```cpp
vector<pi> node[705];
start = 1; finish = 2;
cin >> n >> tem;
node[start].push_back({finish, tem});

for(int ex = 3; ex <= n; ex++) {
    priority_queue<pi, vector<pi>, greater<pi>> pq;
    vector<int> dis(ex + 5, big);
    int m; cin >> m;
    mloop(j) {
        int v, d;
        cin >> v >> d;
        node[v].push_back({ex, d});
        node[ex].push_back({v, d});
    }

    dis[start] = 0;
    pq.push({dis[start], start});

    while(!pq.empty()) {
        int u = pq.top().se;
        int Dis = pq.top().fs;
        pq.pop();

        aloop(node[u]) {
            int v = itr.fs;
            int wei = itr.se;
            if( Dis + wei < dis[v]) {
                dis[v] = Dis + wei;
                pq.push({dis[v], v});
            }
        }
    }
    cout << dis[finish] << " ";
```

Dijk shortest path

```cpp
vector<NODE> edges;

void sol() {
    int n, m;
    cin >> n >> m >> start;
    mloop(i) {
        int x, y, w;
        cin >> x >> y >> w;
        edges.push_back({x,y,w});
    }

    dis[start] = 0;
    for(int i = 0; i < n - 1; ++i) {
        for(int j = 0; j < m; ++j) {
            int u = edges[j].i;
            int v = edges[j].j;
            int w = edges[j].wei;

            if(dis[u] != big && dis[u] + w < dis[v])
                dis[v] = dis[u] + w;
        }
    }
    //chk cycles
    mloop(i) {
        int u = edges[i].i;
        int v = edges[i].j;
        int w = edges[i].wei;
        bool chk =  dis[u] + w < dis[v];
        bool chk1 =  dis[u] != big;
        if(dis[u] + w != big && dis[u] + w < dis[v]){
            cout << -1;
            return ;
```

Bellman

```cpp
 sol() {
int n, m;
cin >> n >> m;
int mp[n + 1][m + 1], ans[n + 1][m + 1];
nloop(i)
    mloop(j)
        cin >> mp[i][j];


NODE start = {0,0,0};
vector<vector<int>> dis(n + 1, vector<int>(m + 1, big));
priority_queue<NODE, vector<NODE>, greater<NODE>> pq;
pq.push(start);
dis[start.u][start.v] = 0;

while(!pq.empty()) {
    int ux = pq.top().u;
    int uy = pq.top().v;
    int uw = pq.top().w;
    pq.pop();

    if(dis[ux][uy] < uw) continue;

    for(int k = 0; k < 4; ++k) {
        int vx = ux + d4x[k];
        int vy = uy + d4y[k];
        int wei = mp[vx][vy];
        if(vx >= 0 && vy >= 0 && vx < n &&
         vy < m && wei + dis[ux][uy] < dis[vx][vy]){

            dis[vx][vy] = wei + dis[ux][uy];
            pq.push({vx, vy, dis[vx][vy]});
```

dijk find min distance every point

```cpp
int check(int n){
    if(dis[n] == -1) return n;
    else {
        dis[n] = check(dis[n]);
        return dis[n];
    }
}

void sol() {
    int n;
    cin >> n;
    for(int i = 0; i < n - 1; ++i) {
        for(int j = i + 1; j < n ; ++j) {
            int x;
            cin >> x;
            node.push_back({x, {i, j}});
        }
    }
    // start
    sort(all(node));
    int ans = 0;
    aloop(node) {
        int u = check(itr.se.fs);
        int v = check(itr.se.se);

        if(u != v) {
            dis[u] = v;
            ans += itr.fs;
        }
    }
    cout << ans;
```

