CS:3330 Homework 1, Fall 2015

Due in class on Tue, Sept 15

1. Prove that there is an input with n men and n women to the Gale-Shapley algorithm that forces it to execute $\Omega(n^2)$ iterations of the **while**-loop. Organize your answer as follows:

   (a) First, describe your input (i.e., preferences of each of the n men and each of the n women) for arbitrary n.

   (b) Next, illustrate your answer for (a) by showing your input for $n = 3$.

   (c) Finally, show that the Gale-Shapley algorithm will need to execute $\Omega(n^2)$ **while**-loop iterations on the input described in (a). Here, you'll have to specify the order in which "free" men are chosen by your algorithm to make proposals.

2. Consider an extension to the stable marriage problem in which each woman expresses preferences over some *subset* of the men; she finds the remaining men unacceptable and she'd rather be alone than be married to any of these unacceptable men. Similarly, each man has preferences over a subset of women and finds the remaining women unacceptable.

   In this setting, a marriage $\mathcal{M}$ is a set of (m, w)-pairs such that each woman appears in *at most one* pair in $\mathcal{M}$; similarly, each man appears in at most one pair in $\mathcal{M}$. Furthermore, for every $(m, w) \in \mathcal{M}$, m finds w acceptable and w finds m acceptable.

   A marriage $\mathcal{M}$ is *stable* if there is no $(m, w) \notin \mathcal{M}$ such that

   (i) m and w are acceptable to each other and

   (ii) m is unmarried or m is married and prefers w to his current partner and

   (iii) w is unmarried or w is married and prefers m to her current partner.

   (a) Describe an *efficient* algorithm (using pseudocode) that computes a stable marriage in this setting. Use the style and level of detail that we used for the Gale-Shapley algorithm in class. Your algorithm needs to be correct, but you do not have to provide a proof of correctness.

   (b) Suppose that each man and each woman has preferences for at most t individuals, where $0 \leq t \leq n$. What is the size of the input to the problem? What is the running time of your algorithm as a function of the input size?

3. Problem 8 from Chapter 1 in the textbook.

4. Take the following list of functions (from nonnegative integers to nonnegative integers) and arrange them in ascending order of growth. Thus, if a function g immediately follows f in the list, then $f = O(g)$.

   (a) 2^n

   (b) $n^2 \log_2 n$

   (c) $n^{4/3}$

   (d) $2^{\sqrt{\log n}}$

   (e) $n \cdot (\log_2 n)^3$

   (f) $100n^2$

   (g) $n^3 / \log_3 n$

5. Analyze the running times of the following code fragments. For each code fragment, express your answer as a function of n, using the Θ notation. Please show your work in order to receive partial credit.

(a)
```
$j \leftarrow n$
for $i \leftarrow 1$ to $n$ do
    $k \leftarrow 1$
    while $k \le j$ do
        print("hello")
        $k \leftarrow k + 1$
    $j \leftarrow j/2$
```

(b)
```
$j \leftarrow n$
for $i \leftarrow 1$ to $n$ do
    $j \leftarrow 1$
    while $j \le i$ do
        print("hello")
        $j \leftarrow 2 \times j$
```

(c)
```
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow i$ to $n$ do
        for $k \leftarrow 1$ to $n/3$ do
            print("hello")
```

Here are some mathematical identities that will help you solve this problem:

- *Geometric series:* If $|r| < 1$, then

$$a + a \cdot r + a \cdot r^2 + a \cdot r^3 + \cdots = \frac{a}{1-r}.$$

- *Arithmetic series:*

$$a + (a + d) + (a + 2d) + \cdots + (a + (n-1)d) = \frac{n(2a + (n-1)d)}{2}.$$

  If we set $a = 1$ and $d = 1$, we get the particularly useful special case

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}.$$

- *Stirling's Approximation:*

$$\ln(n!) = n \ln n - n + O(\ln n).$$

6. You are given a list of n integers. Design an algorithm that returns the number of pairs of duplicates in this list. For example, if the list is $(13, 3, 8, 3, 13, 7, 13, 9, 13)$, then the four 13's in list yield 6 pairs of duplicates and the two 3's in the list yield one pair. The other elements in the list are all unique. Thus your algorithm should return 7 as the total number of pairs of duplicates. Your algorithm should run *asymptotically faster* than the simple $\Theta(n^2)$ time algorithm that examines all pairs of elements in the list.

7. Problem 6 from Chapter 2 in the textbook.