|  |  |  |
| --- | --- | --- |
| BUE final logo  Informatics and Computer Science | **16CSCI01I**  **In - Class Test**  **2016 / 2017**  **Version 1** | |
| Module Title: **Analysis of Algorithms** | | |
| Module Leader **Dr. Mostafa Salama** | | Semester  **One** |
| Assessment Weight  **30% of the total course mark** | Due Date  **Announced on E-Learning** | |

|  |  |
| --- | --- |
| **Student ID:** |  |
| **Student Name:** |  |
| **Group No.** |  |

**Instructions to Students**

* The exam paper is **2** pages long, and is in **1** sections
* You should attempt **all** questions
* The allocation of marks is shown in brackets by the questions
* This examination is **1** hour long

---------------------------------------------------------

**Exam Instructions:**

* You should write down your answers within the test paper in the empty blanks below each question.

The course work is assessed as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Point number | Grade | Student grade |  |
| 1 | 8 marks |  |  |
| 2 | 6 marks |  |  |
| 3 | 6 marks |  |  |
| 4 | 10 marks |  | TA Signature |
| Total | 30 marks |  |  |

Feedback: The marks will be announced on the E-Learning with the model answer of the test.

1. Consider the following Brute-force algorithm that finds a match between a pattern P of length m in a text T of a length n such that m is smaller than n. [8 marks, 2 for each point]

**ALGORITHM** *BruteForceStringMatch(T* [0*..n* − 1]*, P*[0*..m* − 1]*)*

//Input: An array *T* [0*..n* − 1] of *n* characters representing a text and

// an array *P*[0*..m* − 1] of *m* characters representing a pattern

//Output: The index of the first character in the text that starts a

// matching substring or −1 if the search is unsuccessful

**for** *i* ←0 **to** *n* − *m* **do**

*j* ←0

**while** *j <m***and** *P*[*j* ]= *T* [*i* + *j* ] **do**

*j* ←*j* + 1

**if** *j* = *m* **return** *i*

**return** −1

If the average case scenario is when:

*T* = “N O B O D Y \_ N O T I C E D \_ H I M”

P = “**N** O T”

1. What is the best case scenario of this algorithm, give an example?
2. Show how to compute the complexity of the best case scenario.
3. What is the worst case scenario of this algorithm, give an example?
4. Show how to compute the complexity of the worst case scenario?
5. Set up and solve a recurrence relation for the number of times the following algorithm’s basic operation is executed. It computes the sum of the first n cubes: [6 marks]

S(n) = 13 + 23 + ... + n3.

Algorithm S(n)

//Input: A positive integer n

//Output: The sum of the first n cubes

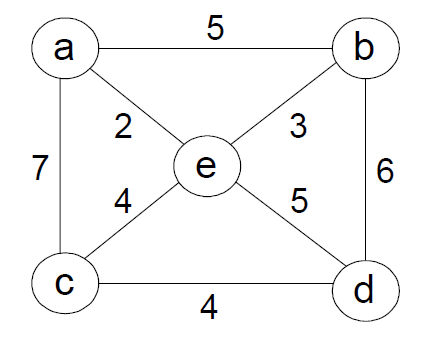
if n = 1 return 1

else return S(n − 1) + n ∗ n ∗ n

1. State the Master Theorem then use it to find the O notation for the following recurrence relations. [6 marks]

*T*(*n*) = 4*T*(*n*/2) + *n*2

1. Use Prim algorithm to find the minimum spanning tree of the following graph, clarify the steps of the algorithm in your solution



***Good Luck***

***Answer here:***

…………………………….…………………………………….…………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………………………….…………………………………….……………………………………………….…………………………………….………………………………………

|  |  |  |
| --- | --- | --- |
| BUE final logo  Informatics and Computer Science | **16CSCI01I**  **In - Class Test**  **2016 / 2017**  **Model Answer**  **Version 1** | |
| Module Title: **Analysis of Algorithms** | | |
| Module Leader **Dr. Mostafa Salama** | | Semester  **One** |
| Assessment Weight  **30% of the total course mark** | Due Date  **Announced on E-Learning** | |

1. Consider the following Brute-force algorithm that find a match between a pattern P of length m in a text T of a length n such that m is smaller than n? [8 marks, 2 for each point]

