CSCI 570 - Summer 2023 - HW 1 Solutions

Reading Assignment: Kleinberg and Tardos, Chapter 1.

1. Solve Kleinberg and Tardos, Chapter 1, Exercise 1.

   Solution:

   False. Suppose we have two men m, m_0 and two women w, w_0. Let m rank w first; w rank m_0 first; m_0 rank w_0 first; and w_0 rank m first. We see that such a pair as described by the claim does not exist.

   Rubric (5 pt):

   2pt: Correctly identifies the statement is false.

   3pt: Provides a correct counterexample as explanation

2. Solve Kleinberg and Tardos, Chapter 1, Exercise 2.

   Solution:

   True. Suppose S is a stable matching that contains the pairs (m, w_0) and (m_0, w), for some $m_0 \neq m$, $w_0 \neq w$. Clearly, the pairing (m, w) is preferred by both m and w over their current pairings, contradicting the stability of S.

   Rubric (5pt):

   2pt: Correctly identifies the statement is true.

   3pt: Provides a correct explanation.

3. State True/False: An instance of the stable marriage problem has a unique stable matching if and only if the version of the Gale-Shapely algorithm where the male proposes and the version where the female proposes both yield the exact same matching.

   Solution:

   True.
   Claim: *"If there is a unique stable matching then the version of the Gale-Shapely algorithm where the male proposes and the version where the female proposes both yield the exact same matching."*

   The proof of the above claim is clear by virtue of the correctness of Gale-Shapely algorithm. That is, the version of the Gale-Shapely algorithm where the male proposes and the version where the female proposes are

both correct and hence will both yield a stable matching. However since there is a unique stable matching, both versions of the algorithms should yield the same matching.

Converse of Claim: *"If the version of the Gale-Shapely algorithm where the male proposes and the version where the female proposes both yield the exact same matching then there is a unique stable matching."*

The proof of the converse is perhaps more interesting. For the definition of best valid partner, worst valid partner etc., see page 10-11 in the textbook.

Let S denote the stable matching obtained from the version where men propose and let S_0 be the stable matching obtained from the version where women propose.

From (page 11, statement 1.8 in the text), in S, every woman is paired with her worst valid partner. Applying (page 10, statement 1.7) by symmetry to the version of Gale-Shapely where women propose, it follows that in S_0, every woman is paired with her best valid partner. Since S and S_0 are the same matching, it follows that for every woman, the best valid partner and the worst valid partner are the same. This implies that every woman has a unique valid partner which implies that there is a unique stable matching.

Rubric (7 pt):

2pt: Correctly identifies the statement is true.

2pt: Provides a correct explanation for forward claim.

3pt: Provides a correct explanation for backward claim.

4. A stable roommate problem with 4 students a, b, c, d is defined as follows. Each student ranks the other three in strict order of preference. A matching is defined as the separation of the students into two disjoint pairs. A matching is stable if no two separated students prefer each other to their current roommates. Does a stable matching always exist? If yes, give a proof. Otherwise give an example roommate preference where no stable matching exists.

Solution:

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.

Rubric (7pt):

2pt: Correctly identifies that stable matching wont always exist.

5pt: Provides a correct explanation and example.

5. Solve Kleinberg and Tardos, Chapter 1, Exercise 3.

   Solution:

   We will give an example (with $n = 2$) of a set of TV shows/associated ratings for which no stable pair of schedules exists. Let a_1, a_2 be set the shows of A and let b_1; b_2 be the set of shows of B. Let the ratings of a_1; a_2; b_1; b_2 be 1, 3, 2 and 4 respectively. In every schedule that A and B can choose, either a_1 shares a slot with b_1 or a_1 shares a slot with b_2. If a_1 shares a slot with b_1, then A has an incentive to make a_1 share a slot with b_2 thereby increasing its number of winning slots from 0 to 1. If a_1 shares a slot with b_2, then B has an incentive to make b_2 share a slot with a_2 thereby increasing its number of winning slots from 1 to 2. Thus every schedule that A and B can choose is unstable.

