Design and Analysis Algorithms lab: Assignment-1

119CS0550

SUBRAT KUMAR PRADHAN

**Q1. 1. Design an algorithm to find the maximum element in a list of n unique numbers.**

**Give a trace for get\_MAX(A,n), where A={1,5,2,9,4,3} and n=6.**

**Design a recurrence relation for the algorithm and find its time complexity.**

**Give a proof of correctness.**

1.2 **Algorithm/pseudo code:**

Given an array A of n unique numbers.

1. get\_MAX(A,n)
2. if(n isEqual 1)
3. return A[0]
4. Else return maximum of(A[n-1],get\_MAX(A,n-1))

1.3 **Recurrence relation:**

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

=O(1)+O(1)+T(n-2)……..

=………….

=O(n)

**Hence the time complexity of the algorithim is O(n).**

**1.4 Tracing:**

enter the sizeof the array

**6**

**enter the elements of the array**

**1 5 2 9 4 3**

**\*\*\*\*\* trace of getting maximum elements of the array**

**\*\*\*\*\*\*\*\*\*called Main function, main(): Activation Record1**

**Variable Address Value**

**n 0x61fdd4 6**

**f 0x61fdd0 15341312**

**activation\_record\_id 0x404014 1**

**///\*\*\*Calling find\_max\_element function(a,6):Activation record 1**

**Variable value address**

**n 6 0x61fd98**

**activation\_record\_id 1 0x404010**

**///\*\*\*Calling find\_max\_element function(a,5):Activation record 2**

**Variable value address**

**n 5 0x61fd58**

**activation\_record\_id 2 0x404010**

**///\*\*\*Calling find\_max\_element function(a,4):Activation record 3**

**Variable value address**

**n 4 0x61fd18**

**activation\_record\_id 3 0x404010**

**///\*\*\*Calling find\_max\_element function(a,3):Activation record 4**

**Variable value address**

**n 3 0x61fcd8**

**activation\_record\_id 4 0x404010**

**///\*\*\*Calling find\_max\_element function(a,2):Activation record 5**

**Variable value address**

**n 2 0x61fc98**

**activation\_record\_id 5 0x404010**

**///\*\*\*Calling find\_max\_element function(a,1):Activation record 6**

**Variable value address**

**n 1 0x61fc58**

**activation\_record\_id 6 0x404010**

**\*\*\*\*\*\*\*\*\*Returning from find\_max\_element(a,1): Activation Record6**

**Variable Address Value**

**n 0x61fc58 1**

**result 0x61fc38 1**

**activation\_record\_id 0x404010 6**

**\*\*\*\*\*\*\*\*\*Returning from find\_max\_element(a,2): Activation Record5**

**Variable Address Value**

**n 0x61fc98 2**

**result 0x61fc78 5**

**activation\_record\_id 0x404010 5**

**\*\*\*\*\*\*\*\*\*Returning from find\_max\_element(a,3): Activation Record4**

**Variable Address Value**

**n 0x61fcd8 3**

**result 0x61fcb8 5**

**activation\_record\_id 0x404010 4**

**\*\*\*\*\*\*\*\*\*Returning from find\_max\_element(a,4): Activation Record3**

**Variable Address Value**

**n 0x61fd18 4**

**result 0x61fcf8 9**

**activation\_record\_id 0x404010 3**

**\*\*\*\*\*\*\*\*\*Returning from find\_max\_element(a,5): Activation Record2**

**Variable Address Value**

**n 0x61fd58 5**

**result 0x61fd38 9**

**activation\_record\_id 0x404010 2**

**\*\*\*\*\*\*\*\*\*Returning from find\_max\_element(a,6): Activation Record1**

**Variable Address Value**

**n 0x61fd98 6**

**result 0x61fd78 9**

**activation\_record\_id 0x404010 1**

**\*\*\*\*\*\*\*\*\*Returning from Main function, main(): Activation Record1**

**Variable Address Value**

**n 0x61fdd4 6**

**f 0x61fdd0 9**

**activation\_record\_id 0x404014 1**

**Final result:the maximum element is 9**

1.5 **proof of correctness of the algorithim:**

We can prove the correctness of the algorithm by inductive

p(n): The value returned by get\_MAX(A,n) is the maximum value in the array

Base case: when n=1 , p(1) is true as an array consists of only single element is the maximum value in the array.

Let’s assume p(k) be true for all 1<k<n (inductive hypothesis)

We have to prove that p(k+1) is also true.

Proof: by substituting k+1 in place of n we get recurrence relation as maximumof(A[k+1-1],getMAX(A,k+1-1))

=>maximumof(A[k],getMAX(A,k))

But earlier we have assumed the p(k) is true by inductive hypotheis i.e p(k) gives maximum of k elements.

Hence p(k+1) gives maximum of k+1th element and p(k).

Hence p(k+1) is true.

