|  |  |
| --- | --- |
|  | **BAHRIA UNIVERSITY,**  **(Karachi Campus)**  *Department of Software Engineering*  **Quiz #01– Fall 2020**  **Solution** |

Sol

|  |  |  |  |
| --- | --- | --- | --- |
| COURSE TITLE: | **D&AA** | COURSE CODE: | **CSC-321** |
| Class: | **BSE 7A&B** | Shift: | **Morning** |
| Course Instructor: | **ENGR. BUSHRA FAZAL KHAN** | Time Allowed: | **5 Hr.** |
| Date: | **7-Nov-2020** | Max. Marks: | **10 Points** |

1. Consider the following version of an important algorithm (2.5)

**ALGORITHM** *GE(A*[0*..n* − 1*,* 0*..n*]*)*

//Input: An *n* × *(n* + 1*)* matrix *A*[0*..n* − 1*,* 0*..n*] of real numbers

**for** *i* ←0 **to** *n* − 2 **do**

**for** *j* ←*i* + 1 **to** *n* − 1 **do**

**for** *k*←*i* **to** *n* **do**

*A*[*j, k*]←*A*[*j, k*]− *A*[*i, k*] ∗ *A*[*j, i*]*/ A*[*i, i*]

# What is its basic operation?

Basic operation is Multiplication

# How many times is the basic operation executed?

# Therefore basic operation is executed

# What is the efficiency class of this algorithm?

Its efficiency class is cubic.

1. Solve the following recurrence relations using Master Theorem. (2.5)

# a. x(n) = 9x(n/3) + 5 for n > 1, x(1) = 0

a=9 b=3 f(n)=5 log3 9 = 2

f(n) = 5 = Θ(1) = 0 => Θ(1) = Θ(n0)

n2 > n0

x(n) = Θ(n2)

# b. x(n) = x(n/2) + n for n > 1, x(1) = 1

a=1 b=2 f(n)=n log2 1 = 0

n0 = 1

Given that n > 1 1 < n

x(n) = Θ(n)

# c. x(n) = x(n/3) + 1 for n > 1, x(1) = 1

a=1 b=3 f(n)=1 log31 = 0

n0 = 1

1 = f(n) => 1 = 1

O(nlogba logk+1n) O(n0log0+1n) => O(logn) x(n) = Θ(log2n)

1. Consider the following recursive algorithm. (5)

**ALGORITHM** *Q(n)*

//Input: A positive integer *n*

**if** *n* = 1 **return** 1

**else return** *Q(n* − 1*)* + 2 ∗ *n* – 1

# Draw a tree of recursive calls for this algorithm and compute its time complexity.

* 1. **Compute complexity using induction method**

T(n)  b

T(n)  T(n-1) + c for n>1

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

= T(n-2) + c T(n-2) = T((n-2)-1) + c T(n-2) = T(n-3) + c

Substituting T(n-1)

 T(n-2) + c + c

 T(n-2) + 2c

Substituting T(n-2)

 T(n-3) + 2c + c

 T(n-3) + 3c

 T(n-k) + kc

In order to reach base case n-k=1

k=n-1

Substituting k=

 T(n-(n-1)) + (n-1)c

 T(n-n-1) + (n-1)c

 T(1) + (n-1)c

 b+ c(n-1) = Θ(n)