

CS 2001 Data Structures

Time Complexity and Big-Oh

Problem1

Assume that basic operations and input output take single time units to complete. Calculate the time complexity function T(n) and big-oh of T(n) for the following program fragments:

1. <pre>int s, i,n; cin>>n; s = 0; for (i=n;i>=1;i--) s++;</pre>	5. <pre>int i,j,sum,n; cin>>n; for (i=1;i<=n;i=i*2) { cout << i; Sum=0; for (j=1;j<=i;j=j*2) { Sum++; cout << i; } cout << Sum; }</pre>
2. <pre>int sum, i, j, k,n; sum = 0; k=0; cin>>n; for (i=0;i<n;++i) { k=0; sum++; for (j=n;j>0;j=j-3) { sum++; k=k*sum; cout << k; } cout << k; }</pre>	6. <pre>int i,j,sum,n; sum = 0; cin>>n; for (i=n;i>=1;i=i/5) { cout << i; cout << sum; for (j=1;j<=i;++j) { cout << j; cout << "*"; sum++; } sum =0; }</pre>
3. <pre>int sum,i,j,n; sum = 0; cin>>n; for (i=1;i<n;i=i*2) for (j=1;j<n;j=j*2) sum++;</pre>	
4. <pre>int sum,i,j,k,n; sum = 0; cin>>n; for (i=1;i<n;i=i*2) { for (j=0;j<n;++j) for (k=1;k<=n;k=k*2) sum++; }</pre>	

Problem 2

Find the **best case and worst case** time complexity for the following code segments and show working along with an example for best and worst case scenarios. Clearly give the time complexity function as well as the asymptotic growth rate (Give tight bounds).

GCD Code fragment

```
int GCD(int n,int m){  
    int min;  
    if(n<m)  
        min = n;  
    else  
        min = m;  
    for(int i=min; i>1; i--)  
        if(n%i ==0 && m%i==0)  
            return i;  
  
    return 1;  
}
```

Code fragment that returns the binary representation of num in arr

```
void binary(int num, int arr[], int &bits)  
{  
    int temp = num;  
    bits = 0;  
    while(temp>0){  
        temp = temp/2;  
        bits++;  
    }  
  
    int i = bits-1;  
    while(num > 0){  
        arr[i] = num%2;  
        i--;  
        num = num/2;  
    }  
}
```

Code fragment for bubble sort

```
void bubbleSort(int arr[],int n)  
{  
    bool done = false;  
    for(int i = 1; (i < n) && !done; i++)  
    {  
        done = true;  
        for (int j=0; j <n-i; j++)  
        {  
            if (arr[j+1] < arr[j])  
            {  
                swap(arr[j+1],arr[j]);  
                done = false;           //a swap is made and so sorting continues  
            }  
        }  
    }  
}
```