Code Template

Binary search

```
int binarySearch(int &arr, int x)
{
    int l=0, r=arr.size()-1;
    while (l <= r) {
        int m = l + (r - l) / 2;

        // Check if x is present at mid
        if (arr[m] == x)
            return m;

        // If x greater, ignore left half
        if (arr[m] < x)
            l = m + 1;

        // If x is smaller, ignore right half
        else
            r = m - 1;
    }
    // not present
    return -1;
}
```

BFS

```
#define pii pair<int,int>
#define uu first
#define vv second

int fx[]={1,-1,0,0};
int fy[]={0,0,1,-1};
int d[1000][1000],vis[1000][1000]; //d means destination from source.
int row,col, dx, dy, sx, sy;
void bfs()
{
  vis[sx][sy]=1;
  queue<pii>q;
  q.push(pii(sx,sy));
  while(!q.empty())
  {
    pii top=q.front(); q.pop();
    for(int k=0;k<4;k++)
    {
```

```
      int tx=top.uu+fx[k];
      int ty=top.vv+fy[k];

      if(tx>=0 and tx<row and ty>=0 and ty<col and vis[tx][ty]==0)
      {
        vis[tx][ty]=1;
        d[tx][ty]=d[top.uu][top.vv]+1;
        q.push(pii(tx,ty));
      }
    }
  }
}
```

Bicolor example:

```
vector<vector<int>> vv;
int vis[205];

bool bicolorable()
{
    memset(vis, -1, sizeof(vis));

    queue<int> q;
    q.push(0);
    int color = 1;
    vis[0] = color;
    while(!q.empty()){
        int t = q.front();q.pop();
        color = vis[t];
        for(int i:vv[t]){
            if(vis[i] == -1){
                vis[i] = (1^color);
                q.push(i);
            }
            else{
                if(vis[i] == color) return false;
            }
        }
    }
    return true;
}
```

Level Order Trversal:

```
queue<TreeNode*>q;
q.push(root);
while(q.empty){
    int sz = q.size();
```

```
    vector<int>v;
    while(sz--){
        TreeNode* f=q.front();
        q.pop();
        v.push_back(f->val);
        //work with f
        if(f->left) q.push(f.left);
        if(f->right) q.push(f->right);
    }
    vvv.push_back(v);
}
```

DSU

```
int par[maxx];

int findPar(int u)
{
    if(par[u] == u) return u;
    return par[u] = findPar(par[u]);
}

void join(int u, int v)
{
    int a = findPar(u);
    int b = findPar(v);

    if(a != b){
        par[a] = b;
    }
}

int main()
{
    for(int i = 0; i < maxx; i++){
        par[i]=i;
    }
}
```

Segment Tree

```
#define maxx 100001

int arr[maxx];
int tree[maxx*3];
```

```
void init(int node, int b, int e)
{
    if(b==e){
        tree[node]=arr[b];
        return;
    }
    int left = node*2;
    int right = node*2 +1;
    int mid = (b+e)/2;

    init(left, b, mid);
    init(right, mid+1, e);

    tree[node]=tree[left]+tree[right];
}

int query(int node, int b, int e, int i, int j)
{
    if(b>j || e<i){//out of range
        return 0;
    }
    if(i<=b && e<=j){//completely in range
        return tree[node];
    }
    int left=node*2;
    int right=node*2+1;
    int mid=(b+e)/2;

    int p1=query(left, b, mid, i, j);
    int p2=query(right, mid+1, e, i, j);

    return p1+p2;
}

void update(int node, int b, int e, int i, int newVal)
{
    if(i<b || i>e){
        return;
    }
    if(b>=i && e <=i){
        tree[node]=newVal;
        return;
    }
    int left=node*2;
    int right=node*2+1;
    int mid=(b+e)/2;

    update(left, b, mid, i, newVal);
    update(right, mid+1, e, i, newVal);

    tree[node]=tree[left]+tree[right];
}
```

Segment Tree with Lazy Propagation

```
//Horrible example- the simplest exercise
#define maxx 100001

int arr[maxx]={0};//initialization optional

struct{
    long long sum, prop=0;
} tree[maxx*3];

void init(int node, int b, int e)
{
    if(b==e){
        tree[node].sum=arr[b];
        tree[node].prop=0;
        return;
    }
    int left = node*2;
    int right = node*2 +1;
    int mid = (b+e)/2;

    init(left, b, mid);
    init(right, mid+1, e);

    tree[node].sum=tree[left].sum+tree[right].sum;
    tree[node].prop=0;
}

void update(int node, int b, int e, int i, int j, long long x)
{
    if(i>e || j<b){
        return;
    }
    if(i<=b && e<=j){
        tree[node].sum += (e-b+1)*x;
        tree[node].prop+=x;
        return;
    }
    int left=node*2;
    int right=node*2+1;
    int mid=(b+e)/2;

    update(left, b, mid, i, j, x);
    update(right, mid+1, e, i, j, x);

    tree[node].sum=tree[left].sum+tree[right].sum + (e-b+1)*tree[node].prop;
}

long long query(int node, int b, int e, int i, int j, long long x=0)
{
    if(i>e || j<b){
```

```
        return 0;
    }
    if(b>=i && j >= e){
        return tree[node].sum + (e-b+1)*x;
    }

    int left = node*2;
    int right = node*2+1;
    int mid = (b+e)/2;

    long long p1 = query(left, b, mid, i, j, x+tree[node].prop);
    long long p2 = query(right, mid+1, e, i, j, x+tree[node].prop);

    return p1+p2;
}
```