# **Lab 3**

**Question 1:**

**Algorithm** beautiful (A,n)

**Input**: An integer array with n elements

**Output**: Sum of the elements

sum ß 0

for iß 0 **to** A.length-1 **do**

sum ß sum + A[i]

**return** sum

This algorithm runs in O(n) and it’s the best-case running time as well as worst case running time.

**Question 2:**

2^n, 2^(n+1), 2^(2n), 2^(2^n)

**Question 3:**

O(1) – Push and pop operations in a stack

O(n) – Search for an element in an array

O(logn) – Binary search

O(nlogn) – Quick sort

O(n^2) – Bubble sort

O(n^3) – Matrix multiplication

O(2^n) – Subset problem

**Question 4:**

**Algorithm** fib(n)

**Input**: natural number n

**Output**: F(n) – the nth Fibonacci number

If(n=0||n=1) then return n

Return fib(n-1) + fib(n-2)

T(n) = T(n-1) + T(n-2) + c

Master’s formula doesn’t apply in this case because T(n) does not satisfy the conditions of the master’s formula.

**Question 5.**

T(1) = 1 T(n) = 2T(n/2) + c

Assume n = 2k

T(bk) = aT(bk-1) + c(bk)m

aT(bk-1) = a2T(bk-2) + ac(bk-1)m

aT(bk-2) = a3T(bk-3) + a2c(bk-2)m

…

ak-2T(b2) = ak-1T(b) + ak-2c(b2)m

ak-1T(b) = akT(1) + ak-1c(b)m

T(n) = c(bk)m+ ac(bk-1)m + a2c(bk-2)m + . . . + ak-2c(b2)m + ak-1c(b)m + akT(1)

T(n) = c[(bk)m+ a(bk-1)m + a2(bk-2)m + . . . + ak-2(b2)m + ak-1(b)m] + akT(1)

a = 2, b = 2, k = 0. a > b0

T(n) = c[1+2+22 + 23 + . . . + 2k-2 + 2k-1] + 2k

T(n) = 2k = O(n)

**Question 6.**

**Example one:**

Merge sort algorithm:

**Algorithm** mergeSort(S)

**Input** sequence S with n

**Output** sequence S sorted

if S.size() > 1 then

(S1 , S2 ) ¬ partition(S, n/2)

mergeSort(S1 )

mergeSort(S2 )

S ¬ merge(S1 , S2 )

return S

T(1) = d

T(n) = 2T(n/2) + cn

a=2

b=2

k=1

a=2=2^1=b^k

T(n) = O(n^klogn) = O(nlogn)

**Example two:**

Find max algorithm

**Algorithm** findMax(A, start, end)

If lower = upper return A[lower]

If lower+1 = upper return max (A[lower], A[upper])

return max (findMax(A, lower, (lower+upper)/2),

findMax(A, ((lower+upper)/2)+1, upper))

T(1) = d

T(n) = 2T(n/2) + c

a=2

b=2

k=0

T(n) = O(nlogb(a)) = O(n)

**Example 3:**

**Binary search algorithm**

**Algorithm** binSearch(A, x, lower, upper)

**Input**: Already sorted array A of size n, value x to be searched for in array section A[lower]..A[upper]

**Output**: true or false

if lower > upper then return false

mid ¬ (upper + lower)/2

if x = A[mid] then return true

if x < A[mid] then

return binSearch(A, x, lower, mid – 1)

else

return binSearch(A, x, mid + 1, upper)

T(1) = d

T(n) = T(n/2) + c

a=1

b=2

k=0

T(n) = O(logn)