree rooted at s. Once the vertex u_1 is discovered by the BFS, both its incident edges $\{u_1, u_2\}$ and $\{u_1, u_k\}$ will be added to the BFS tree. But this is a contradiction since the trees must have been identical. Hence, G must be a tree.



6. *Collaboration.* [5 points]

   Write "I understand the course collaboration policy and have followed it when working on this assignment." List the other students with whom you discussed the problems, or else indicate that you did not discuss any problems with your classmates.