Reference Book

Author: Abu Bakar Istiak

Compile: g++ -std=c++17 -o filename.exe filename.cpp

Run: ./filename.exe

1. Template:

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define pi pair<ll, ll>
#define vi vector<ll>
#define vpi vector<pi>
#define pb push_back
#define endl "\n"
#define yes cout << "YES\n"
#define no cout << "NO\n"
#define cyes cout << "Yes\n"
#define cno cout << "No\n"
#define minus cout << -1 << endl
#define zero cout << 0 << endl
#define afor(i,a,b) for (ll i = (a); i < (b); ++i)
#define rfor(i,a,b) for (ll i = (a); i >= (b); --i)
#define asort(v) sort((v).begin(), (v).end())
#define rsort(v) sort((v).begin(), (v).end(), greater<>())
#define fast() ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define MAX 100000
#define MOD 998244353
void solve() {

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

1. Is Power of Two

```
bool isPowerOfTwo(int n) {
   return n > 0 && (n & (n - 1)) == 0;
}
```

2. Reverse Integer

```
int reverseInt(int n) {
   int rev = 0;
   while (n != 0) {
      rev = rev * 10 + n % 10;
      n /= 10;}
   return rev;
}
```

3. Fibonacci Series:

```
int fibo(int n){
   if(n==0 || n==1){
      return n;
   }
   int ans = fibo(n-1) + fibo(n-2);
   return ans;
}
void solve(){
   int n;
   cin >> n;
   cout << fibo(n) << endl;
}
```

4. Factorial:

```
int fact(int n){
   if(n==0) return 1;
   int chotoFactorial = fact(n-1);
   return chotoFactorial *n;
}  void solve(){
   int n;
   cin >> n;
   cout << fact(n) << endl;
}
```

5. Sieve:

```
void solve(){
   int n; cin >> n;
    vector<bool> prime(n+1, true);
    for(int i=2; i*i<=n; i++){
       if(prime[i]){
          for(int j=i+i; j<=n; j+=i){
             prime[j]=false;
          }}}
    for(int i=2; i<=n; i++){
       if(prime[i]){
          cout << i << " ";
       }}
    cout << endl;
}
```

6. Count Digits

```
int countDigits(int n) {
   int cnt = 0;
   while (n > 0) {
      cnt++;
      n /= 10;
   }
   return cnt;}
```

7. Custom Ceil

```
ll ceil_div(ll a,ll b){
     return (a+b-1)/b;
}
```

8. GCD & LCM Function

```
int gcd(int a, int b) {
   return b == 0 ? a : gcd(b, a % b);}
int lcm(int a, int b) {
   return (a / gcd(a, b)) * b;}
```

9. Prime Number Check

```
bool isPrime(int n) {
   if (n < 2) return false;
   for (int i = 2; i * i <= n; i++) {
      if (n % i == 0) return false;
   }
   return true;
}
```

10. Binary Search Template

```
int binarySearch(vector<int>& a, int target) {
  int l = 0, r = a.size() - 1, ans = -1;
  while (l <= r) {
    int mid = (l + r) / 2;
    if (a[mid] == target) return mid;
    if (a[mid] < target) l = mid + 1;
    else r = mid - 1;
  }
  return -1;}
```

11. Prefix Sum Array

```
vector<int> prefixSum(vector<int>& a) {
  int n = a.size();
  vector<int> pre(n);
  pre[0] = a[0];
  for (int i = 1; i < n; i++) {
    pre[i] = pre[i-1] + a[i];
  }
  return pre;}
```

12. Count Frequencies

```
map<char, int> freq;
for (char c : s) freq[c]++;
```

13. Count Set Bits in Integer

```
int countSetBits(int n) {
  int cnt = 0;
  while (n) {
    cnt += (n & 1);
    n >>= 1;
  }
  return cnt;
}
```

14. Binary Node:

```
class Node {
  public:
    int val;
    Node *left;
    Node *right;
  Node(int val) {
    this->val = val;
    this->left = NULL;
    this->right = NULL;
  }};
```

InOrder:

```
void inorder(Node *root){
  if(root==NULL){
    return;}
  inorder(root->left);
  cout << root->val << " ";
  inorder(root->right);
}
```

