CS:3330 Exam 1, Fall 2015

Thursday, Sept 24 2015, 6:30 pm to 8:30 pm

---

1. Here are two problems 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) $2^{5 \log_2 n}$

   (ii) The running time of the `binarySearch` algorithm.

   (iii) $100n^2 + 1000000$

   (iv) $\sqrt{2^n}$

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

   (vi) $n^3/(\log_2 n)^4$

   (vii) $2^{\sqrt{\log_2 n}}$

   (viii) 100

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

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

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

(iii) Every algorithm known to us, for finding the median of a list of n numbers takes time $\Omega(n^2)$.

(iv) $\log_2 n = \Theta(\log_3 n)$.

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

2. Write down the worst case running time of each of the following code fragments as a function of n. Use the Θ notation to express your answers. Show your work to receive partial credit.

   (a) The arguments to this function are an element k and a list L of length n.

```
function scan(k, list L)
    i ← 1
    while k ≠ L[i] do
        i ← i + 1 if i = n + 1 then
            return "not found"
    return i
```

   (b) The arguments to this function are two $n \times n$ matrices A and B. The function computes the matrix product $A \cdot B$ and returns the resulting $n \times n$ matrix C.

```
function matrixMult(A, B)
    for i ← 1 to n do
        for j ← 1 to n do
            C[i, j] ← A[i, 1] · B[1, j] + A[i, 2] · B[2, j] + ··· + A[i, n] · B[n, j]
    return C
```

(c) The arguments to this function are an element k and a list L of length n.

```
function fastScan(k, list L)
    left ← 1; right ← n
    while left ≤ right do
        mid ← (left + right)/2
        if L[mid] = k then
            return mid
        else if L[mid] < k then
            left ← mid + 1
        else if L[mid] > k then
            right ← mid − 1

    return 0
```

(d)

```
for i ← 1 to n do
    for j ← n downto i do
        print("hello")
```

3. Given a list L of length n ($n \geq 3$), an element $L[i]$, $1 < i < n$, is a *local minimum* if $L[i-1] \geq L[i]$ and $L[i] \leq L[i+1]$. For example, if $L = [3, 1, 7, 7, 7, 2, 11]$ then the elements 1, 7 (the middle one), and 2 are all local minima.

   (a) A length-n list L is called *good* if $L[1] \geq L[2]$ and $L[n-1] \leq L[n]$. Argue (in 2-3 sentences) that any good list has a local minimum.

   (b) Describe an algorithm (using clear pseudocode) that takes as input a good list L and returns the index of a local minimum in L. If L has several local minima, your algorithm can return the index of any of these. It is required that your algorithm run in time that is asymptotically *faster* than $\Theta(n)$.
   Hint: Start by looking at the middle element of L and its neighbors on either side. Depending on how the middle element of L compares to its neighbors, your algorithm can determine if (i) the middle element is a local minimum or (ii) which of the two halves of L is guaranteed to contain a local minimum.

   (c) Express the worst case running time of your algorithm as a function of n (in Θ notation). Here n is the size of the input list L.

4. We want to determine if a given list L with n elements has a *majority* element and if so return it. (Recall that a *majority* element in a length-n list L is one that occurs *more* than $n/2$ times.)

   Here is a simple randomized algorithm for this problem that runs in $\Theta(n)$ time.

```
function majority(L)
    n ← length(L)
    Comment: pick a random index i between 1 and n
    i ← random(1, n)

    Comment: Count the number of time L[i] occurs in the list
    count ← 0
    for j ← 1 to n do
        if L[i] = L[j] then
            count ← count + 1

    Comment: Check if count is more than n/2
    if count > n/2 then
        return L[i]
    else
        return “no majority”
```

   (a) This algorithm does not always produce the correct answer. Explain the circumstances under which it produces an incorrect answer (1-2 sentences). Specifically, discuss whether the algorithm can produce an incorrect answer for each of the two cases: (i) L has a majority element and (ii) L does not have a majority element.

   (b) What is the maximum probability that the algorithm produces an incorrect answer?

(c) Let us suppose that we want to reduces the probability of this algorithm producing an incorrect output to at most 1/10. What changes would you make to the algorithm? You can describe your changes in words, no need for pseudocode.

5. Here are two problems on the *stable marriage* problem.

(a) Show the execution of the Gale-Shapley algorithm on an input with 3 men and 3 women having the following preferences:

$$m_1 : w_2 > w_1 > w_3 \qquad m_2 : w_2 > w_1 > w_3 \qquad m_3 : w_1 > w_2 > w_3$$

$$w_1 : m_2 > m_1 > m_3 \qquad w_2 : m_3 > m_1 > m_2 \qquad w_3 : m_3 > m_2 > m_1$$

Let us suppose that whenever your algorithm has choice of "free" men to pick for making a proposal, it picks the man with lowest index.

(b) Consider the *stable roommate* problem in which we are given n individuals (for even n) and each individual has preferences over the remaining $n - 1$ individuals. (So in this problem, the individuals are not divided into two groups, as in the *stable marriage* problem.) Unlike the stable marriage problem, there are instances of the stable roommate problem for which no solution exists. Consider the following example with 4 individuals A, B, C, and D with the following preferences:

$$A : B > C > D, \qquad B : C > A > D, \qquad C : A > B > D, \qquad D : A > B > C.$$

Prove that for this input, there is no solution.