

C	Vacation	Chọn 1 trong 3 việc (A,B,C) i làm việc j	dp[i][j] = max _{w,j} (dp[i - 1][k]) + cost[i][j]	L	Deque	Game bài từ đầu, Max hóa (Điểm mới - Điểm cũ)	dp[i][j]; Hiệu điểm max doan[i][j]	max(a[i] - dp[i + 1][j], a[j] - dp[i][j - 1])	T	Permutation	Đếm hoán vị dài N thửa mãn chẵn đầu < và >	dp[i][j]; Hoán vị dài i, cuối là số lẻ thứ j	Tối ưu chuyển trạng thái bằng Prefix Sum O(N ²)
D	Knapsack 1	Cài Khi Chọn vớt $\sum w \leq W$ để max $\sum v_i$ (trong trọng w)	dp[w] = max(dp[w], dp[w - w _i] + v _i)	M	Candies	Chia K kéo cho N tré, trả i nhán tối da a, Đếm sô cách	dp[i][j]; Chia cho i người, tống j	Dùng Prefix Sum: dp[i][j] = sum dp[i][...]	U	Grouping	Chia N vật thành các nhóm, Max tổng điểm của các nhóm.	dp[mask] = max(dp[mask \ S] + score(S)) (O(3 ^N))	
E	Knapsack 2	Göng bá D Chóng $W \leq 10^3$, $v \leq 10^3$, đếm giá trị v	dp[v] = min(dp[v], dp[v - v _i] + w _i)	N	Slimes	Gộp 2 cusc slime kế nhau, phai tống kích thước Min phai	dp[i][j]; Phí gấp đoạn	min(a[i] - dp[i][k] + dp[k + 1][j]) + sum(i, j)	V	Subtree	Với mỗi đỉnh, đếm số cách tò mây cầu có liên thông chia nô (mod 2)	dp[i]: Kết quả đỉnh u	Rooting DP: Tính Down xong tính Up.
F	LCS	Tìm dài của chuỗi dài nhất của 2 chuỗi a và t.	dp[i][j]; Độ dài sâu chung[i][j], dp[i - 1][j - 1]	O	Matching	Đếm số cách ghép cặp N nam - N新娘 theo bảng tương thích	dp[mask]; Số cách ghép cặp (Bitmap)	sum(dp[mask \ \{j\}) nếu bit j match với bit cao nhất	W	Intervals	Xử lý phiền. Các đoạn [i, r] cho đỉnh a, nhén toàn số 1. Max điểm	dp[i]; Điểm max tại i	SegTree + Lazy (Range Add, Range Max).
G	Longest Path	Tìm đường dài nhất trong đồ thị có hướng (DAG).	dp[i]; Đường dài nhất bắt đầu từ i	P	Independent	Đếm cây tò mò sau k cho 2 Đen k nhau.	dp[i][0]; Cây con u, u trắng	Nhận quy tắc: Trắng (con tò mò). Đen (con phái trắng)	X	Tower	Xếp chồng khối (nhỏ u, cùng x, giá trị v). Max tổng giá trị.	Knapsack 1 chiều	Sort theo w + x, sau đó DP cả từ nhỏ bài D/E.
H	Grid 1	Đếm cách từ (1,1) → (B, W), tránh tường	dp[i][j]; Số cách đến (i, j)	Q	Flowers	Tìm số cây có chiều cao tăng dần cho tổng giá trị max.	dp[i]; Max trọng số kết thúc tại i	max(a[i], dp[i] + a _i). Dùng Segment Tree lấy Max.	Y	Grid 2	Lưới $10^3 \times 10^6$, có N vật cần. Đếm cách di tránh vật cản.	dp[i]; Cách di chuyển i	Bao hàm loại trừ: Tổng đường - đường đi qua vật cản khác.
