|  |
| --- |
| **Part A** |
| **Aim:**   1. Design algorithms for following problems.   Euclid Algorithm, Matrix Multiplication, Matrix addition   1. Find time and space complexity of algorithms. 2. Implement the algorithms in any programming language. |
| **Prerequisite:** Any programming language |
| **Outcome:** Algorithms and their implementation |
| **Theory:** |
| **Procedure:**   1. Design algorithm and find complexity 2. Implement algorithm in any programming language. 3. Paste output   **Practice Exercise:**   |  |  | | --- | --- | | S.no | Query statement | | 1 | Design and analysis Euclid algorithm for finding GCD of two numbers. | | 2 | Design and analysis algorithm for matrix addition. | | 3 | Design and analysis algorithm for matrix multiplication. | |
| **Instructions:**   1. Design, analysis and implement the algorithms. 2. Paste the snapshot of the output in input & output section. |
| **Part B** |
| **1).**  Design and analysis Euclid algorithm for finding GCD of two numbers. |
| **Algorithm:** Euclid Algorithm for finding GCD of 2 numbers.  **Input:** two Integer values n1 and n2.  **Output:** One Integer Value.  Start:  Gcd(n1,n2):  While(n2 != 0):  Rem = n1%n2  n1 = n2  n2 = rem  return n1  End. |

|  |
| --- |
| **Code:** |
| **Input & Output:** |
| **Time Analaysis:**  In every iteration the reduction in sum (a+b) at minimum is 1.5.  so on solving a+b/(3/2)k =1  We can get the value of k as O(log 3/2 (a+b)) .  Time complexity of this algorithm will be in O(log 3/2 (a+b)) .  **Space Analysis:**  No of variables used is 3 (rem,a,b) so the space complexity will be 1+1+1 = 3 that is O(1). |
| **2).** Design and analysis algorithm for matrix addition. |
| **Algorithm:** Algorithm for Matrix addition.  **Input**: No of rows and columns of 2 matrices and their elements.  **Output:** 1 Matrix of the order of the input matrices.  START:  Addition(a[],b[]):  If order(a) equals to order(b):  for(i=0;i<r;i++)  for(j=0;j<c;j++)  print(a[i][j]+b[i][j])  return;  End. |
| **Code:** |

|  |
| --- |
| **Input&Output:** |
| **Time Analysis:**  In this algorithm there are 2 for loops , each for loop requires time complexity of O(r),O(c).  Since the both loops are nested it becomes O(r\*c). if r = c i.e, for an square matrix the complexity will be O(n^2).  **Space Analysis:**  In this algorithm No of variables used is 4(i,j,r,c) and no of data structures used are 2(A,B) of size r\*c , r\*c . So, the space complexity will be 2\*(r\*c)+4 that is O(r\*c) if r = c i.e, for an square matrix space complexity will be O(n^2). |
| **3).** Design and analysis algorithmm for matrix multiplication. |
| **Algorithm:** Algorithm for Matrix Multiplication.  Input: No of rows and columns of 2 matrices and their elements.  **Output:** 1 Matrix of the order of the input matrices.  START:  Addition(a[],b[]):  If order(a) equals to order(b):  for(i=0;i<r;i++)  for(j=0;j<c;j++)  m[i][j] = 0  for(k=0;k<c;k++)  m[i][j] = a[i][k] + b[k][j]  print(m[i][j])  return;  End. |

|  |
| --- |
| **Code:** |
| **Input&Output:** |
| **Time Analysis:**  In this algorithm there are 3 for loops , each for loop requires time complexity of O(r),O(c),O(c).  Since the both loops are nested it becomes O(r\*c\*c). if r = c i.e, for an square matrix the complexity will be O(n^3).  **Space Analysis:**  In this algorithm No of variables used is 5(i,j,k,r,c) and no of data structures used are 3(A,B,M) of size r\*c , r\*c , r\*c . So, the space complexity will be 3\*(r\*c)+5 that is O(r\*c) if r = c i.e, for an square matrix space complexity will be O(n^2). |
| **Observation & Learning:**  I observed that using loops will increase the time complexity abruptly. And to write a good algorithm with reasonable time complexity we need to think about reducing the usage of loops |
| **Conclusion:**  I understood and learnt calculating time complexity for Euclid’s algorithm, Matrix addition and multiplication algorithm. |

|  |
| --- |
| **Questions:**   1. What is the goodness criteria’s for algorithms? 2. Precision: a good algorithm must have a certain outlined steps. The steps should be exact enough, and not varying.   Uniqueness: each step taken in the algorithm should give a definite result as stated by the writer of the algorithm. The results should not fluctuate by any means.  Feasibility: the algorithm should be possible and practicable in real life. It should not be abstract or imaginary.  Input: a good algorithm must be able to accept a set of defined input.  Output: a good algorithm should be able to produce results as output, preferably solutions.  Finiteness: the algorithm should have a stop after a certain number of instructions.  Generality: the algorithm must apply to a set of defined inputs.   1. How will two algorithms that solve the same problem be compared?   A). Two algorithms that solve the same problem be compared by less time complexity and . space complexity. |