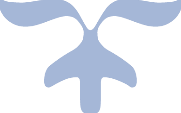


Program analysıs assıgnment 1

Candidate No: 181509



October 21, 2018

Unıversıty of sussex

Q1)

f1 = O(n3), f2 = O(n), f3 = O(log(n2)), f4 = O(3n), f5 = O(2n), f6 = O(n2)

Ascending order:

O(log(n2)), O(n), O(n2), O(n3), O(2n), O(3n) : f3, f2, f6, f1, f5 ,f4

The running time functions belong to the function sets above and arranged according to them.

Q2)

1. Θ(n) ; Regardless the input content, algorithm will always terminate after iterating the whole input.
2. Θ(n) ; The algorithm chooses the limit of for loop as the minimum value between n and m. If n and m are equal, then the algorithm will run for n times. If they are not equal the algorithm will run for n or m times, choosing the one that is lesser.
3. Θ(n3) ; All three of the for loops will run until n without break, giving the n3 runtime.
4. Ω(1) ; If ai < 0 then the algorithm will terminate without going through the while loop.

O(n!) ; If n is equal to m, we get the worst runtime with factorial as n\*n!.

1. a = even

b = odd

a = 2\*k

b = 2\*c+1

a + b = 2\*k + 2\*c + 1 = 2(k + c) + 1

L = k + c

P = a + b

a + b = 2\*L + 1

P = 2\*L + 1

Any odd number by definition summed with any even number by definition, gives an odd number definition.

Q3)

1. If the message is made from a key that is n binary bits, that means a message of length 1 can be constructed in n different ways. Therefore, the runtime of a brute force method in the worst case would be O(an), as each character can be repeated. The best case is Ω(1) if the message length is 1 and the key length is also 1.
2. It should concern because of the fact how fast it can change from simple to complex. If the key is not long enough or the message, the encryption can be weak.
3. 30 days = 43200 minutes. If 1 key is applied in 1 minute, that means 43200 keys are applied in 30 days. If the completion chance is not even 1%, then that means 43200 keys are less than 1% of the possible number of keys. If 43200 keys were the 1%, we could assume the 100% as 4320000 keys at lowest. With the binary system, that means 2n >= 4320000, n >= log2 4320000, n >= 22.043. Thus, n should be 23 as a minimum value.
4. If 0.5% chance in 30 days, then it is 6000 days for 100% of the keys to be produced. 6000 days = 8640000 minutes. Now time to apply a key has changed to t = 0.5 \* n in seconds, which makes 60/t to find per minute. This leads to 1036800000/n <= 2n. Thus, if n is 26, then we get 39876923 <= 67108864 which is valid with n’s minimum value as 26.

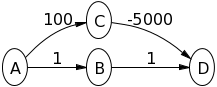
Q4)



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 |
| iteration | **0** | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| 1 | **0** | 5 | ∞ | ∞ | ∞ | ∞ | **2** | ∞ |
| 2 | **0** | 5 | ∞ | **3** | ∞ | ∞ | **2** | 6 |
| 3 | **0** | **5** | ∞ | **3** | 6 | ∞ | **2** | 6 |
| 4 | **0** | **5** | 7 | **3** | **6** | ∞ | **2** | 6 |
| 5 | **0** | **5** | 7 | **3** | **6** | 10 | **2** | **4** |
| 6 | **0** | **5** | 7 | **3** | **6** | **5** | **2** | **4** |
| 7 | **0** | **5** | **7** | **3** | **6** | **5** | **2** | **4** |
| 8 | **0** | **5** | **7** | **3** | **6** | **5** | **2** | **4** |

I think the answer is incorrect because of the negatively weighted edge. The cheapest way to get to V8 is 4, following V1, V7, V4, V5, V8 and to V5 is 4 as well, following V1, V7, V8, V5. Since Dijkstra’s algorithm does not reconsider the vertices it visited before, it cannot see vertex it found first which it also uses to reach to edge -2, making it able to only use the -2 once between V5 and V8 instead of both ways.

1. Multiple negatively weighted edges on a graph could cause inconsistencies when given to a Dijkstra’s algorithm. The reason for this is because the algorithm chooses the next vertex by comparing the edges and finding the path with the cheapest cost. However, the algorithm always finds the cheapest path for a new vertex and closes the paths to that vertex, only looking forward, never considering the past choices again.



In such a graph, the algorithm will first find the path to D through A and B with a total cost of 2. As it makes its way from A to C, it will find -5000 to D. In this case, (A, C, D) should be the cheapest path from A to D but since Dijkstra’s algorithm does not override paths, the output will not be the desired and stay as cost 2 from A to D. There could be graphs with negatively weighted edges which still work without a problem when given to Dijkstra’s algorithm. An example for such cases can be a graph which has a negative edge next to starting vertex.