CSCI 3104 PS8a

Jonathan Phouminh

TOTAL POINTS

12.5 / 14

QUESTION 1

1 2 / 2

- ✓ **+ 2 pts Totally correct.**
- **+ 1.5 pts** Only one pair is incorrect.
- **+ 1 pts** Partially correct.
- **+ 0 pts** Incorrect/Not attempted.

QUESTION 2

2 3 / 3

- ✓ **+ 0.75 pts Correct first iteration**
- ✓ **+ 0.75 pts Correct second iteration**
- ✓ **+ 0.75 pts Correct third iteration**
- ✓ **+ 0.75 pts Correct fourth iteration**
- **+ 0 pts** You are not following the partition procedure presented in class. On future assignments and quizzes/exams, please use the partition procedure presented in class. Need to swap pivot and first element before partitioning.
- **+ 0.3 pts** Correctly chooses pivot for fourth iteration, but does not show fourth iteration explicitly.
- **+ 0.5 pts** Incorrect partitions with insufficient supporting work to illustrate the logic or array manipulations.
- **+ 1 pts** Incorrect partitioning procedure, but some supporting work
- **+ 0.3 pts** Correct pivot for second iteration, but incorrect partitioning
- **+ 0.3 pts** Incorrect pivot for third iteration
- **+ 0.5 pts** Clearly show how the array A is being manipulated. Your work is extremely unclear.
- **+ 0 pts** Incorrect pivot and partitioning at second iteration
- **- 1 pts** Choose end element as pivot, as discussed in class
- **+ 0 pts** Empty solution submitted or incorrect work shown.

QUESTION 3

3 1 / 1

- ✓ **+ 1 pts Explain correctly that can avoid choosing the bad pivot.**
- **+ 0.5 pts** The Median of Medians algorithm doesn't necessarily select the best pivot or actual median; rather, it avoids bad pivots.
- **+ 0 pts** Incorrect.

QUESTION 4

4 2.5 / 4

- ✓ **+ 1 pts Divides the elements correctly into groups of size 5.**
- ✓ **+ 0.5 pts Correctly sorts each group**
- ✓ **+ 0.5 pts Takes median of each group**
- ✓ **+ 0.5 pts Correctly finds median of medians**
- **+ 0.5 pts** Partitions A around median of medians
- **+ 0.5 pts** Uses median of median algorithms again on right sub-array of A.
- ✓ **+ 0 pts Please adhere to the partition algorithm from class. This will be expected on quizzes and exams**
- **+ 0.5 pts** Correct 4th-largest element
- **+ 0 pts** The problem is asking you to work through the algorithm.
- **- 0.5 pts** Extraneous element in array
- **+ 0 pts** Student Dropped
- **+ 0 pts** Incorrect or Not attempted

QUESTION 5

4 pts

5.1 2 / 2

- ✓ **+ 2 pts Correct**
- **+ 1.5 pts** Correct reasoning, but you should provide

the explicit recurrence relation if you are referencing it.

+ **1 pts** Correctly identifies divide and conquer strategy, but does not make the strategy precise

+ **1 pts** Discusses the divide and conquer strategy, but your reasoning is incorrect or is missing details

+ **0 pts** Incorrect answer

+ **1 pts** Incorrect complexity

+ **1.5 pts** Makes incorrect assumptions about the structure of the problem, but otherwise correctly explained why the algorithm runs in sub-linear time

+ **1.5 pts** Used little-o rather than Big-O

+ **1.5 pts** Algorithm adapts binary search, not quicksort

+ **0 pts** No answer

+ **1 pts** Need to discuss the techniques which give rise to this complexity

+ **0.5 pts** Need to discuss the techniques. You also need to explicitly discuss and solve the recurrence. The work you provided is not very descriptive or clear.

+ **1.5 pts** Solve the recurrence

The algorithm basically uses binary search, which is why we obtain sublinear time.

