

## Solutions to Homework 2

Dmitriy Morozov  
Madhuwanti Vaidya

February 14, 2004

Note that these are the ideas of the solutions to the homework problems, all the details are not considered. The solutions are meant as a general guide. If you find any mistakes, send me (Dmitriy) an email.

### Pr. 1 (7-2)

- a. In an array of size  $n$  the probability of choosing any element as the pivot is  $1/n$ . Any element is chosen as pivot out of the possible  $n$  choices. Thus the probability becomes  $1/n$ .

Thus,

$$E(X_i) = 1 \cdot Pr(i^{th} \text{ smallest element is chosen as pivot}) + 0 \cdot Pr(\text{otherwise}),$$

$$E(X_i) = 1/n$$

- b.

$$E(T(n)) = E\left(\sum_{q=1}^n X_q(T(q-1) + T(n-q) + \theta(n))\right)$$

The summation over all  $q$  is taken, this gives the probability of choosing the  $q^{th}$  smallest element in the array. Partitioning is done by taking this element as the pivot. The array will be partitioned into two parts of sizes  $q-1$  and  $n-q$ , thus the terms  $T(q-1)$  and  $T(n-q)$ . Also actually partitioning requires  $\theta(n)$  time.

- c.

$$\begin{aligned} E(T(n)) &= E\left(\sum_{q=1}^n X_q(T(q-1) + T(n-q) + \theta(n))\right) \\ &= 2 \times E\left(\sum_{q=0}^{n-1} X_q(T(q) + \theta(n))\right) \\ &= 2 \times \sum_{q=0}^{n-1} E(X_q(T(q))) + \theta(n) \\ &= 2 \times \sum_{q=0}^{n-1} E(X_q)E(T(q)) + \theta(n) \\ &= 2 \times \sum_{q=0}^{n-1} (1/n)E(T(q)) + \theta(n) \\ &= (2/n) \times \sum_{q=0}^{n-1} E(T(q)) + \theta(n) \end{aligned}$$

QED

d.

$$\begin{aligned}
\sum_{k=1}^{n-1} k \log(k) &= \sum_{k=1}^{n/2-1} k \log(k) + \sum_{k=n/2}^{n-1} k \log(k) \\
&\leq \sum_{k=1}^{n/2-1} k \log(n/2) + \sum_{k=n/2}^{n-1} k \log(n) \\
&= \log(n/2) \sum_{k=1}^{n/2-1} k + \log(n) \sum_{k=n/2}^{n-1} k \\
&= \log(n/2)(n/2-1)(n/4) + \log(n)((n/2)(n-1) - (n/2-1)(n/4)) \\
&= \log(n/2)(n^2/8 - n/4) + \log(n)(3n^2/8 - n/4) \\
&= \log(n)(n^2/8 - n/4) - \log(2)(n^2/8 - n/4) + \log(n)(3n^2/8 - n/4) \\
&= (n^2/2) \log(n) - n^2/8 - (\log(n)(n/2) - n/4) \\
&\leq (n^2/2) \log(n) - n^2/8
\end{aligned}$$

e. Assume that  $T(k) \leq ak \log k - bk$  for all  $k < n$ . Let us show that  $T(n) \leq an \log n - bn$ .

$$E(T(n)) = (2/n) \times \sum_{q=0}^{n-1} E(T(q)) + \theta(n)$$

Since in the summation  $q < n$ , we know from our assumption that  $E(T(q)) \leq aq \log q - bq$ . Therefore, we get

$$\begin{aligned}
E(T(n)) &\leq (2/n) \sum_{q=0}^{n-1} (aq \log(q) - bq) + cn \\
&= (2/n) \left( \sum_{q=0}^{n-1} aq \log(q) - \sum_{q=0}^{n-1} bq \right) + cn \\
&\leq (2/n) (an^2 \log(n)(1/2) - (an^2/8) - (bn^2)/2 + (bn)/2) + cn \\
&= an \log(n) - (an/4) - bn + b/2 + cn \\
&= (an \log(n) - bn) + (b/2 - (an)/4 + cn) \\
&\leq an \log(n) - bn
\end{aligned}$$

The last,  $\leq$  follows from the fact that we can always choose  $a$  so that  $an/4 > cn + b/2$  for sufficiently large  $n$ . Therefore, by the principle of mathematical induction,  $E(T(n)) = O(n \log(n))$ . Since quicksort is a comparison sort, the lower bound on its running time is  $\Omega(n \log n)$ . Therefore,  $E(T(n)) = \Theta(n \log n)$ .