Prim disjointset

```cpp
void dfs(vector<vector<int>> &node,
    int u, int temp, vector<bool> &vis){
    if(vis[u]) {ck = true; return ;}
    vis[u] = true;
    aloop(node[u]) {
        if(itr == temp) continue;
        dfs(node, itr, u, vis);
    }
}

void sol() {
    int n;
    cin >> n;
    nloop(i) {
        int v, e;
        cin >> v >> e;
        vector<vector<int>> node(v, vector<int>());
        vector<bool> vis(v,0);

        for(int i=0; i<e; i++) {
            int x, y;
            cin >> x >> y;
            node[x].pb(y);
            node[y].pb(x);
        }

        ck = false;
        for(int i = 0; i < v; ++i) {
            if(!vis[i]) dfs(node, i, -1, vis);
        }


        if(ck) cout << "YES" << endl;
        else cout << "NO" << endl;
```

detect cycle

```cpp
void dfs(vector<vector<int>> &node, int u, int temp) {
    if(ck) return ;
    if(vis[u]){
        start = u, ck = 1;
        return ;
    }
    vis[u] = true;
    aloop(node[u]) {
        if(itr == temp) continue;
        dfs(node, itr, u);
    }
    if(start == u) {
        ans = cnt + 1;
        return ;
    }
    else if(ck) cnt++;

}

void sol() {
    int n;
    cin >> n;
    vector<vector<int>> node(n, vector<int>());
    nloop(i) {
        int x, y;
        cin >> x >> y;
        node[x].pb(y);
        node[y].pb(x);
    }
    nloop(i) {
        if(!vis[i])
            dfs(node, i, -1);
    }

    cout << ans;
```

max cycle

```cpp
void dfs(vector<vector<int>> &node, int level) {
    vis[level] = 1;
    aloop(node[level]) {
        if(!vis[itr]) {
            dfs(node, itr);
        }
    }
}

void sol() {
    cin >> n >> m;
    vector<vector<int>> node(n + 1, vector<int>());

    mloop(i) {
        int v,e;
        cin >> v >> e;
        node[v].push_back(e);
        node[e].push_back(v);
    }

    int cnt = 0;
    for(int i = 1; i < n + 1; i++){
        if(!vis[i]){
            dfs(node, i);
            cnt++;
        }
    }

    cout << cnt;
```

Connect compo

```cpp
int dfs(int u) {
    vis[u] = true;
    int dept = node[u].size();
    count_edge++;
    mxcnt = max(mxcnt, dept);
    aloop(node[u]) {
        if(!vis[itr])
            dept += dfs(itr);
    }
    return dept;
}

void sol() {
    int n, m;
    cin >> n >> m;
    mloop(i) {
        int v,e;
        cin >> v >> e;
        node[v].push_back(e);
        node[e].push_back(v);
    }

    int cnt = 0;
    for(int i = 0; i < n; i++) {
        mxcnt = 0;
        count_edge = 0;
        if(!vis[i]) {
            int temp = dfs(i) / 2;
            if(temp == count_edge - 1 && mxcnt < 3) cnt++;
        }
    }
    cout << cnt;
```

กราฟเส้นตรง

```cpp
    int v, e, k, ans = 0;
    vector<vector<int>> node(big, vector<int>());
    cin >> v >> e >> k;
    for(int i = 0; i < e; ++i) {
        int n, m;
        cin >> n >> m;
        node[n].push_back(m);
        node[m].push_back(n);
    }
    for(int i = 0; i < v; ++i) {
        vector<int> dis(big, 0);
        vector<bool> vis(big,0);
        queue<int> q;
        int cnt = 1;
        q.push(i);
        vis[i] = true; // visited
        while(!q.empty()) {
            int u = q.front();
            q.pop();
            if(dis[u] == k) break;
            aloop(node[u]) {
                if(!vis[itr]){
                    vis[itr] = true;
                    dis[itr] = dis[u] + 1;
                    q.push(itr);
                    cnt++;
                }
            }
        }
        ans = max(ans, cnt);
    }

    cout << ans;
```

bfs connection k level

```cpp
int start,finish;
priority_queue<pi, vector<pi>, greater<pi>> pq;
vector<int> d8x = {1, -1, 0, 0, 1, -1, 1, -1};
vector<int> d8y = {0, 0, 1, -1, 1, -1, -1, 1};
vector<int> d4x = {0, 1, 0, -1}; // row n
vector<int> d4y = {1, 0, -1, 0}; // column m

vector<pi> node2[big];
vector<int> dis(big,0);
vector<bool> vis(big,0);
```

```cpp
struct NODE {
    int u, v, w;

    bool operator>(const NODE& other) const {
     return w > other.w;
    }
};
```

