University of Waterloo
CS240 Fall 2017
Assignment 2

Problem 1 [3+4+3+12 = 22]

a) $T(n) = cn + T(n-i-1) + T(i)$ where $1 \leq i \leq n$ and $T(1) = d$ where c, d are constants. Finding the mean and partition takes cn time in total.

b) Consider the algorithm is called with m elements $a+tk, a+(t+1)k, \ldots, a+(t+m-1)k$. For the mean of these elements, we have:

$mean = 1/m \sum_{j=t}^{t+m-1}(a + jk) = a + k/m \sum_{j=t}^{t+m-1} j = a + k(t + m/2)$

Hence, the mean of the elements is roughly the $(m/2)$-th element; i.e. the pivot is the median of the elements in every call of the algorithm. Consequently, the best case happens and the algorithm runs in $\Theta(n \log n)$.

c) Consider items $1, 2, 4, \ldots, 2^n$. In every call, the mean of the subarry is close to the last item; i.e. the $(m-1)$-th item will be selected. Hence the runtime complexity is $\Theta(n^2)$.

Problem 2 [10 marks]

Solution 1: Consider every $(\log n)$-th element; i.e. $A[0], A[\log n], A[2\log n], \ldots$. By the given ordering property, these elements are in sorted order. Similarly, $A[1], A[1+\log n], A[1+2\log n], \ldots$ are also in sorted order. In general, elements $A[k], A[k+\log n], A[k+2\log n], \ldots$ will be in sorted order, for $k = 0, 1, \ldots, \log n - 1$. So, we can obtain $\log n$ sorted lists, each with $\Theta(\frac{n}{\log n})$ elements in $O(n)$ time. Next, perform a $O(\log n)$-way merge using a min heap; i.e. place the samllest element from each sorted list onto the min heap, use `deleteMin` to remove the smallest item and then insert the next item from the list the smallest element was removed from. Since the min heap contains $O(\log n)$ elements, heap operations will take $O(\log \log n)$ time so overall time complexity is $O(n \log \log n)$ time.

Solution 2: Partition the array into blocks of size $\log n$; i.e. first block is $A[0, \log n - 1]$, second block is $A[\log n, 2\log n - 1]$ and so on. Then, starting from the left, take every two consecutive blocks and sort the elements. Number of blocks is $\Theta(\frac{n}{\log n})$ and the sorting of two consecutive blocks will take $O((\log n)(\log \log n))$ time with overall time complexity $O(n \log \log n)$.

This document is for the exclusive use of y87feng.

Problem 3 [3+3+5=11 marks]

a) In the **best-case**, we may get lucky and $A[i] == k$ with the first pick of the random variable. This has running time of $\Theta(1)$

b) In the **worst-case**, the algorithm (although very unlikely) may never pick the desired index so would have an infinite running time.

c) Let $T(n)$ be the expected running time of *find-index*. At each iteration of, there is a constant number c of elementary operations, such as obtaining the random index, comparing $A[i]$ with k, branching through the if statement, etc. With a $\frac{1}{n}$ probability we match and the function completes in constant time d and with a $\frac{n-1}{n}$ probability we have a subproblem the same size as the original problem, $T(n)$. Hence,
$T(n) = \frac{1}{n}d + \frac{n-1}{n}T(n) + c$
$\frac{1}{n}T(n) = \frac{1}{n}d + c$
$T(n) = d + cn$ where c, d are constants
So $T(n)$ is $\Theta(n)$.

Problem 4 [6+7+7=20 marks]

a) Each loonie is genuine or counterfeit, but the two cases where all coins are genuine or all are counterfeit are excluded. The total number of possible outcomes is then $2^n - 2$. Each weighing has exactly 3 possible outcomes, so if an algorithm performs at most k weighingsm the number of different possible outcomes the algorithm could return is at most 3^k. So we must have $3^k \geq 2^n - 2$. Solving for k gives the lower bound of $\lceil \log_3(2^n - 2) \rceil$ weighings.

b) Using the algorithm from part c), we need 3 weighings when $n = 4$. Notice that $\lceil \log_3(2^4 - 2) \rceil = \lceil \log_3 14 \rceil = 3$

c) Perform exactly $n - 1$ weighting between the pairs of loonies L_1 and L_i for $2 \leq i \leq n$. L_1 is counterfeit if and only if all weighings determine that L_1 weighs at most as much as any of the other loonies, with L_1 weighing less than at least one other loonie since there is at least one genuine loonie; the counterfeit loonies are those that weigh the same as L_1 and the genuine loonies are those that weigh more than L_1.

The case where L_1 is genuine is similar.

This algorithm is $\Theta(n)$. Obaserve that $\lceil \log_3(2^n - 2) \rceil > \log_3(2^{n-1})$ for $n \geq 2$, and this is equal to $\log_3(2)(n-1)$, so the lower bound from part a) is $\Omega(n)$. Therefore the algorithm is asympototically optimal in the number of weighings.

This document is for the exclusive use of y87feng.

Problem 5 [3+5+10 marks]

a) Swap item $h[i]$ with $h[n-1]$ to maintain the heap structure.
If $h[i] > \mathit{parent}$ bubble-up, else bubble-down to fix the heap ordering property.
Swap is $O(1)$ and bubble-up, bubble-down are $O(\log n)$ (as shown in class).

y87feng

This document is for the exclusive use of y87feng.