Homework #3
Solutions
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 20, 2016

1 Problem 4 (13 points)

We will say that pivot provides $x|n-x$ separation if x elements in array are smaller than the pivot, and $n-x$ elements are larger than the pivot.

Suppose Bob knows the 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 separation $\frac{n}{4}|\frac{3n}{4}$, her technique also works in constant time.

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

To reorder the elements around the pivot at each step takes O(n), which gives us the relation $T_B(n) = T_B(\frac{n}{3}) + T_B(\frac{2n}{3}) + O(n)$ for Bob and $T_A(n) = T_A(\frac{n}{3}) + T_A(\frac{2n}{3}) + O(n)$ for Alice.
We know that in case of separation $\frac{n}{2}|\frac{n}{2}$ we have recurrence: $T_C(n) = 2T_C(\frac{n}{2}) +$

$O(n)$, from merge sort procedure we know that $T_C(n) = \Theta(n \log n)$. We will use it as initial guess for $T_A(n)$ and $T_B(n)$ and will prove it by substitution.

1. (a) For large enough C_B, n_0 and $\forall n \geq n_0$, $T_B(n) \leq C_B n \log n$.
   BC: Trivial.
   IH: $\forall k < n : \;\; T_B(k) \leq C_B k \log k$.
   IS: $T_B(n) = T_B(\frac{n}{3}) + T_B(\frac{2n}{3}) + Cn \leq C_B \frac{n}{3} \log(\frac{n}{3}) + C_B \frac{2n}{3} \log(\frac{2n}{3}) + Cn \leq C_B n \log n - (C_B \frac{\log 3}{3} + C_B \frac{2 \log \frac{3}{2}}{3} - C)n \leq C_B n \log n$. Where last inequality holds for $C_B \geq \frac{3C}{\log 3 + \log \frac{3}{2}}$.
   Therefore $T_B(n) = O(n \log n)$.

   (b) For small enough positive C_B and large enought n_0 and $\forall n \geq n_0$, $T_B(n) \geq C_B n \log n$. Using same idea as below. We will show only induction step. $T_B(n) = T_B(\frac{n}{3}) + T_B(\frac{2n}{3}) + Cn \geq C_B \frac{n}{3} \log(\frac{n}{3}) + C_B \frac{2n}{3} \log(\frac{2n}{3}) + Cn = C_B n \log n + (C - C_B \frac{\log 3}{3} - C_B \frac{2 \log \frac{3}{2}}{3})n \geq C_B n \log n$, where last inequality holds for $C \geq \frac{3C}{\log 3 + \log \frac{3}{2}}$. Therefore $T_B(n) = \Omega(n \log n)$.

2. (a) For large enough C_A, n_0 and $\forall n \geq n_0$, $T_A(n) \leq C_A n \log n$. Same idea as for $T_B(n)$ we will just show induction step.$T_A(n) = T_A(\frac{n}{4}) + T_A(\frac{3n}{4}) + Cn \leq C_A \frac{n}{4} \log(\frac{n}{4}) + C_A \frac{3n}{4} \log(\frac{3n}{4}) + Cn \leq C_A n \log n - (\frac{1}{4} C_A \log 4 + \frac{3}{4} C_A \log \frac{4}{3} - C)n \leq C_A n \log n$, where last inequality holds when $C_A \geq \frac{4C}{\log 4 + 3 \log \frac{4}{3}}$. Therefore $T_B(n) = O(n \log n)$.

   (b) For small enough positive C_A and large enough n_0 and $\forall n \geq n_0$, $T_A(n) \geq C_A n \log n$. Same idea as for $T_B(n)$ we will just show induction step. $T_A(n) = T_A(\frac{n}{4}) + T_A(\frac{3n}{4}) + Cn \geq C_A \frac{n}{4} \log(\frac{n}{4}) + C_A \frac{3n}{4} \log(\frac{3n}{4}) + Cn = C_A n \log n + (C - \frac{1}{4} C_A \log 4 - \frac{3}{4} C_A \log \frac{4}{3})n$, where last inequality holds when $C_A \leq \frac{4C}{\log 4 + 3 \log \frac{4}{3}}$. Therefore $T_B(n) = \Omega(n \log n)$.