Use NODE when bfs

Plague 4 direction and get for t day ->

```cpp
 sol() {
int n, m, t, cnt = 0;
cin >> n >> m >> t;
int node[n + 5][m + 5];
queue<NODE> q;
nloop(i) {
    mloop(j) {
        cin >> node[i][j];
        if(node[i][j] == 1)
            q.push({i, j, 0});
    }
}

while(!q.empty()){
    int ux = q.front().u;
    int uy = q.front().v;
    int day = q.front().ck;
    q.pop();
    if(day > t) break;

    for(int i = 0; i < 4; ++i) {
        int nx = ux + d4x[i];
        int ny = uy + d4y[i];
        if(nx >= 0 && ny >= 0 && nx < n && ny < m && node[nx][ny] == 0) {
            node[nx][ny] = 1;
            q.push({nx, ny, day + 1});
        }
    }
    cnt++;
}
cout << cnt;
```

```cpp
const int INF = 1e9+1;
ll n,m,pa[BIG],sum = 0
int dis[BIG];

int F(int u){
  if(pa[u] == u) return u;
  return pa[u] = F(pa[u]);
}
//int merge(int u,int v);

struct edge{
  int u,v,w;
  bool operator < (edge &compare)const{
    return w < compare.w;
  }
};

vector<edge> E;

void kruskals(){
  sort(E.begin(),E.end());
  for(auto itr : E){
    if(F(itr.u) != F(itr.v)){
      pa[F(itr.v)] = F(itr.u);
      sum += itr.w;
    }
  }
  cout << sum;
```

Krus algo

```cpp
void sol() {
    ll n, m;
    cin >> n >> m;
    vector<ll> temp(m), dp(n + 1, 0);
    mloop(i)
        cin >> temp[i];

    sort(all(temp)); // optimize

    // base case
    dp[0] = 1;
    for(int i = 1; i <= n; i++) {
        for(int j = 0; j < m; j++) {
            if(i >= temp[j]){
                dp[i] = (dp[i] + dp[i - temp[j]]) % big
            }
        }
    }

    cout << dp[n];
}
```

Coin change prob

```cpp
int n,ans=INT_MIN;
cin>>n;
int arr[n+1][n+1]={0};
for (int i=1;i<=n;i++){
    for (int j=1;j<=n;j++){
        int num;
        cin>>num;
        arr[i][j]=arr[i-1][j-1]+num;

        for (int k=1;k<=i && k<=j;k++){
            ans = max(ans,arr[i][j]-arr[i-k][j-k]);
        }
    }
}

ut<<ans;
```

บวกเมตริก แนวทแยง

```cpp
const int INF = 1e9;
for (int i = 1; i <= n; ++i)
    dp[i][0] = -INF;
for (int j = 1; j <= m; ++j)
    dp[0][j] = -INF;
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= m; ++j) {
        if (i == 1 && j == 1)
            dp[i][j] = A[i][j];
        else
            dp[i][j] = max(dp[i-1][j], dp[i][j-1]) + A[i][j];
    }
}
// answer: dp[n][m]
```

Move in table top left to bot right

```cpp
int lis(vector<int>& nums) {
    int n = nums.size();
    vector<int> dp(n, 1);

    for (int i = 1; i < n; ++i) {
        for (int j = 0; j < i; ++j) {
            if (nums[i] > nums[j] && dp[i] < dp[j] + 1) {
                dp[i] = dp[j] + 1;
            }
        }
    }

    return *max_element(dp.begin(), dp.end());
```

LIS problem

```cpp
void sol() {
    ll n;
    cin >> n;
    ++n;
    ll dp[n];
    memset(dp, 0, n * 8);
    // aloop(dp) cout << itr << endl;
    for(int i=0; i<=n; ++i){
        if(i == 0) dp[i] = 1;
        else if(i == 1) dp[i] = 3;
        else dp[i] = ((dp[i - 1] * 2) + dp[i - 2]) % INF;
    }
    cout << dp[n - 1];
}
```

