NSUPS Bootcamp Week 5

Time to learn few advanced stuffs

Topics to be covered

1. Precision
2. Sorting
3. Dequeue
4. Priority Queue
5. Merge two sorted array
6. Binary Search

Precision

1. Due to floating point error in C++ we must handle double comparison very carefully.
2. While comparing two double values, add a very small value(like 0.00000001) with one of them.
3. Always focus on the option that will benefit you.

Bubble Sort

1. Compare each of the adjacent pair of elements N times. If the first element is greater than it’s next element then swap them.
2. Why checking N times? A single number will participate in a swap operation at max N times. It’ll ensure that all possible swap operation is done and there won’t exit any i such that A[i]>A[i+1].
3. C++ has built in sort function: sort(A,A+N); where A is the array and N is the length.

Sample Code of Bubble Sort

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

Compare function

1. Let’s take an example, there are N students and you are given their name, cgpa and student id. You have to sort them according to their student id. How do you sort them?
2. We can use an structure type array to store the informations for each student.
3. We can still use the built in sort function. But now you have to send the information to c++ in what basis it’ll sort the array.
4. That’s why we need compare function.

Sample Code of Sorting Using Compare Function

```
struct student{
    string name;
    int cgpa;
    int student_id;
    student(){}
};

/// C++ wants to place a before b
bool cmp(student a, student b){
    if(a.student_id < b.student_id) return true; /// Okay, i also want the same thing
    return false; /// Nope, i don't want this
}

int N;
student A[100];

int main () {
    sort(A,A+N, cmp);
}
```

Dequeue

1. Similar to queue. But now we can perform both insert and delete operations in both back and front side.
2. *front()* returns front element like queue
3. *pop_front()* erase the front element
4. *back()* returns the back element
5. *pop_back()* erase the back element
6. How to solve 1093 - Ghajini

Priority Queue

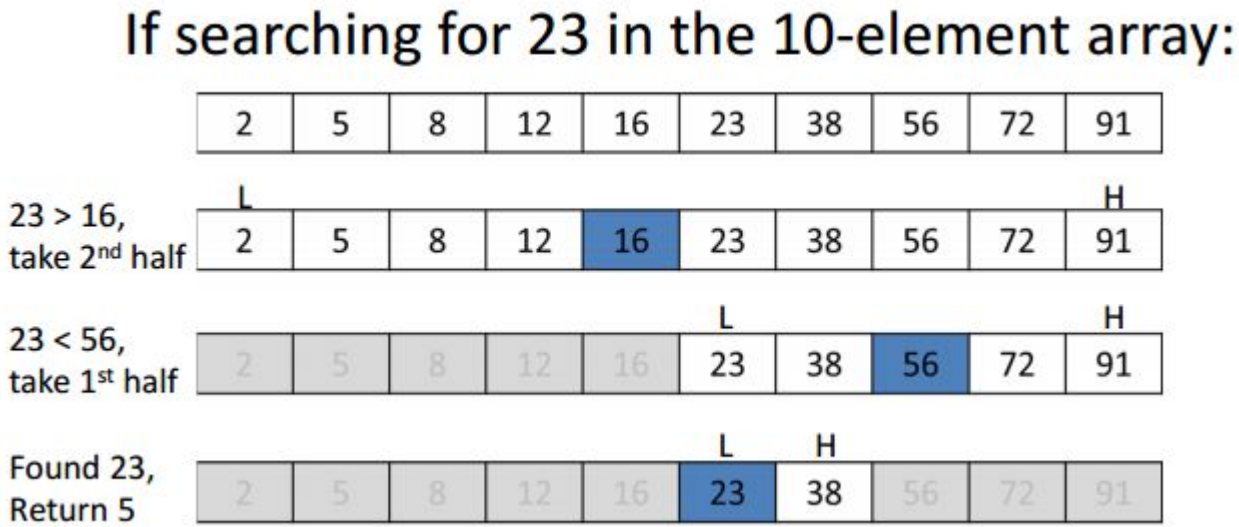
1. Priority queue is basically a heap data structure. We will discuss more about that later.
2. Returns the maximum element in the list.
3. *pop()* erase the maximum element from the priority queue
4. *top()* returns the maximum element in the priority queue
5. You can also use min priority queue which will return the minimum element.
6. Declare your priority queue like this: *priority_queue<int, vector<int>, greater<int> > PQ;*
7. Now let's solve UVa 10954 Add All

Merge 2 Sorted Array

1. You are given two arrays A and B which are already sorted. Now you have to build another array C which will contain all the elements from A and B and must also be sorted.
2. Use the property that for any pair(i, j) if i<j then A[i] will always come before A[j] in array C.
3. The same thing goes for array B too.

Binary Search

1. Given a sorted array find the position of x in that array in NlogN.
2. Will show in the class.



Sample Code of lower_bound and upper_bound

```
int lower_bound(int A[], int N, int X){
    int L = 0, R = N-1;
    while(L<=R){
        int M = (L+R)/2;
        if(A[M]>=X){
            R = M-1;
        }else{
            L = M+1;
        }
    }
    return L;
}

int upper_bound(int A[], int N, int X){
    int L = 0, R = N-1;
    while(L<=R){
        int M = (L+R)/2;
        if(A[M]>X){
            R = M-1;
        }else{
            L = M+1;
        }
    }
    return L;
}
```

Resource

1. Binary Search Hackerearth
2. Binary Search Top Coder
3. Priority Queue C++
4. Dequeue C++
5. STL C++

That’s all for today! Thank you!