   Therefore $T_B(n) = \Theta(n \log n) = T_A(n)$. Asymptotically the solutions are the same.

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 binary search routine to check each row. 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 only need recursevly check if $x \in A^1$, if $x \in A^2$ and if $x \in A^3$.
3. $x > A^4[1,1]$ then he knows that $\forall i, j < n/2 : \quad x > A^1[i,j]$, thus he only need recursevly check if $x \in A^2$, if $x \in A^3$ and if $x \in A^4$.

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

Alice's algorithm, in the worst case, spends $\Theta(\log n)$ time in each of n rows, not finding a match until the n^{th} row and is therefore $\Theta(n \log n)$.

Bob's algorithm, in the worst case, recurses through the whole matrix not finding a match until the last subproblem of size 1. To avoid confusion, let the size of the problem be D. For the initial problem $D = n^2$. This can be described by the recurrence $T(D) = 3T(\frac{D}{4}) + O(1)$, because we split into 3 subproblems of size $\frac{D}{4}$ and throw the remaining $\frac{D}{4}$ data away, and this decision takes constant time. $\forall n \geq n_0$, $O(1) \leq cn^{\log_4 3 - \epsilon} \approx cn^{.79-\epsilon}$ for $0 < \epsilon < .79$ and sufficiently large c, n_0. Therefore, $f(n) \in O(n^{\log 43 - \epsilon})$. Therefore, $T(n) = \Theta(D^{\log_4 3}) = \Theta(n^{2\log_4 3}) \approx \Theta(n^{1.5})$ by case 1 of the master theorem.

$\Theta(n \log n) = O(n^{2\log_4 3})$ but $\Theta(n \log n) \neq \Omega(n^{2\log_4 3})$
$\Theta(n^{2\log_4 3}) = \Omega(n \log n)$ but $\Theta(n^{2\log_4 3}) \neq O(n \log n)$
Therefore, Alice's algorithm is faster.

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. $T(n) = 2T(n/8) + n^{\frac{1}{5}} \log n \log\log n$

   $\forall n \geq n_0$, $n^{\frac{1}{5}} \log n \log\log n \leq n^{\frac{1}{5}} n^{\frac{1}{100}} n^{\frac{1}{100}} \leq n^{\frac{1}{4}} \leq cn^{\log_8 2-\epsilon} = cn^{\frac{1}{3}-\epsilon}$ for $0 < \epsilon < \frac{1}{100}$ and sufficiently large c, n_0. Therefore, $f(n) \in O(n^{\frac{1}{3}-\epsilon})$. Therefore, $T(n) = \Theta(n^{\frac{1}{3}})$ by case 1 of the master theorem.

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

   First of all let's estimate $f(n) = \sum_{i=1}^{n} i^3 \ln(e + \frac{1}{i})$.
   We know that $1 < \ln(e + 1/i) < 2$ then
   $\sum_{i=1}^{n} i^3 \ln(e + \frac{1}{i}) > \sum_{i=1}^{n} i^3 > \sum_{i=n/2}^{n} i^3 > \frac{n}{2}(\frac{n}{2})^3 > C_1 n^4$
   and
   $\sum_{i=1}^{n} i^3 \ln(e + \frac{1}{i}) < 2\sum_{i=1}^{n} i^3 < 2n^4 < C_2 n^4$. Thus $f(n) = \Theta(n^4) = \Theta(n^{\log_b^a})$, thus it's case 2 from Master Theorem. Thus $T(n) = \Theta(n^4 \log n)$.

