L2

From Last SIG, It was clear that competitive programming is made up of easy questions for first 1-3 problems, which can take us to top 1000-2000 coders of the country.
After observation, we will notice from previous questions that many basic algorithms can be used in these questions as well.

In the given 41 questions, basic algorithms that we have encountered till now are

1. Linear Search
2. Binary Search
3. Sorting
    - Creating a compare function for custom sorting
4. Prefix Sum Array
5. Euclidean GCD

There are few concepts whose basics are often helpful in solving those questions

1. Hashing
2. STL vectors
3. Output to some accuracy
4. Importance to size of integer
5. Analysing time complexities
6. Modular Mathematics
7. Basic gate operations
8. STL maps
9. Absolute of a numbers

So let's start

1. Linear Search

If I ask you to find something from a list in which items are all randomly arranged. How will you look for it?
Answer is pretty simple that you will go through each item and check if it is the item that you were searching for it or not.

Its pretty straight forward:

```
for(int i = 0; i < size_of_array; ++i){
    if(arr[i] == item){
        // Code for item found
    }
}
```

2. Binary Search

This is one of the best method ever. Believe it or not but it is used everywhere, from searching something in an array of size 10 to reddit databases. It is working everywhere.

Do you want to see how powerful binary search is?

Given a sorted array, if you use Linear search of order $O(n)$ and $n = 10^6$.

Then Linear search may go upto 10^6 operations.

But Our boy binary search here need at max 22 opertions to find the element or to say if not present.

Wasn't that cool?

There are many questions where Binary search is used in different methods

1. Using to do normal search in the array

It is the most easy question of binary search. You just need to find the number.

2. To find rank of a number in an array

This is one the most important use of binary search. We will be using this in many questions and techniques.

- This technique is popular to bring down time complexity of famous question **Longest increasing subsequence** from $O(n^2)$ to $O(nlogn)$
- Taking about questions where it can be used. Then, I will just refer you to the 2^{nd} most solved question of September Lunchtime 2018.
  Solution to which is using binary search

```
bool lieIn(PLL a, int t){
    return (a.first <= t && a.second > t);
}

ll BinarySearch(PLL *arr, int n, ll t){
    if(t >= arr[n-1].second)return -1;
    if(t <= arr[0].first)return arr[0].first - t;
    int l = 0;
    int r = n-1;
    while(l <= r){
        int mid = (l+r)/2;
        if(lieIn(arr[mid],t))return 0;
        if(mid != 0 && arr[mid-1].second <= t && arr[mid].first > t)return arr[mi
        if(mid != n-1 && arr[mid].second <= t && arr[mid+1].first > t)return arr[
        if(arr[mid].first < t)l = mid+1;
        else r = mid-1;
    }
}

void solve(){
    int n,m;
    cin >> n >> m;
    PLL *arr = new PLL[n];
    loop(i,0,n){
        int l,r;
        cin >> l >> r;
        arr[i] = mp(l,r);
    }
    sort(arr,arr+n);
    while(m--){
        ll a;
        cin >> a;
        cout << BinarySearch(arr,n,a) << endl;
    }
}

int main(){
    int t = 1;
    cin >> t;
    while(t--)solve();
    return 0;
}
```

3. One of the most important techinque of binary search is its operation on real numbers

This is really important so we will discuss this is class.

But for the idea, rather than creating a real array, consider an array of minimum and maximum values

Sample Questions

Q21 and Q22 in L1

Sample code for Binary search i.e. to index of an element

```
int binarySearch(vector<int> arr, int x){
    int low = 0;
    int high = arr.size()-1;
    while(low <= high){
        int mid = (low + high)/2;
        if(arr[mid] == x)return mid;
        if(arr[mid] > x)high = mid-1;
        else low = mid+1;
    }
    return -1;
}
```

Sorting

We can do different kind of sorting but right now we are not focusing on hand implementation of them. Right now we are just focused on using STL library method to sort an array, vector or a string(lexicographical order)

- **Normally sorting an array of size n**
  sort(arr,arr+n);
- **Sorting an array of size n in non-ascending order**
  sort(arr,arr+n,greater());
- **Normally sorting a vector of size n**
  sort(v.begin(),v.end());
- **Sorting an array of size n in non-ascending order**
  sort(v.begin(),v.end(),greater());

Now lets how to arrange a comparitive function for sorting
Suppose we use an abstract data type for array and now we need to sort it. What do we do then? LOL. Any idea?
We define a function which takes two parameters and returns true if first parameters needs to before second one

```
bool comp(pair<int,int> a, pair<int,int> b){
    if(a.first < b.first)return true;
    if(a.first > b.first)return false;
    if(a.second < b.second)return true;
    return false;
}
int main(){
    pair<int,int> arr[100];
    //SOME INPUT
    sort(arr,arr+n,comp);
}
```

Prefix Sum Array

Welcome to the first Dynamic programming algorithm or techinque that we are going to use here

What is a prefix sum array

It is an array in which i_{th} element holds sum of all numbers from 0 to i

Implementation

```
vector<int> prefixSum(arr.size(),0);
prefixSum[0] = arr[0];
for(int i = 1; i < arr.size(); ++i){
    prefixSum[i] = prefixSum[i-1] + arr[i];
}
```

What the use?

Want to count the sum of subsequence, helps you do it in $O(1)$ time complexity

Euclidean GCD

Please think of a time complexity for a program which check GCD of two numbers.
$O(min(a,b))$ is the most popular naive answer.
But Euclid helped us to find an efficient method which we programmers regularly use

According to Euclid's GCD algorithm

$$GCD(a,b) = GCD(b\%a, a)$$

This reduces time complexity from $O(min(a,b))$ to $O(log\{min(a,b)\})$

We will be practicing this algorithm for a long till you get hang of it.
Though for contest one can directly use `__gcd(a,b)` which is Euclidean GCD as well.

Hashing

This is one of the most important concept as well. For this concept I will recommend to attend SIG but otherwise, hackerearth have a very good tutorial on Hashing
Hackerearth Hashing Tutorials

Questions for L2

1. Anagrams(Hashing)
2. Bishu and Soldier(Rank Binary Search and Prefix Sum array)
3. Picu Bank (Binary Search on real Numbers)
4. BooBoo and Upsolving(Binary Search on real Numbers)
5. Gaurav and SubArray(Binary Search on possible answers)
6. Compare Strings(Binary Search or Queues)
7. Sumit And Chocolates(Binary Search with Ranks and prefix Sum Array)
8. Candies (Binary Search on possible answers)
9. Three Parts of an array (Linear or Binary searches)

10. World Cup (Linear Search or Binary Search)
11. Letters
12. Save Patients
13. Max Power (Its hard to understand, what do author wants)
14. Ice Cream Parlor(Linear Search)
15. Missing Numbers(Using maps)
16. Agressive Cows(Binary Search)
17. Horses
18. EID
19. DIVSUM (Sieve)