ESO207 (Data Structures and Algorithms): Problem Set 1

Problems

Q1. Given an array $P[0 : n-1]$ giving the daily price quotes for a stock for n consecutive days. The span of a stocks price on a given day i is the maximum number of consecutive days before i that the stocks price is less than or equal to its price on day i. This includes i-th day, so span will be at least 1. For example if $P = [50, 45, 35, 40, 60, 50, 55]$, then span is $S = [1, 1, 1, 2, 5, 1, 2]$.

(a) Give the "natural" algorithm (directly from the definition) to compute the array N. Determine the time and space complexity.

(b) Using a suitable data structure design a new algorithm for the problem which is more efficient. Determine its time and space complexity.

Q2. Given an $n \times m$ (2D) array $A[.,.]$, $A[i,j]$ is said to be a *local max* if $A[i,j] \geq A[i+a, j+b]$ for each $a \in \{-1,0,1\}$ and $b \in \{-1,0,1\}$ if $0 \leq i+a \leq n-1$ and $0 \leq j+b \leq n-1$. Design an efficient algorithm to compute all the local maxima, using divide and conquer technique.

Q3. Consider the algorithm

```
Input: A positive integer $n$
$a := 0$;
$b := 1$;
for $i := 1$ to $n - 1$ do
    $b := a + b$;
    $a := b - a$;
end
return $b$;
```

Algorithm 1: $F(n)$

(a) What does this algorithm compute? Using an invariant prove your claim.

(b) Determine the time complexity of this algorithm in terms of bit operations, i.e., determine how many bit operations are performed (not arithmetic operations) to compute $F(n)$?

Note: Normally we assume the time complexity of, say, addition of two numbers to be $O(1)$. But here we will assume that each bit operation takes $O(1)$ time. So addition of two numbers takes $O(n)$ if the larger of the two has n bits.

Q4. Can we perform the product of two complex numbers $(a + ib).(c + id)$ more efficiently than computing 4 real-number multiplications? If yes, then design such an algorithm. Assuming each real number, a, b, c, d, is an n bit integer, determine the time complexity of your algorithm in terms of the following notations. By $S(n)$ denote the number of bit operations to compute sum of two numbers where the larger has n bits, and by $M(n)$ denote the number of bit operations to multiply two numbers where the larger has n bits.

Q5. Find the time complexity of the following algorithms. Show the analysis.

(a)

```
Input: Array $A$ and integers $r, c, n$
if $r \geq n$ OR $c \geq n$ then
    return 1;
else
    if $r = n$ AND $c = n$ then
        return 0;
    else
        return $A[r, c] + \min\{Rec(A, r + 1, c), Rec(A, r, c + 1)\}$;
    end
end
```

Algorithm 2: $Rec(A, r, c)$

Hint: Find a recurrence for the number of times $Rec(r + i, c + j)$ will be invoked.

(b) Suppose *Store* is a global array containing -1 at all locations

```
Input: Array $A$ containing positive entries at all locations and integers
       $r, c, n$
if ($r \geq n$ OR $c \geq n$) then
    return 1;
else
    if ($r = n$ AND $c = n$) then
        return 0;
    else
        if $S[i, j] = -1$ then
            return $A[r, c] + \min\{Majic(A, r+1, c), Majic(A, r, c+1)\}$ ;
        end
    end
end
```

Algorithm 3: $Majic(A, r, c)$