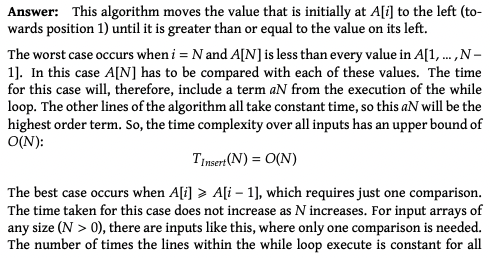
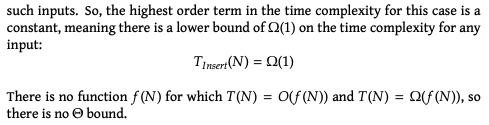
**2018-2019 Algos Exam**

\*Some proposed solutions may have errors, please comment / correct if so

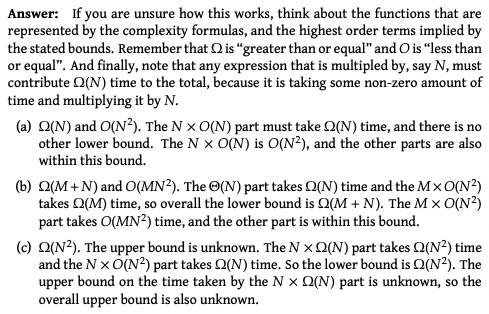
**Question 1**

1. Tutorial 1 Q1





1. i, ii, and iii) Tutorial 1 Q2



1. i) By the formal definition of , we know that there exists c1,c2 such that for all . As a result, we cannot choose a c3 such that for (formal definition of O(N)).

ii) It is unclear which one will take longer. By formal definitions, we know that and for . There can be some cases where if c3>N then the upper bound of Y would be greater than X and it may take longer for it to run. E.g. if we are working with an array of length 3 and Y contains 4 for loops.

**Question 2**

a)

**Procedure SimplifiedMaxSum(A, L, M, R)**

maxsofar = 0

maxdirection = 0

For i=M+1 to R:

maxdirection += A[i]

if maxdirection > maxsofar

maxsofar = maxdirection

endif

endfor

maxdirection = 0

For i=M-1 to L:

maxdirection += A[i]

if maxdirection > maxsofar

maxsofar = maxdirection

endif

endfor

maxsofar += A[M] + A[M-1]

Return maxsofar

Endprocedure

b) i)

**Procedure MaxSubSum(A, L, R)**

If R =L

return A[L]

Endif

M = (R+L)/2

Return max(MaxSubSum(A,L,M), MaxSubSum(A,M+L,R), SimplifiedMaxSum(A,L,M,R))

Endprocedure

ii) \*best guess

T(N) = 2T(N/2) + f(N), f(N) is the call to function simplified max sum

iii) \*best guess

Master method a=2, b=2

By case 2 of the Master Method,

**Question 3**

1. i)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **J** | **M** | **F** | **O** | **E** | **D** | **V** | **N** | **K** | **A** | **H** |
| F | J | J | E | O | V | D | D | J | J | A |
| M | F | M | M |  | N | N | V | A | K | V |
| K | O | N |  |  |  | H | F |  | H |  |
| A |  |  |  |  |  | A |  |  | V |  |
|  |  |  |  |  |  |  |  |  |  |  |

ii)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Distance=0 | D=1 | D=2 | D=3 | D=4 |
|  |  |  |  |  |
|  |  | A |  |  |
|  | V | H |  | O |
| D |  |  | J |  |
|  | N | F | M | O |
|  |  |  | N |  |
|  |  |  |  |  |
|  |  |  |  |  |

iii) Depth first travels to the end of a path (reaching max distance from source) before returning to previously unsearched vertices and travelling to end of other paths. Thus, graphs cannot look the same as Breadth first travels to d=1 from source, then d=2, then etc...

1. i)

Procedure Splitcoin(C)

Coins[C]

For i=0 to C+1:

Coins[i]=i

Value =

If Value > Coins[i]

Coins[i] = Value

Endif

Value = 0

Endfor

Return Coins[C]

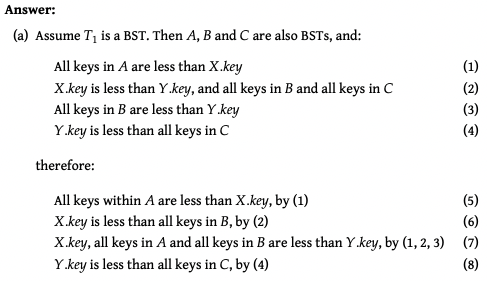
Endprocedure

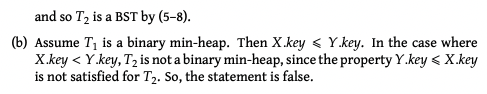
ii) Time complexity in all cases is . Has to solve all subproblems regardless of input.

iii) Optimal substructure is if the optimal solution can be constructed from the solution of its subproblems. Overlapping subproblems are where the algo solves the same subproblem over and over again. In this case yes it does apply as if written recursively, there could be situations where the algo solves the same subproblem repeatedly (e.g. if coin = 80, then it could call splitcoin again with c=40, 26, 20, and c=40 would in turn call c=20 again thus it would be solving c=20 twice).

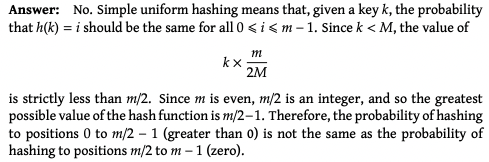
**Question 4**

1. Tutorial 6 Q1





1. i) Tutorial 6 Q2



ii) \*best guess  
In the worst case, k would have to be compared to every other key (N keys). Thus the time complexity would be O(N).

In the expected case, the probability that k is mapped to position i is 2/m within the range 0 to m/2 and 0 within the range m/2 to m. Thus,

iii) \*best guess

Yes if all keys mapped to a different space in the table then there would be no collisions and thus the time complexity for searching would be theta(1).

iv) This occurs when m = 2N.