6. Solve Kleinberg and Tardos, Chapter 1, Exercise 4.

   Solution:

   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, 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 secondtype (there is a mutually beneficial swap between hospitals and students), then h must not have admitted s' when it consideredit before s, which implies that s' prefers h' to h, a contradiction. The instability was not of the secondtype. 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).

Rubric (15 pts):

8pts: Algorithm

- 1pt: Loop condition (line 1)
- 2 pts: hospitals offer next highest ranked student (line 2)
- 2pts: case that s is free (lines 3-4)
- 3pts: cases if s is at another h'

7pts: Proof

- 1 pt: Algorithm terminates in finite steps (optional to mention in $O(mn)$ steps)
- 2pts: All positions get filled
- 2pts: Explain why no instability of first type
- 2pts: Explain why no instability of secondtype

7. What is the worst-caseruntime performance of the procedure below?

```
c = 0
i = n
while i > 1 do
   for j = 1 to i do
      c = c + 1
   end for
   i = floor(i/ 2)
end while
return c
```

Solution:

There are *i* operations in the for loop and the while loop terminates when *i* becomes1. The total time is

$$n + \lfloor n/2 \rfloor + \lfloor n/4 \rfloor + \cdots \leq (1 + 1/2 + 1/4 + \cdots) \cdot n \leq 2n = O(n).$$

Rubric (5 pts):

2pts: if bound correctly found as $O(n)$

3pts: Provides a correct explanation of the runtime

2pts total if bound given is $O(n \log n)$ (i.e., not a tight upper bound)

8. Arrange these functions under the *O* notation using only = (equivalent) or $\subset$ (strict subset of):

(a) $2^{\log n}$

(b) 2^{3n}

(c) $n^{n \log n}$

(d) $\log n$

(e) $n \log(n^2)$

(f) n^{n^2}

(g) $\log(\log(n^n))$

E.g. for the function n, $n + 1$, n^2, the answer should be

$$O(n + 1) = O(n) \subset O(n^2).$$

Solution:

First separate functions into logarithmic, polynomial, and exponential. Note that

$$2^{\log n} = n, \quad n^{n \log n} = 2^{n(\log n)^2}, \quad n^{n^2} = 2^{n^2 \log n},$$

we have:

(a) Logarithmic: $\log n$, $\log(\log(n^n))$

(b) Polynomial: $2^{\log n}$, $n \log(n^2)$

(c) Exponential: 2^{3n}, $n^{n \log n}$, n^{n^2}

- Since

$$\log n \leq 1 \cdot \log(n \log n) = \log(\log(n^n)),$$

so $\log n = O(\log(\log(n^n)))$. On the other hand,

$$\log(\log(n^n)) = \log(n \log n) \leq \log(n^2) = 2 \cdot \log n,$$

so $\log(\log(n^n)) = O(\log n)$. Thus

$$O(\log n) = O(\log(\log(n^n))).$$

- Since every logarithmic grows slower than every polynomial,

$$O(\log(\log(n^n))) \subset O(2^{\log n}).$$

- $2^{\log\ n} = O(n) \subset O(n \log n) = O(2 \cdot n \log n) = O(n \log n^2)$. Thus

$$O(2^{\log n}) \subset O(n \log(n^2)).$$

- Since every exponential grows faster than every polynomial,

$$O(n \log(n^2)) \subset O(2^{3n}).$$

- Since

$$O(3n) \subset O(n(\log n)^2) \subset O(n^2 \log n),$$

so

$$O(2^{3n}) \subset O(2^{n(\log n)^2}) = O(n^{n \log n}) \subset O(2^{n^2 \log n}) = O(n^{n^2}).$$

Therefore,

$$O(\log n) = O(\log(\log(n^n))) \subset O(2^{\log n}) \subset O(n \log(n^2)) \subset O(2^{3n}) \subset O(n^{n \log n}) \subset O(n^{n^2})$$