**Pr. 2 (9-2)**

a. For simplicity suppose that  $n$  is even (the case where  $n$  is odd can be treated similarly). There are  $n/2 - 1$  elements that are less than the (non-weighted) median which we shall call  $x_m$  and  $n/2$  elements that are larger than  $x_m$ . If the weight of each element in the array is  $1/n$ , then the total weight of the elements that are less than the median is  $\sum_{x_i < x_m} w_i = (1/n) \cdot (n/2 - 1) = 1/2 - 1/n < 1/2$ , similarly the weight of the elements that are larger than the median is equal  $1/2$ . Therefore, by definition of the weighted median the (non-weighted) median  $x_m$  is the weighted median of this array.

- b. 1: Sort the elements of the array by their value (*not weight*)
- 2:  $total \leftarrow 0$
- 3: **for**  $i \leftarrow 1$  to  $n$  **do**
- 4:    $total \leftarrow total + A[i].weight$
- 5:   **if**  $total > 1/2$  **then**
- 6:     **return**  $i$
- 7:   **end if**
- 8: **end for**

Sorting the elements takes time  $O(n \lg n)$ , **for**-loop takes  $O(n)$ .

- c. Note that this is a rough idea of one way in which the algorithm could work (it might be missing some details).  
 WEIGHTEDMEDIAN( $A, weight_l = 1/2, weight_u = 1/2$ )
  - 1: Find the (non-weighted) median of  $A$  using linear time SELECT
  - 2: Partition the array around the median of  $A$ , and compute the weights  $w_l, w_u$  of the elements the are smaller and larger than the median
  - 3: **if**  $w_l < weight_l$  and  $w_u \leq weight_u$  **then**
  - 4:   **return** the index of the median
  - 5: **else if**  $w_l < weight_l$  **then**
  - 6:   WEIGHTEDMEDIAN( $A, weight_l, weight_u - w_u$ )
  - 7: **else**
  - 8:   WEIGHTEDMEDIAN( $A, weight_l - w_l, weight_u$ )
  - 9: **end if**

The recurrence that describes the running time of this algorithm is  $T(n) = T(n/2) + \Theta(n)$ . From the Master Method we get that  $T(n) = \Theta(n)$ .

**Pr. 3 (6.5-8)**

Put the first elements of each list into a minheap (this construction will take time  $O(k)$ ). Repeat the following procedure  $n$  times: remove the minimum element from minheap and place it into the sorted array, add an element from the array from which this minimum element came from to minheap. This step will take time  $O(\lg k)$  and it will be performed for each element, so the total time will be  $O(n \lg k)$ .  $O(k + n \lg k) = O(n \lg k)$ .

**Pr. 4 (12-1)**

- a. The asymptotic running time would be  $O(n^2)$ , as the tree would be completely unbalanced. Inserting  $n$  items with equal in a tree is as bad as inserting  $n$  items in a list and thus the running time is  $O(n^2)$ .
- b. The asymptotic running time with this method would be  $O(n \times \log(n))$ . This is because the tree will now be balanced because of the flag that is maintained.
- c. The asymptotic running times for this method would be  $O(n)$ , as each insertion would require only constant time. It would be compared only with the root and then inserted into the list stored at the root. Thus running time would be  $O(n)$ .
- d. Worst case: This would be if each time  $x$  is set to left( $x$ ) [or right( $x$ )]. Now the tree would grow only on one side and would be unbalanced. The running time is  $O(n^2)$ . Average case: right( $x$ ) and left( $x$ ) would be chosen on average half the time each, so the resulting tree would be balanced.

**Problem 4, Homework 1 (3-3)**

$1, n^{1/\lg n}$

$\lg(\lg^* n)$

$$\frac{\lg^*(n) \lg^*(\lg n)}{2^{\lg^*(n)}}$$

$$\frac{\ln \ln n}{\sqrt{\lg n}}$$

$$\frac{\lg^2 n}{2^{\sqrt{2} \lg n}}$$

$$\sqrt{2}^{\lg n}$$

$$n, 2^{\lg n}$$

$$n \lg n, \lg(n!)$$

$$n^2, 4^{\lg n}$$

$$n^3$$

$$(\lg n)!$$

$$(\lg n)^{\lg n}, n^{\lg \lg n}$$

$$(3/2)^n$$

$$2^n$$

$$n2^n$$

$$e^n$$

$$n!$$

$$(n+1)!$$

$$2^{2^n}$$

$$2^{2^{n+1}}$$