

CSCI 432 - Assignment G2

Sean White, Kierstyn Brandt, Rostik Mertz, Norman Tang

September 11, 2017

Question A. Let A be an unsorted array of n integers, with $A[0] \geq A[1]$ and $A[n-2] \leq A[n-1]$. Call an index i a local minimum if $A[i]$ is less than or equal to its neighbors.

(a) How would you efficiently find a local minimum, if one exists?

Algorithm 1 FIND-LOCAL-MINIMA(A)

```
1: In:  $A$  an unsorted array of  $n$  integers, with  $A[0] \geq A[1]$  and  $A[n-2] \leq A[n-1]$ 
2: Out:  $m$  first local minima
3:
4:  $l \leftarrow \|A\|$ 
5: if  $A[0] == A[1]$  then
6:    $m \leftarrow A[0]$ 
7:   return  $m$ 
8: end if
9: if  $A[l-1] == A[l-2]$  then
10:   $m \leftarrow A[l-1]$ 
11:  return  $m$ 
12: end if
13: for  $i \leftarrow 1 \dots (n-2)$  do
14:   if  $A[i] \leq A[i-1] \&\& A[i] \leq A[i+1]$  then
15:     $m \leftarrow A[i]$ 
16:    return  $m$ 
17:   end if
18: end for
```

(b) For any loops in your algorithm, state what the loop invariant is.

The loop invariant is the fact that there exists a local minima $m \in A$

Question B. The `rand()` function in the standard C library returns a uniformly random number in $[0, RANDMAX - 1]$. Does `rand() mod n` generate a number uniformly distributed in $[0, n - 1]$?

Question C. In this class, we assume the real-RAM model of computation for our analysis. Explain why we must define what model of computation we are using in an algorithms class, or, more generally, when talking about the complexity of an algorithm.

We need a clearly defined model of computation because we need to be able to contextualize the algorithms we are analyzing. The complexity of an algorithm can change depending on the model you choose, and loops can have different implications on it overall.

Question D. Which of the following correctly capture the runtime complexity of Mergesort. (Remember, you are expected to justify your answers to questions like this).

(h) $\Theta(n \log n)$

Merge sort works in the following steps:

- 1) Select the middle index of the array: $\Theta(1)$
- 2) Split the array at the middle index: $\Theta(1)$
- 3) Sort the left array recursively: $T(\frac{n}{2})$
- 4) Sort the right array recursively: $T(\frac{n}{2})$

So we get $T(n) = \Theta(1) + \Theta(1) + T(\frac{n}{2}) + T(\frac{n}{2})$
which simplifies to $2T(\frac{n}{2}) + 2\Theta(1)$

By Master's Theorem this means that $T(n) = \Theta(n \log n)$