Rubric (10 pts):

−2 pts for each "inversion" in the order
For example, $O(2^{3n}) \subset O(n^{n^2}) \subset O(n^{n \log n})$ has one inversion and $O(n^{n \log n}) \subset O(n^{n^2}) \subset O(2^{3n})$ has two inversions

−2pts for any = mistaken to be $\subset$ or vice versa

9. Given functions f_1, f_2, g_1, g_2 such that $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.

(a) $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$

(b) $f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$

(c) $f_1(n)^2 = O(g_1(n)^2)$

(d) $\log_2 f_1(n) = O(\log_2 g_1(n))$

Solution:

By definition, there exist $c_1, c_2 > 0$ such that

$$f_1(n) \leq c_1 \cdot g_1(n) \text{ and } f_2(n) \leq c_2 \cdot g_2(n)$$

for *n* sufficiently large.

(a) True.

$$f_1(n) \cdot f_2(n) \leq c_1 \cdot g_1(n) \cdot c_2 \cdot g_2(n) = (c_1 c_2) \cdot (g_1(n) \cdot g_2(n)).$$

(b) True.

$$\begin{aligned} f_1(n) + f_2(n) &\leq c_1 \cdot g_1(n) + c_2 \cdot g_2(n) \\ &\leq (c_1 + c_2)(g_1(n) + g_2(n)) \\ &\leq 2 \cdot (c_1 + c_2) \max(g_1(n), g_2(n)). \end{aligned}$$

(c) True.

$$f_1(n)^2 \leq (c_1 \cdot g_1(n))^2 = c_1^2 \cdot g_1(n)^2.$$

(d) False. Consider $f_1(n) = 2$ and $g_1(n) = 1$. Then

$$\log_2 f_1(n) = 1 \neq O(\log_2 g_1(n)) = O(0).$$

Rubric (4 pts for each subproblem):

1pts: Correct T/F claim

3pts: Provides a correct explanation or counterexample

10. Given an undirected graph *G* with *n* nodes and *m* edges, design an $O(m + n)$ algorithm to detect whether *G* contains a cycle. Your algorithm should output a cycle if *G* contains one.

Solution:

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

Starting from an arbitrary vertex *s*, run BFS to obtain a BFS tree *T*, which takes $O(m + n)$ time. If $G = T$, then *G* is a tree and has no cycles. Otherwise, *G* has a cycle and there exists an edge $e = (u, v) \in G \setminus T$. Let *w* be the least common ancestor of *u* and *v*. There exist a unique path T_1 in *T* from *u* to *w* and a unique path T_2 in *T* from *w* to *v*. Both T_1 and T_2 can be found in $O(m)$ time. Output the cycle *e* by concatenating P_1 and P_2.

Rubric (15 pts):

No penalty for not mentioning disconnected case.

7pts: for detecting whether *G* contains a cycle

5pts: for finding (the edges in) a cycle if *G* contains one

3pts: describing that the runtime is $O(m + n)$ in each step (and thus total)

Ungraded problems

11. N men and N women were participating in a stable matching process in a small town named Walnut Grove. A stable matching was found after the matching process finished and everyone got engaged. However, a man named Almazo Wilder, who is engaged with a woman named Nelly Oleson, suddenly changes his mind by preferring another woman named Laura Ingles, who was originally ranked right below Nelly in his preference list, therefore Laura and Nelly swapped their positions in Almanzos preference list. Your job now is to find a new matching for all of these people and to take into account the new preference of Almanzo, but you don't want to run the whole process from the beginning again, and want to take advantage of the results you currently have from the previous matching. Describe your algorithm for this problem.
Assume that no woman gets offended if she got refused and then gets proposed by the same person again.