5.2 2 / 2

✓ + **2 pts** Correct

+ **0 pts** Incorrect, insufficient or no answer

+ **0.5 pts** Correctly modifies the base case. How do you handle the recursive calls?

+ **1.5 pts** Minor error in base case.

+ **1 pts** Missing base case

+ **0.5 pts** Missing base case and incorrect recursive calls

+ **0.5 pts** Your answer lacks sufficient detail or has major errors

+ **0.5 pts** Return statements are mismatched and you are not taking into account the left endpoints

+ **1 pts** Your return statements are mismatched

+ **0.5 pts** Modify the algorithm from class. Don't just provide linear search

+ **1.5 pts** Minor errors with recursive calls.

+ **0 pts** Student Dropped

+ **0.5 pts** Incorrect recursive calls.

Correct, but note that you can invoke findValley(A, mid+1, r) and findValley(A, p, mid-1). If mid is not a valley, you need not consider it again on future recursive calls.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 8a (14 points) **Fall 2019, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 8a (14 points) **Fall 2019, CU-Boulder**

1. (2 pts) If the arrays, $A = [12, 14, 23, 34]$ and $B = [11, 13, 22, 35]$ are merged, list the indices in A and B that are compared to each other. For example, $A[0], B[0]$ means that $A[0]$ is compared to $B[0]$.

   Solution.

   $A[0]$, $B[0]$
   $A[0]$,$B[1]$
   $A[1]$,$B[1]$
   $A[1]$,$B[2]$
   $A[2]$,$B[2]$
   $A[2]$,$B[3]$
   $A[3]$,$B[3]$
   $B[3]$

2. (3 pts) Illustrate how to apply the QuickSelect algorithm to find the $k = 4$th smallest element in the given array: `A = [5, 3, 4, 9, 2, 8, 1, 7, 6]` by showing the recursion call tree.

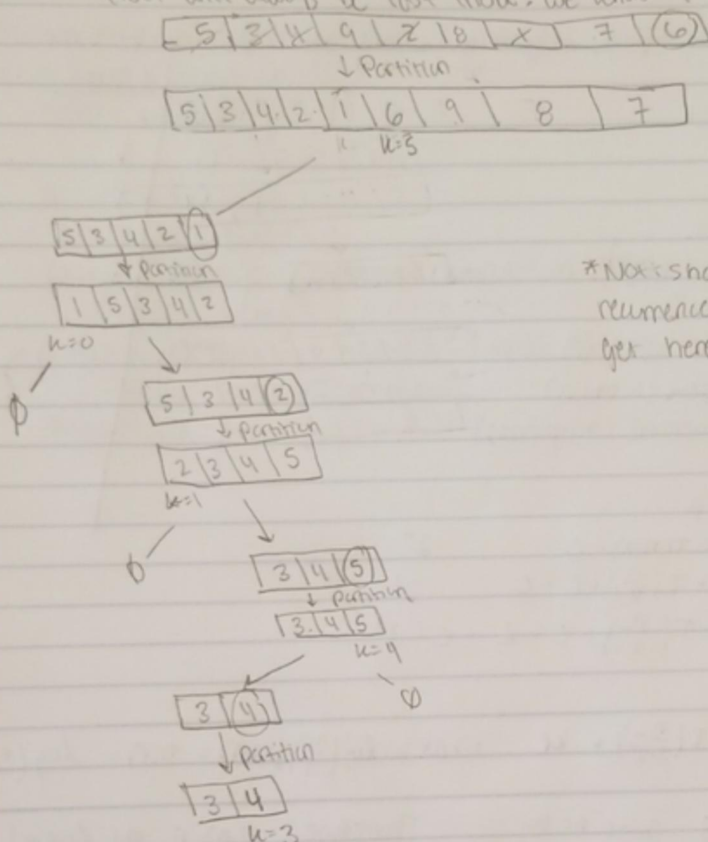
   Solution.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 8a (14 points) **Fall 2019, CU-Boulder**

2. Pivot will always be last index. We know 4th smallest will be located at k=3, stop when a pivot is placed at index=3



*Not showing right side recurrence as it will never get here

STOP, since k=3, we have found the fourth smallest element

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 8a (14 points) **Fall 2019, CU-Boulder**

3. (1 pt) Explain in 2-3 sentences the purpose of the Median of Medians algorithm.