PreOrder:

```
void preorder(Node *root){
  if(root==NULL){
    return;}
  cout << root->val << " ";
  preorder(root->left);
  preorder(root->right);
}
```

PostOrder:

```
void postorder(Node *root){
  if(root==NULL){
    return;}
  postorder(root->left);
  postorder(root->right);
  cout << root->val << " ";
}
```

15. Input Tree:

```
Node* input_tree(){
  int val; cin >> val;
  Node *root;
  if(val == -1) root = NULL;
  else root = new Node(val);
  queue<Node* > q;
  if(root) q.push(root);
  while (!q.empty())
  {
    // 1st step;
    Node* p = q.front();
    q.pop();
    // 2nd step;
    int l, r;
    cin >> l >> r;
    Node* myLeft;
    Node* myRight;
    if(l==-1) myLeft = NULL;
    else myLeft = new Node(l);
    if(r==-1) myRight = NULL;
    else myRight = new Node(r);

    p->left = myLeft;
    p->right = myRight;

    // 3rd step:
    if(p->left) q.push(p->left);
    if(p->right) q.push(p->right);
  }
  return root;
}
void solve(){
  Node* root= input_tree();
}
```

16. Adjacency List:

```
void solve(){
  int n, e;
  cin >> n >> e;
  vector<int> mat[n];
  while (e--){
    int a, b;
    cin >> a >> b;
    mat[a].push_back(b);
    mat[b].push_back(a);
  }
  for (int i = 0; i < mat[3].size(); i++){
    cout << mat[3][i] << " ";
  }
}
```

17. BFS-src,des:

```
vector<int> v[1005];
bool vis[1005];
int parent[1005]; // path
int level[1005]; // path
void bfs(int src) {
  queue<int> q;
  q.push(src);
  vis[src] = true;
  level[src] = 0; // path
  parent[src] = -1; // path

  while (!q.empty()) {
    int par = q.front();
    cout << par << endl;
    q.pop();

    for (int child : v[par]) {
      if (!vis[child]) {
        q.push(child);
        vis[child] = true;
        level[child] = level[par] + 1; //path
        parent[child] = par; // path
      } } }}
void print_path(int des) {
  if (!vis[des]) {
   cout << "No path to destination node.\n";
    return;
  }
  vector<int> path;
  int x = des;
  while (x != -1) {
    path.push_back(x);
    x = parent[x];
  }
  reverse(path.begin(), path.end());
  for (int val : path) {
    cout << val << " ";
  }
  cout << "\n";
  cout << level[des] << "\n";
}
void solve(){
  int n, e;
  cin >> n >> e;
  while(e--){
    int a,b;
    cin >> a >> b;
    v[a].push_back(b);
    v[b].push_back(a);
  }
int src, des;
cin >> src >> des; // des for path
memset(vis, false, sizeof(vis));
bfs(src);
print_path(des);
}
```

Bfs.cpp-src

```
#include<bits/stdc++.h>
using namespace std;
vector<int> v[1005];
bool vis[1005];
void bfs(int src){
  queue<int> q;
  q.push(src);
  vis[src] = true;
  while (!q.empty()){
    int parent = q.front();
    q.pop();
    cout << parent << endl;
    for(int child : v[parent]){
      if(vis[child] == false){
        q.push(child);
        vis[child] = true;
      }}}}
int main(){
  int n, e;
  cin >> n >> e;
  while(e--){
    int a,b;
    cin >> a >> b;
    v[a].push_back(b);
    v[b].push_back(a);
  }
  int src;
  cin >> src;
  memset(vis, false, sizeof(vis));
  bfs(src);
  return 0;
}
```