3. $T(n) = 4T(n-2) + 2^{2n}$ $n = \log a$
   $T(n) = T(\log a) = 4T(\log a - \log 4) + 2^{2\log a} = 4T(\log \frac{a}{4}) + a^2$
   $T(n) = T(\log a) = S(a) = 4S(\frac{a}{4}) + a^2$.
   Case 3 of Master theorem: $a^2 = \Omega(a^{1+\varepsilon})$, and for $C = 1/2$:
   $4f(\frac{a}{4}) = \frac{4n^2}{16} = \frac{n^2}{4} \leq Cf(n) = Ca^2$ Thus $S(a) = \Theta(f(n)) = \Theta(a^2)$. But at the same time $T(n) = \Theta(2^{2n})$

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

   $\forall n \geq n_0$, $0 \leq cn^{log_3 1+\epsilon} = cn^{\epsilon} \leq n^3 \log n + 2n^2 - \sqrt{\log(n+1)}$ for $0 < \epsilon < 3$ and sufficiently large c, n_0. Therefore, $f(n) \in \Omega(cn^{\epsilon})$. $f(\frac{n}{3}) \leq cf(n)$ for $c = .5$. Therefore, $T(n) = \Theta(f(n)) = \Theta(n^3 \log n)$ by case 3 of the master theorem.

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

   $\forall n \geq n_0$, $0 \leq c_1 n^{log_2 8} = c_1 n^3 \leq n^3 - 8n \log n \leq c_2 n^{log_2 8} = c_2 n^3$ for $c_1 = .5$, $c_2 = 1$, and sufficiently large n_0. Therefore, $f(n) \in \Theta(n^3)$. Therefore, $T(n) = \Theta(n^3 \log n)$ by case 2 of the master theorem.

6. $T(n) = T(n/2) + \log n$
   $n = 2^k$
   $T(2^k) = T(2^k/2) + k = T(2^{k-1}) + k$
   $T(2^k) = S(k) = S(k-1) + k$
   Thus $T(n) = S(k) = \sum_{i=1}^{k} i = \Theta(k^2) = \Theta((\log n)^2)$.

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

   Let $T(n) \leq ca^n$ hold for smaller values of n. Then,
   $T(n) = T(n-1) + T(n-2) \leq ca^{n-1} + ca^{n-2} = ca^n + (\frac{c}{a} + \frac{c}{a^2} - c)a^n$
   where $(\frac{c}{a} + \frac{c}{a^2} - c)a^n$ is a positive term iff for some positive a, $a^2 - a - 1 \leq 0 \rightarrow a \leq \frac{1+\sqrt{5}}{2}$.
   Therefore $T(n) = O((\frac{1+\sqrt{5}}{2})^n)$

   Let $T(n) \geq ca^n$ hold for smaller values of n. Then,
   $T(n) = T(n-1) + T(n-2) \geq ca^{n-1} + ca^{n-2} = ca^n + (\frac{c}{a} + \frac{c}{a^2} - c)a^n$
   where $(\frac{c}{a} + \frac{c}{a^2} - c)a^n$ is a negative term iff for some positive a, $a^2 - a - 1 \geq 0 \rightarrow a \geq \frac{1+\sqrt{5}}{2}$.
   Therefore $T(n) = \Omega((\frac{1+\sqrt{5}}{2})^n)$
   Therefore $T(n) = \Theta((\frac{1+\sqrt{5}}{2})^n)$
   From $T(1) = 1$ we can conclude that constant $C = \left(\frac{1+\sqrt{5}}{2}\right)^{-1}$.

8. $T(n) = 3T(n^{\frac{2}{3}}) + \log n$
   $T(n) = T(a^k) = 3T((a^k)^{\frac{2}{3}}) + \log a^k$
   $T(a^k) = S(k) = 3S(\frac{2}{3}k) + \log a^k$
   $\log a^k = O(k^{\log_{\frac{3}{2}} 3-\epsilon}) \approx O(k^{2.7-\epsilon})$ for some ϵ
   Therefore $T(n) = S(k) = \Theta(k^{\log_{\frac{3}{2}} 3}) = \Theta((log_a n)^{\log_{\frac{3}{2}} 3})$