Homework #3
Introduction to Algorithms/Algorithms 1
600.363/463
Spring 2016

Due on: Thursday, Feb 18th, 11.59pm
Late submissions: will NOT be accepted
Format: Please start each problem on a new page.
Where to submit: On blackboard, under student assessment
Please type your answers; handwritten assignments will not be accepted.
To get full credit, your answers must be explained clearly,
with enough details and rigorous proofs.

February 11, 2016

1 Problem 1 (13 points)

Let's say that a pivot provides $x|n-x$ separation if x elements in an array are smaller than the pivot, and $n-x$ elements are larger than the pivot.

Suppose Bob knows a secret way to find a good pivot with $\frac{n}{3}|\frac{2n}{3}$ separation in constant time. But at the same time, Alice knows her own secret technique, which provides $\frac{n}{4}|\frac{3n}{4}$ separation and works in constant time.

Alice and Bob applied their secret techniques as a subroutine in QuickSort algorithm. Whose algorithm works **asymptotically** faster? Prove your statement.

2 Problem 2 (13 points)

Alice and Bob are solving a problem. They are given a matrix A_n of size $n \times n$, where elements are sorted along every column and every row. For example, consider the case when $n = 4$ and matrix A_4 is as following:

$$A_4 = \begin{bmatrix} 3 & 5 & 9 & 12 \\ 4 & 8 & 10 & 32 \\ 12 & 13 & 29 & 43 \\ 16 & 19 & 30 & 60 \end{bmatrix}$$

They need to provide an algorithm which takes integer x as an input, and checks if x is an element of matrix A_n or not. Both Alice and Bob decided to apply divide and conquer method.
Alice's algorithm is quite simple, she uses a binary search routine to check each row for the input integer x.
Bob's algorithm is more advanced. Bob found out that if matrix A_n is represented in block-view:

$$A_n = \left[\begin{array}{c|c} A^1 & A^2 \\ \hline A^3 & A^4 \end{array}\right],$$

where all matrices A^1, A^2, A^3, A^4 are of the size $\frac{n}{2} \times \frac{n}{2}$, then he can compare element x with $A^4[1,1]$ and consider only three options:

1. $x = A^4[1,1]$ then his algorithm can output "yes".
2. $x < A^4[1,1]$ then he knows that $\forall i, j < n/2 : \quad x < A^4[i,j]$, thus he knows for sure that $x \notin A^4$, and only need to recursively apply his algorithm to check if $x \in A^1$, $x \in A^2$ or $x \in A^3$.
3. $x > A^4[1,1]$ then he knows that $\forall i, j < n/2 : \quad x > A^1[i,j]$, thus he knows for sure that $x \notin A^1$, and only need to recursively apply his algorithm to check if $x \in A^2$, $x \in A^3$ or $x \in A^4$.

Whose algorithm is faster on the worst case input? Prove your statement.

3 Problem 3 (24 points)

Resolve the following recurrences. Use Master theorem, if applicable. In all examples, assume that $T(1) = 1$. To simplify your analysis, you can assume that $n = a^k$ for some a, k.

1. $T(n) = 2T(n/8) + n^{\frac{1}{5}} \log n \log \log n$
2. $T(n) = 16T(n/2) + \sum_{i=1}^{n} i^3 \ln(e + \frac{1}{i})$
3. $T(n) = 4T(n-2) + 2^{2n}$

4. $T(n) = T(n/3) + n^3 \log n + 2n^2 - \sqrt{\log(n+1)}$

5. $T(n) = 8T(n/2) + n^3 - 8n \log n$

6. $T(n) = T(n/2) + \log n$

7. $T(n) = T(n-1) + T(n-2)$

8. $T(n) = 3T(n^{\frac{2}{3}}) + \log n$