18. DFS-source,des:

```
vector<int> v[1005];
bool vis[1005];
int parent[1005];
void dfs(int src) {
  cout << src << " ";
  vis[src] = true;
  for (int child : v[src]) {
    if (!vis[child]) {
      parent[child] = src; // path
      dfs(child);
    }}}
void print_path(int des) {
  if (!vis[des]) {
   cout << "No path to destination node.\n";
    return;}
  vector<int> path;
  int x = des;
  while (x != -1) {
    path.push_back(x);
    x = parent[x];
  reverse(path.begin(), path.end());
  for (int val : path) {
    cout << val << " "; }
  cout << "\n";}
```

```
void solve() {
   int n, e;
   cin >> n >> e;
   while (e--) {
      int a, b;
      cin >> a >> b;
      v[a].push_back(b);
      v[b].push_back(a); // undirected
   }
   int src, des; // src,des for path
   cin >> src >> des; // path
   memset(vis, false, sizeof(vis));
   parent[src] = -1; // path
   dfs(0); // path dfs(src)
   print_path(des); // path}
```

Dfs.cpp-src:

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e+5;
vector<int> v[N];
bool vis[N];
void dfs(int src){
   cout << src << endl;
   vis[src] = true;
   for(int child: v[src]){
      if(!vis[child]){
         dfs(child);
   }}}
int main(){
   int n, e;
   cin >> n >> e;
   while (e--){
      int a,b;
      cin >>  a >> b;
      v[a].push_back(b);
      v[b].push_back(a);
   }
   memset(vis, false, sizeof(vis));
   dfs(0);
   return 0;
}
```

input:

```
6 6
0 1
0 2
0 3
1 4
3 5
3 2
```

19. BFS (Grid):

```
bool vis[20][20];
int dis[20][20];
vector<pair<int, int>> d = {{0,1}, {0,-1}, {-1, 0}, {1, 0}};
int n, m;
char a[20][20];
bool vaild(int i, int j){
if(i<0 || i>=n || j<0 || j>=m && a[i][j] == '#'){
      return false;}
   return true;}
void bfs_grid(int si, int sj){
   queue<pair<int,int>> q;
   q.push({si,sj});
   vis[si][sj] = true;
   dis[si][sj] = 0;
   while (!q.empty()){
      pair<int, int> par = q.front();
      int a = par.first;
      int b = par.second;
      q.pop();
      for(int i=0; i<4; i++){
         int ci = a + d[i].first;
         int cj = b + d[i].second;
         if(vaild(ci, cj) == true && vis[ci][cj] == false){
            q.push({ci, cj});
            vis[ci][cj] = true;
            dis[ci][cj] = dis[a][b] + 1;
         }}}}
void solve(){
   cin >> n >> m;
   for(int i=0; i<n ; i++){
      for(int j=0; j<m; j++){
         cin >> a[i][j];
      }}
   int si, sj;
   cin >> si >> sj;
   memset(vis, false, sizeof(vis));
   memset(dis, -1, sizeof(dis));
   bfs_grid(si, sj);
   cout << dis[2][3];
}
```

Input:

```
3 4
....
....
....
0 0
```

20. BFS (Cycle-undirected):

```
const int N = 1e5+5;
vector<int> adj[N];
bool vis[N];
bool ans = false;
int parentArray[N];

void bfs(int src){
   queue<int> q;
   q.push(src);
   vis[src] = true;
   while (!q.empty()){
      int par = q.front();
      q.pop();
      cout<< par <<" ";
      for(int child : adj[par]){
         if(vis[child] == true && parentArray[par] != child){
            ans = true;
         }
         if(vis[child] == false){
            q.push(child);
            vis[child] = true;
            parentArray[child] = par;
         }}}}
void solve(){
   int n,e;
   cin >> n >> e;
   while (e--){
      int a,b;
      cin >> a >> b;
      adj[a].push_back(b);
      adj[b].push_back(a);}
```

```
    int source;
    cin>>source;
    bfs(source);
    if(ans){
        cout << "Cycle dectected" << endl;
    }else cout << "Cycle not decteded" << endl;
}
```

21. MST_Kruskals:

```
const int n = 1e5+5;
int parent[n];
int group_size[n];
void dsu_initialize(int n){
    for(int i=0; i<n; i++){
        parent[i] = -1;
        group_size[i] = 1;
    }}
int dsu_find(int node){
    if(parent[node] == -1){
        return node;}
    int leader = dsu_find(parent[node]);
    parent[node] = leader;
    return leader;}
void dsu_union_by_size(int node1, int node2){
    int leaderA = dsu_find(node1);
    int leaderB = dsu_find(node2);
    if(group_size[leaderA] > group_size[leaderB]){
        parent[leaderB] = leaderA;
        group_size[leaderA] += group_size[leaderB];
    }else{
        parent[leaderA] = leaderB;
        group_size[leaderB] += group_size[leaderA];
    }}
class Edge{
public:
    int u,v,w;
    Edge(int u, int v, int w){
        this->u = u;
        this->v = v;
        this->w = w;
    }};
bool cmp(Edge a, Edge b){
    return a.w < b.w;
}
void solve(){
    int n,e;
    cin >> n >> e;
    dsu_initialize(n);
    vector<Edge> edgeList;
    while (e--){
        int u,v,w;
        cin >> u >> v >> w;
        edgeList.push_back(Edge(u,v,w));
    }
    sort(edgeList.begin(), edgeList.end(), cmp);
    int totalCost = 0;
    for(Edge ed : edgeList){
        int leaderU = dsu_find(ed.u);
        int leaderV = dsu_find(ed.v);
        if(leaderU == leaderV){
                continue;}
        else{
            dsu_union_by_size(ed.u, ed.v);
            totalCost += ed.w;
        }}
    cout << totalCost << endl;}
```

22. Dijkstra:

```
const int N = 100;
vector<pair<int,int>> v[N];
int dis[N];
class cmp{
    public:
        bool operator()(pair<int,int> a, pair<int,int> b){
            return a.second < b.second;
        }};
void dijkstra(int src){
    priority_queue<pair<int,int>, vector<pair<int,int>>, cmp> pq;
    pq.push({src, 0});
    dis[src] = 0;
    while (!pq.empty()){
        pair<int,int> parent = pq.top();
        pq.pop();
        int node = parent.first;
        int cost = parent.second;
        for(pair<int,int> child : v[node]){
            int childNode = child.first;
            int childCost = child.second;
            if(cost + childCost < dis[childNode]){
                // Path relax;
                dis[childNode] = cost + childCost;
                pq.push({childNode, dis[childNode]});
            }}}}
void solve(){
    int n, e;
    cin >> n >> e;
    while (e--){
        int a,b,c;
        cin >> a >> b >> c;
        v[a].push_back({b,c});
        v[b].push_back({a,c});
    }
    // memset(dis, INT_MAX, sizeof(dis));
    for(int i=0; i<n; i++){
        dis[i] = INT_MAX;
    }
    dijkstra(0);
    for(int i=0; i<n; i++){
        cout << i << "-> " << dis[i] << endl;
    }}
```

```
Input:
5 8
0 1 10
0 2 7
0 3 4
1 4 3
2 4 5
2 1 1
3 4 5
3 2 1
```

23. Dijkstra Path:

```
const ll N = 1e5 + 5;
vector<pi> v[N];
ll dis[N];
ll par[N];
class cmp{
public:
```

```
    bool operator()(pi a, pi b){
        return a.second > b.second;
    }};
void dijkstra(ll s){
    priority_queue<pi, vector<pi>, cmp> pq;
    pq.push({s, 0});
    dis[s] = 0;
    while (!pq.empty()){
        pi parent = pq.top();
        pq.pop();
        ll parentNode = parent.first;
        ll parentCost = parent.second;

        for (pi child : v[parentNode]){
            ll childNode = child.first;
            ll childCost = child.second;
            if (parentCost + childCost < dis[childNode]){
                dis[childNode] = parentCost + childCost;
                pq.push({childNode, dis[childNode]});
                par[childNode] = parentNode;
            }}}}
void solve(){
    ll n, e;
    cin >> n >> e;
    while (e--){
        ll a, b, c;
        cin >> a >> b >> c;
        v[a].push_back({b, c});
        v[b].push_back({a, c});
    }
    for (ll i = 1; i <= n; i++){
        dis[i] = 1e18;
        par[i] = -1;
    }dijkstra(1);
    if (dis[n] == 1e18)
        cout << -1 << endl;
    else{
        ll x = n;
        vector<ll> path;
        while (x != -1){
            path.push_back(x);
            x = par[x];}
reverse(path.begin(), path.end());
        for (ll val : path)
            cout << val << " ";
        cout << endl;
    }}
```

24. Bellman-Ford:

```
class Edge{
public:
    int u,v,c;
    Edge(int u, int v, int c){
        this->u = u;
        this->v = v;
        this->c = c;
    }};
const int N = 1e5+5;
int dis[N];
void solve(){
    int n,e;
    cin >> n >> e;
    vector<Edge> EdgeList;
    while (e--){
        int u, v, c;
        cin >> u >> v >> c;
        EdgeList.push_back(Edge(u,v,c));
    }
    for(int i=0; i<n; i++){
        dis[i] = INT_MAX;
    }
    dis[0] = 0;
    for(int i=1; i<=n-1; i++){
        for(Edge ed : EdgeList){
            int u,v,c;
            u = ed.u;
            v = ed.v;
            c = ed.c;
            if(dis[u] < INT_MAX && dis[u] + c < dis[v]){
                dis[v] = dis[u] + c;
            }}}
for(int i=0; i<n; i++){
        cout << i << " -> " <<dis[i] << endl;
    }}
```

for cycle:

```
bool cycle = false;
    for(Edge ed : EdgeList){
        int u,v,c;
        u = ed.u;
        v = ed.v;
        c = ed.c;
        if(dis[u] < INT_MAX && dis[u] + c < dis[v]){
            cycle = true;
            break;
        }}
```

25. Floyd Warshall:

```
void solve(){
    ll n, e;
    cin >> n >> e;
    ll adj[n][n];
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            adj[i][j] = INT_MAX;
            if (i == j){
                adj[i][j] = 0;
            }}}
    while (e--){
        int a, b, c;
        cin >> a >> b >> c;
        adj[a][b] = c;
    }
    for (int k = 0; k < n; k++){
        for (int i = 0; i < n; i++){
            for (int j = 0; j < n; j++){
                if (adj[i][k] + adj[k][j] < adj[i][j]){
                    adj[i][j] = adj[i][k] + adj[k][j];
                }}}}
```

```
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            if (adj[i][j] == INT_MAX) cout << "INF ";
            else cout << adj[i][j] << " ";
        }
        cout << endl;
    }}
```

26. 0-1_knapsack

```
int knapsack(int n, int weight[], int value[], int W){
    //base case
    if(n == 0 || W==0) return 0;
    if(weight[n-1] <= W){
        int op1 = knapsack(n-1, weight, value, W-weight[n-1]) + value[n-1];
        int op2 = knapsack(n-1, weight, value, W);
        return max(op1, op2);
    }else{
        int op2 = knapsack(n-1, weight, value, W);
        return op2;
    }}
void solve(){
    int n;
    cin >> n;
    int weight[n], value[n];
    for(int i=0; i<n; i++){
        cin >> weight[i];}
    for(int i=0; i<n; i++){
        cin >> value[i];}
    int W; cin >> W;
    cout << knapsack(n, weight, value, W) << endl;
}
```

27. 0-1 Knapsack Top Down:

```
const int maxN = 1000;
const int maxW = 1000;
int dp[maxN][maxW];
int knapsack(int n, int weight[], int value[], int W){
    //base case
    if(n == 0 || W==0) return 0;
    if(dp[n][W] != -1){
        return dp[n][W];
    }
    if(weight[n-1] <= W){
        int op1 = knapsack(n-1, weight, value, W-weight[n-1]) + value[n-1];
        int op2 = knapsack(n-1, weight, value, W);
        return dp[n][W] = max(op1, op2);
    }else{
        int op2 = knapsack(n-1, weight, value, W);
        return dp[n][W] = op2;
    }}
void solve(){
    int n;
    cin >> n;
    int weight[n], value[n];
    for(int i=0; i<n; i++){
        cin >> weight[i];}
    for(int i=0; i<n; i++){
        cin >> value[i];}
    int W; cin >> W;
    for(int i=0; i<=n; i++){
        for(int j=0; j<=W; j++){
            dp[i][j] = -1;
        }}
    cout << knapsack(n, weight, value, W) << endl;
}
```

28. 0-1 knapsack bottom up:

```
void solve(){
    int n;
    cin >> n;
    int weight[n], value[n];
    for(int i=0; i<n; i++){
        cin >> weight[i]; }
    for(int i=0; i<n; i++){
        cin >> value[i]; }
    int W;
    cin >> W;
    int dp[n+1][W+1];
    for(int i=0; i<=n; i++){
        for(int j=0; j<=W; j++){
            if(i == 0 || j == 0){
                dp[i][j] = 0;
            }}}
    for(int i=1; i<=n; i++){
        for(int j=1; j<=W; j++){
            if(weight[i-1] <= j){
                int op1 = dp[i-1][j-weight[i-1]] + value[i-1];
                int op2 = dp[i-1][j];
                dp[i][j]  = max(op1, op2);
            }else{
                int op2 = dp[i-1][j];
                dp[i][j] = op2;
            }}}
    cout << dp[n][W] << endl;
}
```

29. Subset sum top down

```
int dp[1005][1005];
bool subset_sum(int n, int ar[], int s){
    if(n==0){
        if(s==0) return true;
        else return false;
    }
    if(dp[n][s] != -1) return dp[n][s];
    if(ar[n-1] <= s){
        bool op1 = subset_sum(n-1, ar, s-ar[n-1]);
        bool op2 = subset_sum(n-1, ar, s);
        return dp[n][s] = op1 || op2;
    }else{
        return dp[n][s] = subset_sum(n-1, ar, s);
    }}
void solve(){
    int n; cin >> n;
    int ar[n];
    for(int i=0; i<n; i++){
        cin >> ar[i];}
    int s; cin >> s;
    for(int i=0; i<=n; i++){
        for(int j=0; j<=s; j++){
            dp[i][j] = -1;
```

```
    }
}
if(subset_sum(n,ar,s)) cout << "YES" << endl;
else cout << "NO" << endl;
}
```

30. Count of subset sum:

```
int dp[1005][1005];
int subset_sum(int n, int ar[], int s){
  if(n==0){
    if(s==0) return 1;
    else return 0;}
  if(dp[n][s] != -1) return dp[n][s];
  if(ar[n-1] <= s){
    int op1 = subset_sum(n-1, ar, s-ar[n-1]);
    int op2 = subset_sum(n-1, ar, s);
    return dp[n][s] = op1 + op2;
  }else{
    return dp[n][s] = subset_sum(n-1, ar, s);
  }
}
void solve(){
  int n; cin >> n;
  int ar[n];
  for(int i=0; i<n; i++){
    cin >> ar[i];
  }
  int s; cin >> s;
  for(int i=0; i<=n; i++){
    for(int j=0; j<=s; j++){
      dp[i][j] = -1;
    }
}
  cout << subset_sum(n,ar,s);
}
```

31. Equal sum:

```
void solve(){
  int n; cin >> n;
  int a[n];
  int s=0;
  for(int i=0; i<n; i++){
    cin >> a[i];
    s+=a[i];
  }
  if(s%2==0){
    int sum = s/2;
    bool dp[n+1][sum+1];
    dp[0][0] = true;
    for(int i=1; i<=sum; i++){
      dp[0][i] = false;
    }
    for(int i=1; i<=n; i++){
      for(int j=0; j<=sum; j++){
        if(a[i-1] <= j){
          dp[i][j] = dp[i-1][j-a[i-1]] || dp[i-1][j];
        }else{
          dp[i][j] = dp[i-1][j];
        }
    }
}

    if(dp[n][sum]) cout << "YES" << endl;
    else cout << "NO" << endl;
   }
else{
   cout << "NO" << endl;
  }
}
```

32. Title: KMP Pattern Matching

```
vector<int> kmpPrefix(string& s) {
  int n = s.size();
  vector<int> pi(n);
  for (int i = 1, j = 0; i < n; i++) {
    while (j > 0 && s[i] != s[j]) j = pi[j-1];
    if (s[i] == s[j]) j++;
    pi[i] = j;}
  return pi;
}
```

33. Title: Max Subarray Sum

```
int maxSubArray(vector<int>& nums) {
  int maxSum = nums[0], curr = nums[0];
  for (int i = 1; i < nums.size(); i++) {
    curr = max(nums[i], curr + nums[i]);
    maxSum = max(maxSum, curr);
}
  return maxSum;
}
```

34. Title: Generate All Subsets

