CS:3330 Exam 1, Spring 2017

Tuesday, Feb 21 2017, 6:30 pm to 8:30 pm

1. This problem is on understanding the growth rate of functions that represent running times of algorithms and the use of asymptotic notation.

   (a) 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 $f = O(g)$. Some of these functions are described in words and some as summations. For every function described in words or as a summation, write it in standard form first before placing it in the sorted list of functions. Show your work in order to receive partial credit.

      (i) $2^{2.5 \log_2 n}$

      (ii) The running time of the MergeSort algorithm.

      (iii) $100n^2 + 1000000$

      (iv) $(\log_2 n)^2 \cdot \sum_{i=1}^{n} \Theta(1/2^i)$

      (v) $n^{1.5}/(\log_2 n)^4$

      (vi) $2^{7\sqrt{\log_2 n}}$

   (b) For each statement below, write down if it is `True` or `False`. Provide a 1-2 sentence justification for your answer.

      (i) $100n^3 + 10n^2 + 15 = \Theta(n^2)$.

      (ii) I prefer an algorithm running in $\Theta(\sqrt{2^n})$ time relative to an algorithm running in $\Theta(3^{\log_2 n})$ because the first algorithm is more efficient.

      (iii) There is an algorithm that solves MVC (i.e., produces an optimal solution for every input) in $O((m+n)\log n)$ time.

      (iv) $n^{\log_2 n} = \Theta(2^{(\log_2 n)^3})$.

2. Write down the worst case running time of each of the following code fragments. Use the Θ notation to express your answers and show your work to receive partial credit.

   (a) Express your answer as a function of n

```
function printHello(n)
    for $i \leftarrow 1$ to $n$ do
        $B \leftarrow 1$
        while $(B < n)$ do
            print("hello")
            $B \leftarrow 2 * B$
```

   (b) Express the running time as a function of n and ϵ.

```
    for $i \leftarrow 1$ to $n$ do
        $j \leftarrow n$
        while $j > 0$ do
            print("hello")
            $j \leftarrow j - 2 \cdot \epsilon$
```

   (c) The arguments to this function are a length-n list S of positive integers and a positive integer target weight W. Express the running time as a function of n and W.

```
function subsetSum(list $S[1,\dots,n]$, $W$)
    $M \leftarrow Array(n+1, W+1)$   // an empty 2-dimensional (n+1) by (W+1) array
    for $w \leftarrow 0$ to $W$ do
        $M[0,w] \leftarrow 0$
    for $i \leftarrow 1$ to $n$ do
        for $w \leftarrow 0$ to $W$ do
            if $w < S[i]$ then
                $M[i,w] \leftarrow M[i-1,w]$
            else
                $M[i,w] \leftarrow \max\{M[i-1,w], 1 + M[i-1, w - S[i]]\}$
    return $M[n,W]$
```

3. The Stable Matching Problem, as discussed in class, 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 strictly higher than m' on her preference list (i.e., 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) A *strong instability* in a perfect matching S consists of a man m and a woman w (who are not matched in S), such that each of m and w prefers the other to their current 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.

   (b) 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.

4. For each $k = 2, 3, \dots$, we define a graph G_k as follows. Let $n = k!$. Start with a subset U containing n vertices labeled $u_1, u_2, \dots, u_n$. We will add to the graph other vertices and edges so that the degree of each vertex in U ends up being k. Now add to the graph, k subsets of vertices $V_1, V_2, \dots, V_k$, where $|V_j| = n/j$ for each j, $1 \leq j \leq n$. Now we will describe the edges incident on each vertex in V_j. Consider the n/j vertices in V_j in some order. Connect the first vertex in V_j to vertices $u_1, u_2, \dots, u_j$, then the second vertex in V_j to $u_{j+1}, u_{j+2}, \dots, u_{2j}$, then the third vertex in V_j to $u_{2j+1}, u_{2j+2}, \dots, u_{3j}$, and so on. Thus every vertex in V_j has degree j and every vertex in U is connected to exactly one vertex in V_j, for any j.

   (a) Carefully draw and label the graph G_3.

(b) Now consider the GreedyDegreeBased algorithm for the *Minimum Vertex Cover (MVC)*. (Recall this algorithm from lectures and HW4.) Execute this algorithm on G_3 with the following tie-breaking rule: whenever there is a tie between two or more vertices of maximum degree, your algorithm should choose a vertex from $\cup_{j=1}^k V_j$, rather than a vertex from U. It does not matter how ties are broken between pairs of vertices in $\cup_{j=1}^k V_j$ or between pairs of vertices in U. What is the vertex cover produced by the algorithm in G_3? What is an optimal vertex cover for G_3?

(c) Your friend claims that the GreedyDegreeBased algorithm is a 3-approximation algorithm. What is the smallest member of the family of graphs G_k defined above that you could use as a counterexample to disprove your friend's claim? Justify your answer in 1-2 sentences.
Note: Here it may help you to know that $\sum_{i=1}^{10} 1/i$ is approximately 2.93 and $\sum_{i=1}^{11} 1/i$ is approximately 3.02.

5. We are given a length-n binary list L. In other words, $L[j] \in \{0, 1\}$ for all $j = 1, 2, \ldots, n$. Our problem is to count the number of 0's in L. We could of course solve this problem in $\Theta(n)$ time by simply scanning L, but we want to solve it faster and so we turn to randomization. The randomized algorithm we use is the following.

```
function CountZeroes($L$)
    $n \leftarrow$ length($L$)
    $count \leftarrow 0$

    for $j \leftarrow 1$ to 100 do
        Comment: pick a random index $i$ between 1 and n
        $i \leftarrow random(1, n)$
        if $L[i] = 0$ then
            $count \leftarrow count + 1$

    return $n \cdot count/100$
```

(a) What is the running time of this algorithm?

(b) Suppose that $n = 10^6$ and exactly a quarter of the elements in L are 0. What is the probability that the CountZeroes function returns 0?

(c) A random variable X has *binomial distribution* with parameters m (a positive integer) and p (a real number between 0 and 1) if

$$\Pr[X = k] = \binom{m}{k} \cdot p^k \cdot (1-p)^{m-k}. \quad (1)$$

The way to think about X being binomially distributed is that we perform m independent random trials and each trial can either succeed or fail. The "success" probability of a trial is p and so the failure probability of a trial is $1 - p$. Then the random variable X represents the number of successful trials, out of m total random trials. To understand the formula in (1) note that the probability of k successful trials is p^k, the probability of $(m - k)$ unsuccessful trials is $(1 - p)^{m-k}$, and there are $\binom{m}{k}$ ways of distributing the k successful trials among the m random trials.

As in (b) suppose that $n = 10^6$ and exactly a quarter of the elements in L are 0. Using the above expression for a binomial distribution, write down the expression for the probability that CountZeroes returns the correct answer, namely $10^6/4$.