   Solution.

   This algorithm lets us achieve an optimal pivot position at each iteration for our partition sub-routine. It returns an approximation to the median of the original array in $\Theta(n)$ time which is what we want.

4. (4 pts) Illustrate how to apply the Median of Medians algorithm (A Deterministic QuickSelect algorithm) to find the 4th largest element in the following array: `A = [6, 10, 80, 18, 20, 82, 33, 35, 0, 31, 99, 22, 56, 3, 32, 73, 85, 29, 60, 68, 99, 23, 57, 72, 25]`.

   Solution.

   We will perform the quicksort algorithm, but by using the MoM to find an approximate median for each sub array we will ensure optimal partitions. In the first iteration we will find the median of 5 different sub arrays arrays of the original array to get the approximate median.

```
    first sort each sub-array then find the median
A1 = [6, 10, 80, 18, 20] ; median = 18
A2 = [82, 33, 35, 0, 31] ; median = 33
A3 = [99, 22, 56, 3, 32] ; median = 32
A4 = [73, 85, 29, 60 , 68] ; median = 68
A5 = [94, 23, 57, 72, 25] ; median = 57

    make an array of all of the medians then find the median of that after sort
MoM = [18, 33, 32, 68, 57]

MoM = 33
```

   now we use the MoM as our pivot in the first iteration of partition subroutine. Then we carry out the quickselect algorithm until a pivot is placed in $A[n-4]$ since that will be where the 4th largest element will be. Essentially, applying this MoM algorithm will improve the overall runtime of the quickselect algorithm

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 8a (14 points) **Fall 2019, CU-Boulder**

5. (4 pts) In Tuesday's lecture, we saw how the peaked array algorithm can find the maximum element in an array with one peak. For example, $A = [15, 16, 17, 14, 12]$ is a peaked array.

   (a) (2 pts) Explain how the peaked array algorithm works in sub-linear time? (You may use the recurrence relation to help with the explanation)

   Solution.

```
Cost | Time
c1     1
c2     1
c3     1
c4     T(n/2)  , only one of these because we only recurse once
```

   If we look at the algorithm it can be easily visualed that it only considers 1/2 of each subarray at each iteration, completely disregarding other half, so it cannot be the case that this function looks at all n values, but we can see that this function runs in sub linear time if we unroll it.

   Recurrence: $T(3) = a$, $T(\frac{n}{2}) + C$

```
k=0   T(n) = T(n/2) + C
k=1   T(n) = T(n/4) + C
k=2   T(n) = T(n/8) + C
    .
    .
    .
k=j   T(n) = T(n/2*2^j) + kc

When we solve for when we hit the base case we will
get that k = log(2n/3) thus kc will equal log(2n/3)*C.
Therefore T(n) is upper bounded by O(log(n)) and
thus our algorithm is sub-linear
```

   (b) (2 pts) Re-write the peaked array algorithm to find a single valley in an array, such as $A = [56, 43, 32, 21, 23, 25, 57]$. The valley would be 21.

   Solution.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 8a (14 points) **Fall 2019, CU-Boulder**

```
findValley(a,p,r):
    mid = floor((p+r)/2)

    if p-r+1 == 3:
        return A[mid]
    else if A[mid-1] > A[mid] and A[mid+1] > A[mid]:
        return A[mid]
    else if A[mid+1] < A[mid]: # valley must be on the right side
        return findValley(a, mid, r)
    else: return findValley(a,p,mid)  # valley must be on the left side
```

Collaborated with: Zach Chommala, Bao Nguyen