```
vector<vector<int>> subsets;
vector<int> current;
void generate(int i, vector<int>& nums) {
  if (i == nums.size()) {
    subsets.push_back(current);
    return;
}
  generate(i + 1, nums);
  current.push_back(nums[i]);
  generate(i + 1, nums);
  current.pop_back();
}
```

35. Title: Fibonacci DP

```
const int MAXN = 1e5;
int dp[MAXN];
int fib(int n) {
  if (n <= 1) return n;
  if (dp[n] != -1) return dp[n];
  return dp[n] = fib(n-1) + fib(n-2);
}
```

4.2 Line Segment Intersection(CSES)

/* There are two line segments: the first goes through the points (x1,y1) and (x2,y2), and the second goes through

the points (x3,y3) and (x4,y4). Your task is to determine if the line segments intersect, i. e., they have at least one common point. Input

The first input line has an integer t: the number of tests. After this, there are t lines that describe the tests. Each

line has eight integers x1, y1, x2, y2, x3, y3, x4 and

y4. Output

For each test, print "YES" if the line segments intersect and "NO" otherwise. Constraints

* 1 <= t <= 10^5

* -10^9 <= x1,y1,x2,y2,x3,y3,x4,y4 <= 10^9

* (x1,y1) != (x2,y2) * (x3,y3) != (x4,y4)

Example Input: 5

1 1 5 3 1 2 4 3

1 1 5 3 1 1 4 3

1 1 5 3 2 3 4 1

1 1 5 3 2 4 4 1

1 1 5 3 3 2 7 4

Example Output: NO

YES

YES

YES

YES */

C++ Solution :

```
#define int long long
#define P complex<int>
#define X real()
#define Y imag()
int cross(P a, P b, P c)
{
   P u = b - a;
   P v = c - a;
   return (conj(u) * v).Y;
}
bool comp(const P &a, const P &b){
   return (a.X == b.X) ? (a.Y < b.Y) : (a.X < b.X);
}
bool mid(P a, P b, P c){
   vector<P> v = {a, b, c};
   sort(v.begin(), v.end(), comp);
   return (v[1] == c);
}
int sgn(int x){
   return (x > 0) - (x < 0);
}
bool check(P a, P b, P c, P d){
   int cp1 = cross(a, b, c);
   int cp2 = cross(a, b, d);
   int cp3 = cross(c, d, a);
   int cp4 = cross(c, d, b);
   if (cp1 == 0 && mid(a, b, c))
      return 1;
   if (cp2 == 0 && mid(a, b, d))
      return 1;
   if (cp3 == 0 && mid(c, d, a))
      return 1;
   if (cp4 == 0 && mid(c, d, b))
      return 1;
   if (sgn(cp1) != sgn(cp2) && sgn(cp3) != sgn(cp4))
      return 1;
   return 0;
}
signed main(){
   int t;
   cin >> t;
   while (t--){
      int x, y;
      P a, b, c, d;
      cin >> x >> y;
      a = {x, y};
      cin >> x >> y;
      b = {x, y};
      cin >> x >> y;
      c = {x, y};
      cin >> x >> y;
      d = {x, y};
      cout << (check(a, b, c, d) ? "YES" : "NO") << endl;
   }
}
```

4.6 Polygon Area(CSES)

/* Your task is to calculate the area of a given polygon. The polygon consists of n vertices (x1,y1),(x2,y2),...,(xn, yn). The vertices (xi,yi) and (xi+1,yi+1) are adjacent

for i=1,2,...,n1, and the vertices (x1,y1) and (xn,yn) are also adjacent. Input

The first input line has an integer n: the number of vertices. After this, there are n lines that describe the vertices. The ith such line has two integers xi and yi. You may assume that the polygon is simple, i.e., it does not

intersect itself. Output

Print one integer: 2a where the area of the polygon is a

(this ensures that the result is an integer). Constraints

* 3 <= n <= 1000

* -10^9 <= xi,yi <= 10^9

Example Input: 4

1 1

4 2

3 5

1 4

Example Output: 16 */

C++ Solution : #define int long long

