CS 498 algorithm Spring 2019 Ray Ying (xinruiy2@illinois.edu)
Homework 1 Problem 2 Aditya Pillai (apillai4@illinois.edu)

Solution:

1. The formal description of the algorithm is as following:

   (a) Choose one element uniform at random from the array and find its rank.

   (b) If the rank it's some where between $n/4$ to $3n/4$, then we choose this element as pivot. Otherwise, we repeat the previous step.

   (c) Perform quicksort on that pivot and divided the array into three subarray: those smaller than pivot, those larger than pivot, and the pivot itself.

   (d) Perform the small algorithm above on the smaller subarrays and concatenate the result.

   The Pseudocode for the algorithm is in the last page.

2. Let total running time of the randomized quicksort $T(n)$, and $E(T(n))$ be the expected running time of the randomized quicksort. For $T(n)$, the running time for each sort time will have two parts: choosing the pivot, sorting based on the good pivot. Since you will always end up with the good pivot and do the sorting. The expected running time for ranking good pivot is n(the list size) and the running time for sorting list based on this good pivot is n as well. Let z be the number of choosing bad pivots before getting the good pivot, the expected running time for choosing the bad pivot is $\sum_z 1/2^z \cdot zn = n \cdot \sum_z 1/2^z \cdot z = 2n$. Therefore, the total expected running time will be $4n$. As for the recursive steps, the running time will be based on the rank of the pivot. Let i be the rank of the pivot ranging from $[n/4, 3n/4]$, the expected running time for the recursive steps will be: $\sum_i (1/(n/2)) \cdot (T(i-1) + T(n-i))$. Therefore, the total expected $E(T(n)) = 4n + \sum_i (2/(n)) \cdot (T(i-1) + T(n-i))$. With base case $T(n) = 0$, in each level of recursive tree, the sum is in $O(n)$. There will be at most $log_{3/4}n$ levels. Therefore, $E(T(n))$ is $O(nlogn)$.

3. Let T_i be the comparisons performed at level i of the recursion. Then the run-time of the algorithm is $\sum_{i=1}^{M} T_i$, M is the number of levels and $M \leq \log_{4/3} n$ since the pivot is always chosen in a "good" region. Let $T_{i,k}$ be the number of comparisons done for the kth subarray in the ith recursive call, and n_k be the size of the subarray k at level i
   $\mathbb{E}(T_i) = \sum_{k=1}^{2^i} \mathbb{E}(T_{i,k}) = \sum_{k=1}^{2^i} 2n_k = 2n$ because the sum of the size of subarrays at any level is n and in expectation it take 2 tries to pick a good pivot, each try costs n_k comparisons $\Pr(T_i > 8n/3) \leq \mathbb{E}(T_i)/(8n/3) = 3/4$. The number of comparisons it takes across different levels is independent so the probability it takes more than 8n/3 comparisons across all levels is $(\frac{3}{4})^M \leq (\frac{3}{4})^{\log_{4/3}(n)} = 1/n$.
   Then by the above formula the probability that quicksort takes at at most $\frac{8n}{3} \log_{4/3} n$ time is at at least $1 - 1/n$ (we can improve this some $c > 1$ and show $1 - 1/n^c$ probability by choosing a constant bigger than 8/3)

■

Algorithm 1 Randomized Quicksort

Require: A is the array of size n,lo is the lowest index and hi is the highest index.

```
RandomizedQuicksort(A, lo,hi):
n ← hi − lo + 1
i ← random(lo...hi)
Count ← 0
for j ← lo to hi do
  if A[j] < A[i] then
    Count = Count +1
  end if
end for
while Count < n/4 || Count > 3n/4 do
  i ← random(lo..hi)
  for j ← lo to hi do
    if A[j] < A[i] then
      Count = Count +1
    end if
  end for
end while
temp ← A[Count]
A[Count] ← A[i]
A[i] ← temp
leftstart ← lo
rightstart ← Count + 1
while start < Count do
  if A[leftstart] > A[Count] then
    temp ← A[leftstart]
    A[leftstart] ← A[rightstart]
    A[rightstart] ← temp
    rightstart = rightstart + 1
  else
    leftstart = leftstart + 1
  end if
end while
RandomizedQuicksort(A, lo, Count − 1)
RandomizeQuicksort(A, Count + 1, hi)
```