Combi Table 11 (00, 01, 10)

```cpp
pair<int, pair<int, int>> maxSubarraySum(vector<int>& A) {
    int n = A.size();
    int maxSum = A[0];
    int currentSum = A[0];
    int start = 1;
    int end = 1;
    int tempStart = 1;
    for (int i = 1; i < n; ++i) {
        if (currentSum < 0) {
            currentSum = A[i];
            tempStart = i + 1;
        } else {
            currentSum += A[i];
        }
        if (currentSum > maxSum) {
            maxSum = currentSum;
            start = tempStart;
            end = i + 1;
        }
    }
    return {maxSum, {start, end}};
}
int main() {
    vector<int> A = {-2, 1, -3, 4, -1, 2, 1, -5, 4};
    auto result = maxSubarraySum(A);
    cout << "Maximum Subarray Sum: " << result.first << endl
    cout << "Start Index: " << result.second.first << endl;
    cout << "End Index: " << result.second.second << endl;
```

Max sub array

```cpp
int longestCommonSubsequence(const string& str1, const string& str2)
    int m = str1.length();
    int n = str2.length();

    // Create a 2D table to store the solutions to subproblems
    vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

    // Build the table bottom-up
    for (int i = 1; i <= m; ++i) {
        for (int j = 1; j <= n; ++j) {
            if (str1[i - 1] == str2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1; // Characters match
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]); // Char
            }
        }
    }

    return dp[m][n]; // Length of the LCS
```

Longest common sub seq

```cpp
int knapsack(int W, vector<int>& weight, vector<int>& value) {
    int n = weight.size();
    vector<vector<int>> dp(n + 1, vector<int>(W + 1, 0));    ider

    for (int i = 1; i <= n; ++i) {
        for (int w = 1; w <= W; ++w) {
            if (weight[i - 1] <= w) {
                dp[i][w] = max(dp[i - 1][w],     identifier "max"
                    dp[i - 1][w - weight[i - 1]] + value[i - 1]);
            } else {
                dp[i][w] = dp[i - 1][w];
            }
        }
    }

    return dp[n][W];
```

Knapsack 0 1

```cpp
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    }
    vector<int> dp(n + 1, 0);
    dp[0] = 0;
    dp[1] = 1;
    for (int i = 2; i <= n; ++i) {
        dp[i] = dp[i - 1] + dp[i - 2];
    }
    return dp[n];
```

```cpp
bool isPalindrome(const string &str) {
    for (size_t i = 0, j = str.length() - 1; i < j; ++i, --j) {
        while (i < j && !isalnum(str[i])) ++i;
        while (i < j && !isalnum(str[j])) --j;
        if (tolower(str[i]) != tolower(str[j])) return false;
    }
    return true;
}
```

```cpp
bool isPrime(int n) {
    if (n <= 1) return false;
    if (n <= 3) return true;
    if (n % 2 == 0 || n % 3 == 0) return false;
    for (int i = 5; i * i <= n; i += 6)
        if (n % i == 0 || n % (i + 2) == 0) return false;
    return true;
}
```

```
void cal(vector<string> &v, string temp, int num, int n, int m, bool chk) {
    if(num == n) { // counter complete
        if(cnt == m || chk) cout << temp << endl;
    }
    else {
        if(cnt == m) chk = true; // complete bits
        num++;
        cnt++;

        cal(v, temp + '0', num, 0, n, m, chk);
        cal(v, temp + '1', num, cnt, n, m, chk);
    }


void sol() {
    int n ,m, cnt = 0;
    string temp = "";
    cin >> n >> m;
    vector<string> v;
    cal(v, temp, 0, cnt, n, m, false);
```