Solution:

First, Almanzo leaves Nelly and proposes to Laura:

- If Laura rejects Almanzo and decides to stay with her current fianc (her current fiance is ranked higher than Almanzo in her preference list), then Almanzo just goes back and proposes to Nelly again and gets engaged with her. Algorithm stops. This is where you take advantage of the previous matching.
- Else, Laura accepts Almanzo proposal, she will get engaged with him and leave her current fiance (say Adam). And here comes the main part of the matching puzzle.
  (a) Note that Adam may or may not simply propose to Nelly and be engaged with her. The reason is Nelly may be far below Laura in his preference list and there are more women (worse than Laura and better than Nelly), to whom he prefers to propose first. Furthermore, the fact that Nelly becomes single can create more instabilities for those men, who once proposed to Nelly and were rejected by her because she preferred Almanzo. Thus, Nelly is ranked higher than the current fiances of those once unlucky men and they are ready to propose to Nelly again for a second chance.
  (b) Thus, one way is to put Adam and all those rejected-by-Nelly men in the pool of single men, and put Nelly and the fiancees of the rejected-by-Nelly men in the pool of free women. Similar to Nellys case, moving any woman from the engaged state to the single state may create more instabilities; thus one needs execute step (b) recursively to build the pools of free men and women. In the worst case, all men and all women will end up in the

single pools; in the best case (when Laura accepted Almanzos proposal), only Adam and Nelly are in the 2 single pools, one for men and another for women.

(c) Execute G-S algorithm until there is no free man. For each man, if there is a woman who once rejected him and is now free, he will try to propose to her again. Otherwise, he will propose to those women that he has never proposed to, moving top down in his preference list. In the latter case, more men (subsequently more women) may go to the single pools.

12. Solve Kleinberg and Tardos, Chapter 2, Exercise 6.

Solution:

(a) The outer loop runs for exactly n iterations, the inner loop runs for at most n iterations, and the number of operations needed for adding up array entries $A[i]$ through $A[j]$ is $i + j - 1 = O(n)$. Therefore, the running time is in $n^2 \cdot O(n) = O(n^3)$.

(b) Consider those iterations that require at least $n/2$ operations to add up array entries $A[i]$ through $A[j]$. When $i \leq n/4$ and $j \geq 3n/4$, the number of operations needed is at least $n/2$. So there are at least $(n/4)^2$ pairs of (i, j) such that adding up $A[i]$ through $A[j]$ requires at least $n/2$ operation. Therefore, the running time is at least $\Omega((n/4)^2 \cdot n/2) = \Omega(n^3/32) = \Omega(n^3)$.

(c) Consider the following algorithm:

```
for i = 1,2, .. . ,n - 1 do
   B[i, i + 1] ← A[i] + A[i + 1]
end for
for j = 2,3, .. . ,n - 1 do
   for i = 1,2, .. . ,n - j do
      B[i, i + j ] ← B[i, i + j - 1] + A[i + j ]
   end for
end for
```

It first computes $B[i, i + 1]$ for all i by summing $A[i]$ with $A[j]$. This for loop requires $O(n)$ operations. For each j, it computes all $B[i, i + j]$ by summing $B[i, i + j - 1]$ with $A[i + j]$. This works since the value $B[i, i + j - 1]$ were already computed in the previous iteration. The double for loop requires $O(n) \cdot O(n) = O(n^2)$ time. Therefore, the algorithm runs in $O(n^2)$.

13. Solve Kleinberg and Tardos, Chapter 3, Exercise 6.

Solution:

Proof by Contradiction: assume there is an edge $e = (x, y)$ in G that does not belong to T. Since T is a DFS tree, one of x or y is the ancestor of the other. On the other hand, since T is a BFS tree, x and y differs by at

most 1 layer. Now since one of x and y is the ancestor of the other, x and y should differ by exactly 1 layer. Therefore, the edge $e = (x, y)$ should be in the BFS tree T. This contradicts the assumption. Therefore, G cannot contain any edges that do not belong to T.