**ALGORITHM** *BruteForceStringMatch(T* [0*..n* − 1]*, P*[0*..m* − 1]*)*

//Input: An array *T* [0*..n* − 1] of *n* characters representing a text and

// an array *P*[0*..m* − 1] of *m* characters representing a pattern

//Output: The index of the first character in the text that starts a

// matching substring or −1 if the search is unsuccessful

**for** *i* ←0 **to** *n* − *m* **do**

*j* ←0

**while** *j <m***and** *P*[*j* ]= *T* [*i* + *j* ] **do**

*j* ←*j* + 1

**if** *j* = *m* **return** *i*

**return** −1

If the average case scenario is when:

*T* = “N O B O D Y \_ N O T I C E D \_ H I M”

P = “**N** O T”

1. What is the best case scenario of this algorithm, give an example?
2. Show how to compute the complexity of the best case scenario.
3. What is the worst case scenario of this algorithm, give an example?
4. Show how to compute the complexity of the worst case scenario?
5. **Model Answer**:
   1. The best case scenario of this algorithm is when the pattern P is “N O B”
   2. The complexity of best case is O(1)
   3. The worst case scenario of this algorithm is when the algorithm may have to make all m comparisons before shifting the pattern, and this can happen for each of the n − m + 1 tries. This is when T = “M M M M M M M M M M M M M M M M M M M” and P=”M M T”
   4. The complexity of the worst case scenario that make the algorithm makes m(n − m + 1) character comparisons, which puts it in theO(nm) class.
6. Set up and solve a recurrence relation for the number of times the following algorithm’s basic operation is executed. It computes the sum of the first n cubes: [6 marks]

S(n) = 13 + 23 + ... + n3.

Algorithm S(n)

//Input: A positive integer n

//Output: The sum of the first n cubes

if n = 1 return 1

else return S(n − 1) + n ∗ n ∗ n

**Model Answer:**

The recurrence relation

C(n) = C(n − 1) + 1,

C(0) = 1 (there is a call but no multiplications when n = 0).

The solution of the recurrence

C(n) = C(n − 1) + 1

= [C(n − 2) + 1] + 1 = C(n − 2) + 2 = ...

= C(n − i) + i = ...

= C(0) + n

= 1+n.

1. State the Master Theorem then use it to find the O notation for the following recurrence relations. [6 marks]

*T*(*n*) = 4*T*(*n*/2) + *n*2

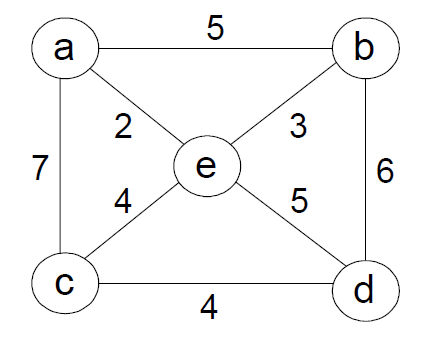
**Model Answer**:

f(n) = a f(n/b) + cnd

*a =* 4, *b* = 2, d=2 ⇒ a = 4 ? bd = 22=4 🡺 equal

Case 2: T(n) = O(n2 log2 n)

1. Use Prim algorithm to find the minimum spanning tree of the following graph, clarify the steps of the algorithm in your solution. [10 marks]



**Model Answer**

Prim Algorithm

*T* = ϕ; //Set of selected edges

*S =* ϕ; //Set of selected vertexes in MST

For every vertex *v* in *V*; key(*v*) = ∞; initial key value ∞

Pick random vertex *a* in *V*; key(*a*)= 0; initial key value 0

while (*S*≠*V*){

Get vertex *u* of minimum key value in *V*, such that *u*∉*S*

*T* = *T* ⋃ {(*u*, *a*)}, such that *a*∈*S* && weight(*u-a*)=key(*u)*

*S* = *S* ⋃{*u*}, and remove *u* from *V*

For every vertex *v* connected to *u* such that *v*∉*S*

If weight(*u*-*v*) weight is less than key(*v*)

Then key(*v*) = weight(*u*-*v*)

Tree vertices Priority queue of remaining vertices

a(-,-) b(a,5) c(a,7) d(a,∞) e(a,2)

e(a,2) b(e,3) c(e,4) d(e,5)

b(e,3) c(e,4) d(e,5)

c(e,4) d(c,4)

d(c,4)

The minimum spanning tree found by the algorithm comprises the edges

ae, eb, ec, and cd.