```
#define X first
#define Y second
signed main()
{
   int n;
   cin >> n;
   pair<int, int> a[n];
   for (int i = 0; i < n; i++)
      cin >> a[i].X >> a[i].Y;
   int ans = 0;
   // shoelace formula
   for (int i = 0; i < n; i++)
   {
      ans += (a[i].X * a[(i + 1) % n].Y - a[(i + 1) % n].X * a[i].Y);
   }
   cout << abs(ans);
}
```

4.7 Polygon Lattice Points(CSES)

/* Given a polygon, your task is to calculate the number of

lattice points inside the polygon and on its boundary. A lattice point is a point whose coordinates are

integers. The polygon consists of n vertices (x1,y1),(x2,y2),...,(xn, yn). The vertices (xi,yi) and (xi+1,yi+1) are adjacent

for i=1,2,...,n1, and the vertices (x1,y1) and (xn,yn) are also adjacent. Input

The first input line has an integer n: the number of vertices. After this, there are n lines that describe the vertices. The ith such line has two integers xi and yi. You may assume that the polygon is simple, i.e., it does not

intersect itself. Output

Print two integers: the number of lattice points inside the polygon and on its boundary. Constraints

* 3 <= n <= 10^5

* -10^6 <= xi,yi <= 10^6

Example Input: 4

1 1

5 3

3 5

1 4

Example Output: 6 8 */

C++ Solution :

```
#define int long long
#define P complex<int>
#define X real()
#define Y imag()
signed main()
{
   int n;
   cin >> n;
   vector<P> v(n);
   for (int i = 0; i < n; i++)
   {
      int x, y;
      cin >> x >> y;
      v[i] = {x, y};
   }
   v.push_back(v[0]);
   int area = 0;
   int b = 0;
   for (int i = 0; i < n; i++)
   {
      P x = v[i], y = v[i + 1];
      area += (conj(x) * y).Y;
      P z = y - x;
      int g = gcd(z.X, z.Y);
      b += abs(g);
   }
   // 2*area = 2*a + b - 2
   int a = abs(area) - b + 2;
   cout << a / 2 < < < < b;
}
```

4.3 Minimum Euclidean Distance(CSES)

/ Given a set of points in the two-dimensional plane, your*

task is to find the minimum Euclidean distance between

two distinct points. The Euclidean distance of points (x1,y1) and (x2,y2) is sqrt ((x1-x2)^2 + (y1-y2)^2). Input

The first input line has an integer n: the number of points.

After this, there are n lines that describe the points. Eachline has two integers x and y. You may assume that each point is distinct. Output

Print one integer: d^2 where d is the minimumEuclideandistance (this ensures that the result is an integer). Constraints

** 2 <= n <= 2*10^5*

** -10^9 <= x,y <= 10^9*

Example Input: 4

2 1

4 4

1 2

6 3

*Example Output: 2 */*

C++ Solution :

```
#include <bits/stdc++.h>
using namespace std;

#define int long long
#define P pair<int, int>
#define X first
#define Y second

int norm(P a, P b) {
   return (b.X - a.X) * (b.X - a.X) + (b.Y - a.Y) * (b.Y - a.Y);
}

signed main() {
   int n;
   cin >> n;
   vector<P> v(n);
   for (int i = 0; i < n; i++) {
      int x, y;
      cin >> x >> y;
      v[i] = {x, y};
   }
   sort(v.begin(), v.end());  // sort by x-coordinate
   set<P> s = {{v[0].Y, v[0].X}};  // (y, x) so we can query by y
   int d = LLONG_MAX;
   int j = 0;

   for (int i = 1; i < n; i++) {
      int dd = sqrtl(d) + 1;

      while (j < i && v[j].X < v[i].X - dd) {
         s.erase({v[j].Y, v[j].X});
         j++;
      }
      auto l = s.lower_bound({v[i].Y - dd, -1e18});
      auto r = s.upper_bound({v[i].Y + dd, 1e18});

      for (auto it = l; it != r; ++it) {
         d = min(d, norm({it->Y, it->X}, v[i]));
      }
      s.insert({v[i].Y, v[i].X});
   }
   cout << d << "\n";
   return 0;
}
```