Permu with m '1'

```
void cal(vector<int> &v, vector<int> &vm, int cnt, int n, vector<
    if(cnt == n) {
        aloop(v)
            cout << itr << ' ';
        cout << endl;
        return ;
    }

    for(int i = 0; i < n; i++) {
        if(chk[i] == false && (vm[i] == - 1 || chk[vm[i]] == true)
            // check element is not already used and no constrain
            v.push_back(i);
            chk[i] = true;
            cal(v, vm, cnt + 1, n, chk);
            v.pop_back(); // back tracking
            chk[i] = false;
        }
    }
}

void sol() {
    int n ,m, cnt = 0;
    cin >> n >> m;
    vector<int> vm(n, -1);
    mloop(i) {
        int x, y;
        cin >> x >> y;
        vm[y] = x;

    ector<int> v;
    ector<bool> chk(n, false);
    al(v, vm, cnt, n, chk);
```

Permu constrant

```
d sol() {
 ll n, m;
 cin >> n >> m;
 vector<ll> seat(n);
 nloop(i) cin >> seat[i];
 mloop(i) {
     ll cusq, ans = 0, l = 0, r = INF, qnow;
     cin >> cusq;
     while(l <= r) {
         qnow = 0;
         ll mid = (l + r) / 2;
         aloop(seat) qnow += (mid / itr) + 1; // count the serve
         if(qnow >= cusq){ // check customer can be seat??
             ans = mid;
             r = mid - 1; // move range to find min time
         }
         else l = mid + 1; // move range to find min time
     }
     cout << ans << endl;
 }
```

Bst check fit queue

(ภาพล่าง) greedy fr knapsack

```
bool cmp(pair<db, db> x, pair<db, db> y) {
    db temp1 = (db)x.fs / (db)x.se;
    db temp2 = (db)y.fs / (db)y.se;
    return temp1 > temp2;
}

void sol() {
    db w,n,x;
    cin >> w >> n;
    vector<db> val;
    vector<pair<db, db>> wei;
    db nowwei = 0, ans = 0.0;
    nloop(i) {
        cin >> x; val.pb(x);
    }
    nloop(i) {
        cin >> x; wei.push_back({val[i], x});
    }

    sort(all(wei), cmp);

    aloop(wei){
        if(nowwei + itr.se <= w) {
            nowwei += itr.se;
            ans += itr.fs;
        }
        else {
            db tem = w - nowwei;
            ans += itr.fs*((db)tem / (db)itr.se);
            break;
        }
    }
```

```
int A[big];

void sol() {
    int n, m, k;
    cin >> n >> m >> k;
    for(int i = 1 ;i <= n ;i++){
        cin >> A[i];
        A[i] += A[i - 1];
        // cout << A[i] << " ";
    }
    mloop(i) {
        int L, r = n, ans = 0, money;
        cin >> L >> money;
        L++;
        int l = L;
        while(l <= r) {
            int mid = (l + r) / 2;
            if(A[mid] - A[L - 1] + (mid - L + 1) * k <= money) {
                l = mid + 1 ,ans = A[mid] - A[L - 1];
            }
            else r = mid - 1;
        }

        cout << ans << '\n';
    }
```

Money fill ในช่วง(เก็บช่วงแรก ลบค่าช่วงหลัง)

Sweepline ->

```
struct Sweep_Line {
    int r,val;
    bool operator<(const Sweep_Line &joox) const {
        if(r==joox.r) return val<joox.val;
        return r<joox.r;
    }
};
vector <Sweep_Line> line;
vector <pair<int,int>> res;

void process(void) {
    int n,k,last,sum=0;
    bool check=false;
    cin >> n >> k;
    for(int i=1,x,y,z;i<=n;i++) {
        cin >> x >> y >> z;
        line.push_back({x,z});
        line.push_back({y+1,-z});
    }
    sort(line.begin(),line.end());
    for(auto v:line) {
        sum+=v.val;
        if(sum<=k) {
            if(check) res.emplace_back(last,v.r-1);
            check=false;
        }else if(!check) {
            last=v.r;
            check=true;
        }
    }
    for(auto v:res) {
        cout << v.st << ' ' << v.nd;
        cout << '\n';
    }
```