Valentinno Cruz

Homework Assignment 5

April. 21st, 2021

1. Algorithms

1. Develop an algorithm that takes as input a list of n integers and finds the location of the last odd integer in the list or returns 1 if there are no odd integers in the list.

- **Solution**

1. Declare int x as variable
2. Get input for x
3. algorithm takes list and size x
4. Algorithm input(list,x)
5. set index to -1
6. Declare an array of x ints
7. loop through list
8. for i from 0 to n-1
        check if number in index i is odd. yes
    end for loop
9. for i = 1 to x
        if list[i] %2 == 1
            set index = i
        end if
    end for
10. return index

2. Develop an algorithm that inserts an integer a in the appropriate position into the list $x_1, x_2, ..., x_n$ of integers that are in decreasing order.

- **Solution**

1. declare a list an array of x ints
2. for i = 1 to x
        get the input of the ith index of the list
    end for
4. declare an int z and get int of z
5. set a flag to false
        if z $\geq$ list[i]
            insert z at the position i on the list
            assign flag to true
            otherwise break loops
        end if
    end for
6. if the flag is false
        put z at last position of the list
    end if

2. Order of Complexity

1. State whether each of these functions is or is not $O(x^2)$. If you state that the function is not $O(x^2)$ then that is all you need to do. If you state that the function is $O(x^2)$ then find witnesses C and k such that $f(x) \leq Cg(x)$ where $g(x) = x^2$ when $x > k$.

(a) $f(x) = 17x + 11$

- **Solution**
  $f(x) = 17x + 11$ which is $O(x)$ this gives us $O(n^2)$
  this means it is $O(n^2)$
  which means we get C=28 and K =1
  $\therefore$ 17x+11 $\leq 28 * x^2$ for all $x > 1$

(b) $f(x) = xlog_2x$

- **Solution**
  We have $O(xlnx)$
  this means we have $O(x^2)$
  we can assign C and K to 1
  $\therefore$ xlnx $\leq 1 * x^2$ for every $x \geq 1$

(c) $f(x) = x^4/2$

- **Solution**
  This is $O(x^4)$ not $O(x^2)$
  $\therefore$ not possible

2. Give a big-O estimate (in terms of n) for the number of additions used in this segment of an algorithm

- **Solution**
  the value of i will iterate from 1 till $n = O(n)$
  so for each value of i the j value will iterate from 0 to $n = O(n)$
  $\therefore$ the complexity is $O(n) * O(n)$
  which gives us $O(n^2)$

3. Suppose you have a computer which takes 10^9 seconds to perform a bit operation. What is the largest problem, in terms of n, that this computer can solve in one second using an algorithm that requires $f(n)$ bit operations if

(a) $f(n) = log_2 n$

- **Solution**
  $f(n) = log_2 n$
  this means $log_2 n \leq 10^9$
  so $2^{log_2 n} \leq 2^{10^9}$
  $\therefore n = 2^{10^9}$

(b) $f(n) = n$

- **Solution**
  $f(n) = log_2 n$
  this means $n \leq 10^9$

(c) $f(n) = 2^n$

- **Solution**
  $f(n) = 2^n$
  which means $2^n \leq 10^9$
  so $log 2^n \leq log 10^9$
  $n log 2 \leq log 10^9$
  $n \leq 9 log 10 / log 2$
  which is $9/log 2 \simeq 30$
  $\therefore n < 30$

(d) $f(n) = n!$

- **Solution**
  $n! \leq 10^9$
  we know that $11! = 39,916,800$ which is $\leq 10^9$
  so $12! = 479,001,300$ is $\leq 10^9$
  and $13! = 627,020,800$ which is $\geq 10^9$
  $\therefore n = 12$