Homework #3 Solutions
Introduction to Algorithms
601.433/633
Spring 2020

Due on: Tuesday, February 25th, 12pm
Format: Please start each problem on a new page.
Where to submit: On Gradescope, please mark the pages for each question

1 Problem 1 (24 points)

Recall that when using the QuickSort algorithm to sort an array A of length n, we picked an element $x \in A$ which we called the *pivot* and split the array A into two arrays A_S, A_L such that $\forall y \in A_S, y \leq x$ and $\forall y \in A_L, y > x$.

We will say that a pivot from an array A provides $t|n-t$ separation if t elements in A are smaller than or equal to the pivot, and $n-t$ elements are strictly 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 separation $\frac{n}{4}|\frac{3n}{4}$, her technique also works in constant time.

Recall that in the QuickSort algorithm, we picked the pivot by picking an element x *randomly* from A.

Alice and Bob applied their secret techniques as subroutines in the QuickSort algorithm to pick pivots. Whose algorithm works **asymptotically** faster? Or are the runtimes **asymptotically** the same? Prove your statement.

Proof. 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: $\forall k < n : \quad T_B(k) \leq C_B k \log k$.
   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 enough 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.

$\square$

2 Problem 2 (13 points)

Resolve the **asymptotic complexity** of the following recurrences, i.e., solve them and give your answer in Big-Θ notation. 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.

Your final answer should be as simple as possible, i.e., it should not contain any sums, recurrences, etc.

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

   $\forall n \geq n_0$, $0 \leq c_1 n^{log_2 8} = c_1 n^3 \leq n^3 - 8n \log n \leq c_2 n^{log_2 8} = c_2 n^3$ 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.

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

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

   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 $C = \left(\frac{1+\sqrt{5}}{2}\right)^{-1}$.

5. $T(n) = 3T(n^{\frac{2}{3}}) + \log n$
$T(n) = T(a^k) = 3T((a^k)^{\frac{2}{3}}) + \log a^k$
$T(a^k) = S(k) = 3S(\frac{2}{3}k) + \log a^k$
$\log a^k = O(k^{\log_{\frac{3}{2}} 3-\epsilon}) \approx O(k^{2.7-\epsilon})$ for some ϵ
Therefore $T(n) = S(k) = \Theta(k^{\log_{\frac{3}{2}} 3}) = \Theta((log_a n)^{\log_{\frac{3}{2}} 3})$ using 1 of Master's Thm.

3 Problem 3 (13 points)

Let A and B be two sorted arrays of n elements each. We can easily find the median element in A – it is just the element in the middle – and similarly we can easily find the median element in B. (Let us define the median of $2k$ elements as the element that is greater than $k-1$ elements and less than k elements.) However, suppose we want to find the median element overall – i.e., the nth smallest in the union of A and B.

Give an $O(\log n)$ time algorithm to compute the median of $A \cup B$. You may assume there are no duplicate elements.

As usual, prove correctness and the runtime of your algorithm.

Proof. We call UnionMedian(A, B).

The correctness of the algorithm follows from inducting on the length of the arrays n. The base case, corresponding to $|A|, |B| \leq 2$ is trivially true.

IH: Assume that for all pairs of sorted arrays X ,Y of length $n' < n$ each the algorithm correctly returns the median of $X \cup Y$.

IS:

Algorithm 1 UnionMedian(X, Y)

if $|X| = |Y| \leq 2$ **then**
  return brute force compute median(X, Y)
else if median$(X) <$ median (Y) **then**
  return UnionMedian($X[|X|/2:], Y[:|Y|/2]$)
else
  return UnionMedian($X[:|X|/2], Y[|Y|/2:]$)
end if

- Case i). Median(X) $<$ Median(Y). Since X and Y are sorted, we know that the $n/2$ elements in $Y[|Y|/2:]$ are larger than the median of $X \cup Y$ since otherwise the elements $X[:|X|/2] \cup Y[:|Y|/2]$ would all be smaller than the median. A similar argument can be made to show that the $n/2$ elements in $X[|X|/2]$ are smaller than the median of $X \cup Y$. The result follows by applying the IH on the smaller arrays.

- Case ii). Median(X) $>$ Median(Y). An analogous argument to Case i) follows.

To prove running time, notice that in each step of the algorithm at least $1/2$ of X and $1/2$ of Y is discarded. Hence the running time of the algorithm can be upper bounded by the following recurrence

$$\begin{aligned} T(n) &= 2T(\frac{n}{2}) + C \\ &= \sum_{i=1}^{\log n} C = O(\log n) \end{aligned}$$

□