Algorithms Advanced updated on Jan 2025

1. How do you count number of bits in a Integer?

1. Not Possible

2. Possible but have to convert Integer to binary format and check manually
3. Solution exists in C
4. Possible using assembly language

Solution: 3

```
int no_of_bits = 0;
int given_number;       //this is the number taken as input
while (given_number != 0)
{
if (given_number % 2)

no_of_bits ++;

given_number = given_number >> 1;
}
```

This algorithm checks if the number is zero, in which case the number of bits set will be zero, and enters 'while' loop if it is not zero. Every time it enters the loop it will check if the right most digit is set or not by doing mod 2 (any number mod 2 returns 1 if the right most digit is one else returns 0). If it is set then it will increment the counter by 1 and then right shifts the number by 1 digit. Like this it will check for all digits until the number becomes zero and exits the loop. When the loop exits *no_of_bits* will be the answer.

Alternate Solution:

```
int count = 0;

int n;       //input number

while (n)

{

n = n & (n-1);

count ++;

}
```

2. Given a string, can you find the first non-repeated character in it?

1. Solution does not exists

2. Solution exists @ O(n^2)

3. Solution exists @ O(n.logn)
Solution: 3

/** declare a linked list named *llist,* also write basic functions to add and delete node from a linked list

```
char given_array [];                //this is the array taken as input

add (llist, given_array [0] );      // adds the first character of array to linked list (make sure                    //the add function appends the new value at the end)

for (int i = 1; given_number[i]; i++)

{

do

{

if (llist->value == given_array [i] )

delete (llist, given_array [i] );

else

add (llist, given_array [i] );

}while (llist = llist->next);

}
```

Form a linked list with the characters in the string as nodes. While adding the character to the list, check if its already existing. If it exists then delete that node from the linked list and go for the next character. Once the whole string is completed, the node at the head of the linked list contains the first non-repeated character of the string.

3. What does the following algorithm do?

1. Reverse an integer bit-wise. E.g., if input is 12 (00001110) output will be 01110000.

2. Reverse an Integer normally. E.g., if input is 1234, output will be 4321

3. Right Rotate a number by N positions

4. Left Rotate a number by N positions

Solution:1

```
int given_number;       //this is the number taken as input
int output_number = 0;

for ( int i = 0; i < INTEGER_SIZE; i++ )

{

output_number = output_number << 1;

if (given_number % 2)

output_number ++;

given_number = given_number >> 1;

}
```

4. Given an algorithm to revers an integer. Output of 1234 should be 4321.

1. Reverse an integer bit-wise. E.g., if input is 12 (00001110) output will be 01110000.

2. Reverse an Integer normally. E.g., if input is 1234, output will be 4321

3. Right Rotate a number by N positions

4. Left Rotate a number by N positions

Solution:2

```
int given_number;       //this is the number taken as input
int output_number = 0;

while (given_number != 0)

{

output_number = output_number * 10;

output_number = given_number % 10;

given_number = given_number / 10;

}
```

5. Can you right-rotate a string by k-positions.

1. Not Possible in C

2. Solution exists @ O(log n)

3. Solution exists @ O(n^2)

4. Solution exists @ O(n.log n)

Solution:4

```
int k;          // given as input - 'k' position to be rotated

char input [];      // given as input

int len;          // length of the input string

for (int i = 0; i < k; i++)

{

char c = input [len - 1];

for (int j = len -1; j > 0; j++)

input [j] = input [j -1];

input [0] = c;

}
```

6. Can you compare two strings whether the second string is rotated version of the first? E.g., 'GOOGLE' is rotated version of 'GLEGOO', 'OOGLEG', etc,.

1. No Solution

2. Solution exists @ O(log n)

3. Solution exists @ O(n)

4. Solution exists @ O(n.log n)

Solution:3

Solution for this problem is a bit tricky one. First we have to form a circularly linked list of the two strings. Now declare two pointers for the two lists and move both the pointers forward. If the current characters match, move both the pointers by 1 node else move only the first pointer. One more condition which has to be checked is lets say both the strings are moved by few characters and they matched but after that they are different, in this case the second pointer has to be reset to its head. At some point if both lists have same nodes for the iterations equal to length of the string then they are rotated versions else if first list reaches the end (i.e., completes traversing for the length of the string) then they are not rotated versions.

7. What does the following algorithm do?

```
int f1 (int x,int y)
{
if (!y)
return(x);
else
return(f1(y,x(mod)y);
}
```

1. Find X^Y

2. Find GCD of 2 Numbers

3. Find LCM of 2 Numbers

4. Find Yth element in GP of X, XY, XY2,..

Solution:2

8. In selecting the pivot for QuickSort, which is the best choice for optimal partitioning:

a.The first element of the array

b.The last element of the array

c.The middle element of the array

d.The largest element of the array

e.The median of the array

Solution:e

While the choices a,b,c will always not guarantee **O(NlogN)** complexity,choice d always gives quadratic run time.choice e guarantees even partition of the array.Hence it is the optimal partition.

hence the sol is e.

9. What is the worst case scenario for QuickSort.

1. Pivot is Maximum

2. Pivot is Median

3. Both of the above.

Solution:1

In the worst case,the pivot selected will always be the maximum element leading to quadratic time complexity.In this case as it depicts the behaviour of bubble sort,where in maximum element always bubbles to the end

10. What the following algorithm do?

```
void rep_str (char *str)
{
if (*str!=NULL)
rep_str(str+1);
printf (“%c”,str);
}
```

1. Prints next character of every character in the string

2. Prints the string character by character

3. Prints reverse of the string

4. Prints just last character of the string

Solution:3