Data Structure and Algorithm

Assignment

**Q.1 Write an algorithm for calculating the factorial of a number.**

1. Read n
2. Initialize fact =1, i=2
3. While i<n repeat step 4 and 5
4. Fact=fact\*i
5. i++
6. Display fact

**Q.2 write an algorithm that tells the nature of the roots of equation.**

1. Read a,b,c
2. Initialize condition=b2-4ac
3. If condition > 0 then, roots are real and unequal
4. Else if condtion < 0 then, roots are unreal
5. Else if condition == 0 , then roots are real and equal

**Q.3 Write an algorithm for finding the maximum number in an array also discuss its time complexity in best, worst and average case.**

1. Read arr // array
2. Initialize len,i=0 // len is the length of array
3. Max = arr[0]
4. If len ==1 ,return max
5. Repeat steps 5,6,7 while i < arr
6. If max > arr[i] then goto step 6
7. Max = arr[i]
8. i++
9. Display max

**Time Complexity:**

**SORTED ARRAY:**

|  |  |  |
| --- | --- | --- |
| Best | Average | Worst |
| **Item is present** | | |
| 1 | 1 | 1 |
| **Item is absent** | | |
| 1 | 1 | 1 |

**UNSORTED ARRAY:**

|  |  |  |
| --- | --- | --- |
| Best | Average | Worst |
| **Item is present** | | |
| 1 | N/2 | N |
| **Item is absent** | | |
| N | N | N |

**Q.4 Write an algorithm that finds the maximum number of three given numbers by the user.**

1. Read a,b,c
2. if a>b
3. if a>c , return a is the largest
4. else, return c is the largest
5. if b>c, return b is the largest

**Q.5 write an algorithm that determines whether a number is prime.**

1. Read n
2. Initialize i=2, flag = false
3. While i<n/2 repeat steps 4,5,6
4. If(i%2 == 0)
5. Flag = true, break
6. i++
7. if flag = false , return n is prime
8. else, return n is not prime

**Q.6 Write an algorithm that finds a number from array of size N (no repetition), also discuss time complexity.**

1. Read n
2. Initialize i=0, check = false, N(size of array)
3. While i<N repeat steps 4,5,6,7
4. If arr[i] == n goto step 5
5. Check = True
6. Else check = false goto step 7
7. i++

**Time Complexity:**

Sorted Array:

|  |  |  |
| --- | --- | --- |
| Best | Average | Worst |
| **Item is present** | | |
| 1 | N/2 | N |
| **Item is absent** | | |
| 1 | N/2 | N |

Unsorted Array:

|  |  |  |
| --- | --- | --- |
| Best | Average | Worst |
| **Item is present** | | |
| 1 | N/2 | N |
| **Item is absent** | | |
| N | N | N |

**Q7. Write an algorithm for generating Fibonacci series.**

1. Initialize sum , N=5
2. f0=0,f1=1
3. display f0,f1
4. while I <= 5 repeat steps 4,5,6,7
5. sum = f1+f0
6. f0 = f1
7. f1 = sum
8. i++
9. display sum

**Q8. Write an algorithm for finding an element in an array using binary search.**

1. Read target, arr[N] // N is size of sorted array
2. Set max = N-1
3. Set min = 0
4. While max >= min repeat steps5,6,7,8,9
5. mid = (max+min)/2
6. if target == arr[mid], return target
7. else if target > arr[mid]
8. max = mid-1
9. else min = mid+1
10. return element not found

**Time Complexity:**

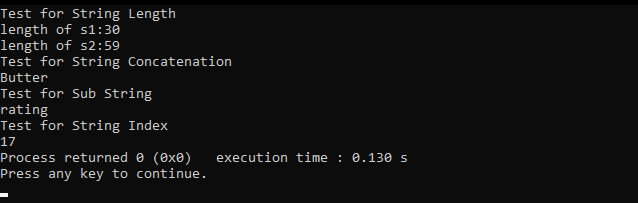
Sorted Array:

|  |  |  |
| --- | --- | --- |
| Best | Average | Worst |
| **Item is present** | | |
| 1 | Log2(N) | Log2(N) |
| **Item is absent** | | |
| Log2(N) | Log2(N) | Log2(N) |

return 0;

}

OUTPUT:



**Q. Design an algorithm that takes a string S and a number n as input. It then concatenates S repeatedly to itself until the length of the resultant string becomes greater than or equal to n.**

**Concatenate(s,n)** // s is string and n is length of string

1. Initialize x=0
2. Repeat steps 3 and 4 while x<=n
3. s.append(s)
4. x=size of string
5. add null character at the of the string

**Q. Design an algorithm that takes a string S and a char a as input. It then displays the indices of all occurrences of a in S.**

**Find(S,a)**

1. Initialize i=0
2. Repeat steps 3 and 4 until null character
3. if s[i] == a
4. Display i