*Hence by induction p(n) is true. Hence get\_MAX(A,n) gives the maximum element in the list of n unique numbers.*

**Q2. Design an algorithm to find all the permutations of an array having n unique elements.**

**Give a trace for permutations(A,n), where A={1,2,3} and n=3.**

**Design a recurrence relation for the algorithm and find its time complexity.**

**Give a proof of correctness.**

a)**Algorithm or pseudocode:**

Given an array A of n elements

1. Permutation(A,start,end)
2. If (start isEqual end)
3. Print\_array(A,end)
4. return
5. For i=start to i=end
6. Swap(a[start],a[i])
7. Permutation(A,start+1,end)
8. Swap(a[start],a[i])//backtrack

b) **Tracing**

Enter the size of the array3

\*\*\*\*\* trace of getting maximum elements of the array

\*\*\*\*\*\*\*\*\*called Main function, main(): Activation Record0

Variable Address Value

n 0x61fdd4 3

activation\_record 0x408034 0

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[1 2 3 ],(0,2)): Activation record 1.

Variable value address

start 0 0x61fda8

end1 2 0x61fdb0

activation\_record 1 0x404010

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[1 2 3 ],(1,2)): Activation record 2.

Variable value address

start 1 0x61fd68

end1 2 0x61fd70

activation\_record 2 0x404010

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[1 2 3 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

\*\*\*\*\*\*\*\*\*\*\*Returning from permutation function--->permutation(arr[1 2 3 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

**the permutation of the array is 1 2 3**

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[1 3 2 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

\*\*\*\*\*\*\*\*\*\*\*Returning from permutation function--->permutation(arr[1 3 2 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

**the permutation of the array is 1 3 2**

\*\*\*\*\*\*\*\*\*Returning from the permutation function---->permutation(arr[1 2 3 ],(1,2))--> activation record 2

Variable value address

start 1 0x61fd68

end1 2 0x61fd70

activation\_record 2 0x404010

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[2 1 3 ],(1,2)): Activation record 2.

Variable value address

start 1 0x61fd68

end1 2 0x61fd70

activation\_record 2 0x404010

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[2 1 3 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

\*\*\*\*\*\*\*\*\*\*\*Returning from permutation function--->permutation(arr[2 1 3 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

**the permutation of the array is 2 1 3**

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[2 3 1 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

\*\*\*\*\*\*\*\*\*\*\*Returning from permutation function--->permutation(arr[2 3 1 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

**the permutation of the array is 2 3 1**

\*\*\*\*\*\*\*\*\*Returning from the permutation function---->permutation(arr[2 1 3 ],(1,2))--> activation record 2

Variable value address

start 1 0x61fd68

end1 2 0x61fd70

activation\_record 2 0x404010

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[3 2 1 ],(1,2)): Activation record 2.

Variable value address

start 1 0x61fd68

end1 2 0x61fd70

activation\_record 2 0x404010

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[3 2 1 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

\*\*\*\*\*\*\*\*\*\*\*Returning from permutation function--->permutation(arr[3 2 1 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

**the permutation of the array is 3 2 1**

\*\*\*\*\*\*\*\*\*\*\*\*Calling permutation function-->permutation(arr[3 1 2 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

\*\*\*\*\*\*\*\*\*\*\*Returning from permutation function--->permutation(arr[3 1 2 ],(2,2)): Activation record 3.

Variable value address

start 2 0x61fd28

end1 2 0x61fd30

activation\_record 3 0x404010

**the permutation of the array is 3 1 2**

\*\*\*\*\*\*\*\*\*Returning from the permutation function---->permutation(arr[3 2 1 ],(1,2))--> activation record 2

Variable value address

start 1 0x61fd68

end1 2 0x61fd70

activation\_record 2 0x404010

\*\*\*\*\*\*\*\*\*Returning from the permutation function---->permutation(arr[1 2 3 ],(0,2))--> activation record 1

Variable value address

start 0 0x61fda8

end1 2 0x61fdb0

activation\_record 1 0x404010

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Returning from main function: Activation record\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 0.

Variable value address

n 3 0x61fdd4

activation\_record 0 0x408034

c)**recurrence relation and time complexity**

T(n)=n\*(O(1)+T(n-1)+O(1))

=n\*(n-1)\*(n-2)\*……….(n-k)\*(O(1)+T(n-k-1))=n!

**But for printing each permutation takes O(n). hence the time complexity is O(n\*n!).**

d)**Proof of correctness of the algorithim:-**

we can prove the algorithim by method of induction

Let p(n):denotes the permutation of all the numbers where n denotes total no of elements

we have to prove that p(n) is true

Base case: when n=1, p(1) is always true as a single element is the permutation of itself.There is only 1 permutation.

Inductive hypothesis: Let p(k) be true for all 1< k<n. i.e p(k) gives all the permutation of numbers from 1 to k.

Inductive step: we have to prove that p(k+1) is also true.

From inductive hypothesis we know that p(k) gives all permutation of elements from 1 to k. Now (k+1)th element calls p(k) each time which by our assumption gives all the permutation from 1 to k. So (k+1)th element can be permutated with each permutations of p(k) which gives all the permuations of (k+1) elements. Hence p(k+1) is true.

***Hence p(n) is always true by fundamental principle of induction.i.e p(n) gives all the permutaitons of n numbers.***

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* THE END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*