CSCI 432 - Assignment G2

Sean White, Kierstyn Brandt, Rostik Mertz, Norman Tang

September 13, 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)

In: A an unsorted array of n integers, with $A[0] \geq A[1]$ and $A[n-2] \leq A[n-1]$
Out: m first local minima

$l \leftarrow \|A\|$
if $A[0] == A[1]$ **then**
    $m \leftarrow A[0]$
    return m
end if
if $A[l-1] == A[l-2]$ **then**
    $m \leftarrow A[l-1]$
    return m
end if
for $i \leftarrow 1...(n-2)$ **do**
    if $A[i] \leq A[i-1] \&\& A[i] \leq A[i+1]$ **then**
        $m \leftarrow A[i]$
        return m
    end if
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 $[0, n - 1]$?

Not in all cases can we say that it is evenly distributed. In the case where $n > RANDMAX$, we will not get even distribution from $[0, n - 1]$.
This is because $rand()modn$ will always produce a number that will equal the result of that particular instance of rand().
Since the maximum value of rand() will never exceed $RANDMAX$, only the values from $[0, RANDMAX - 1]$can possibly occur.
If $n > RANDMAX$, then what of the values from $[RANDMAX, n - 1]$? They have no chance of ever occurring because under these situations $rand()modn$ cannot produce them.
If the values from $[RANDMAX, n - 1]$ have no probability of being produced, then $[0, n - 1]$ can't be said to be uniformly distributed.

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) $\Theta(nlogn)$

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 $T(n) = \Theta(nlogn)$