CS218 Data Structures

Spring 2020 FAST-NU, Lahore

Instructor: Hafiz Muhammad Hamza

Assignment 1 – Time Complexity Analysis and Asymptotic Bounds

Q1. Calculate Big-O (Big-O), Big-Ω (Big-Omega) and Big- Θ (Big-Theta) of the following functions:

a) $4n^3+n^2\log n$
b) $3n^4+2(\log n)^3$

Q2. Show that $f(x)=3x^2+2x+7$is O ¿).

Q3. Formulate $T(n)$ and then compute Big-O of the following pieces of code.

(a)

```
int count=0;
for (int i=1;i<n;i=i*3){
      for (int j=0;j<i;++j){
            count++;
      }
      cout << count << endl;
}
```

(b)

```
int count;
for (int i=1;i<=n;++i){
      cout << i;
      count=0;
      for (int j=1;j<=i;++j){
            cout << j;
            for (int k=1;k<=j;++k){
                  cout << k;
                  count++;
            }
      }
      cout << count;
}
```

(c)

```
int c=2;
for (int i=1;i<=n;i+=2*c){
      cout << i;
      for (int j=1;j<=n;j*=2+c){
            cout << j;
      }
}
```

(d)

```
void foo(int n){
      n=n*n;
      for (int i=0;i<n;i++){
            if(n<10)
                  cout << i;
            else
                  break;
      }
}
```

Q4. Show that the time complexity of Binary Search algorithm is $O(\log(n))$.

Q5. Given the code below, formulate $T(n)$ and compute Big-O of ***selectionSort***.

```
void swap(int &n1, int &n2){
     int t=n1;
     n1=n2;
     n2=t;
}

int findSmallest(int index, int *arr, int n){
     int smallest=index;
     for (int i=index+1;i<n;i++){
          if (arr[i]<arr[smallest])
               smallest=i;
     }
     return smallest;
}

void selectionSort(int *arr, int n){
     for (int i=0;i<n-1;i++){
          int smallest=findSmallest(i, arr, n);
          swap(arr[j], arr[smallest]);
     }
}
```

Q6. Suppose we have an array containing red, yellow and green balls. Give an algorithm (in the form of code or pseudocode) to arrange them in groups e.g. all red balls are grouped at the start of the array, all yellow balls in the middle and all green balls at the end of the array. The time complexity of your algorithm must be $O(n)$ and it must not use more than $O(1)$auxiliary space.

Q7. Given the code for Bubble sort.

```
void bubbleSort(int *arr, int n){
     for (int i=0;i<n-1;i++){
          for (int j=0;j<n-i-1;j++){
               if (arr[j]>arr[j+1])
                    swap(arr[j],arr[j+1]);
          }
     }
}
```

a) Perform time complexity analysis of Bubble sort and show its running time in the form of Big-O.

b) The running time of the given algorithm is $\Omega(n^2)$ i.e. it takes n^2 time in the best case too. Optimize the algorithm in such a way that it has running time of $\Omega(n)$.