I	Coin	N vòi xác suất ngẫu p _i . Tính xác suất số ngẫu> sấp	dp[i][j]; Tung i xu, j mặt	R	Walk	Đếm số đường đi dài dùng K trong đồ thị hướng.	Ma trận M^K	Tổng các phần tử của M^K (Nhận ma trận)	Z	Frog 3	Nhảy i → j, pH $(h_i - h_j)^2 + C$. Tính phi min. $N = 2 \cdot 10^5$.	dp[i]; Chi phí tối thiểu	Đang y = mx + c. Dù vector<vector<int>> course_lists; vector<vector<int>> conflicts; vector<int> assigns; vector<int> loads;
J	Sushi	N đồ ăn 1 mảng. Chọn kg/nhìn để ăn. Thêm kỹ quyết	dp[i][j]; Ký vọng khi còn i, j, K loại 1,2,3	S	Digit Sum	Đếm số lượng số a ∈ [1, K] có tổng chữ số chia hết cho D.	dp[pos][right][rem]	Digit DP đếm số thỏa mãn mod D = 0				int check(int course, int teacher)	Balance courses
K	Stones	Đóng đá K. Bốc a, viên Al không bốc được thì thu.	dp[i]; Thủ Thuật									void assign(int course)	if (assigns[course] == -1)
	CBUS											for (int teacher = 0; teacher < m; teacher++)	if (check(course, teacher))
	CVRP											assigns[course] = teacher;	loads[teacher]++;
												if (course == (n - 1))	{ MIN = min(MIN,
												*max_element(loads.begin(), loads.end()));	}
												else	{ int MAX = *max_element(loads.begin(), loads.end());
												if (MAX <= MIN) assign(course + 1);	}
												loads[teacher]--;	assigns[course] = -1;
												}	You, 3 days ago • Create balanced_courses_assignment
												int main()	{
												cin >> m >> n;	course_lists.resize(m, vector<int>());
												for (int i = 0; i < m; i++)	{
												int k;	int course;
												cin >> course;	course_lists[i].push_back(course - 1);
												}	
												int k;	conflicts.resize(n, vector<int>(n, 0));
												while (k--)	{
												int i, j;	cin >> i >> j;
												conflicts[i - 1][j - 1] = 1;	conflicts[j - 1][i - 1] = 1;
												}	assigns.resize(n, -1);
												loads.resize(m, 0);	assign(0);
												cout << MIN << "\n";	return a.
												int backtrack(int k)	{
												if (k == n + 1) return 1;	
												int h = recs[k][0];	int w = recs[k][1];
												for (int i = 1; i <= H; i++)	{
												for (int j = 1; j <= W; j++)	{
												if (canPlace(h, w, i, j))	{ place(h, w, i, j, 1);
												if (backtrack(k + 1)) return 1;	place(h, w, i, j, 0);
												}	if (h != w && canPlace(w, h, i, j))
												{ place(w, h, i, j, 1);	if (backtrack(k + 1)) return 1;
												}	place(w, h, i, j, 0);
												}	return 0;
												int main()	{
												cin >> H >> W;	
												cin >> n;	
												int area = 0;	for (int i = 1; i <= n; i++)
												cin >> recs[i][0] >> recs[i][1];	
												area += (recs[i][0] * recs[i][1]);	
												if (area > H * W)	
												{ cout << 0 << "\n";	
												return 0;	
												sort(recs.begin() + 1, recs.end(), [](vector<int> a, vector<int> b) { return a[0] * a[1] > b[0] * b[1];});	
												cout << backtrack(1) << "\n";	

Bus-inter-city (BFS to build weighted directed graph and dijkstra)

```
int n, m;
vector<int> c, d, visited, dp;
vector<vector<int>> adj;
vector<long long> dis;
vector<vector<pair<int, int>> adj1;
void bfs(int i, int limit)
{
    fill(visited.begin(), visited.end(), 0);
    queue<pair<int, int>> q;
    visited[i] = 1;
    while (!q.empty())
    {
        int u = q.front().first;
        int depth = q.front().second;
        q.pop();
        if (u != i)
        {
            adj1[i].push_back({u, c[i]});
        }
        if (depth >= limit) continue;

        for (auto v : adj[u])
        {
            if (!visited[v])
            {
                visited[v] = 1;
                q.push({v, depth + 1});
            }
        }
    }
}
void dijkstra(int s)
{
    priority_queue<pair<long long, int>, vector<pair<long long, int>>, greater<pair<long long, int>> pq;
    dis[s] = 0;
    pq.push({0, s});

    for (int i = 0; i < n; i++)
    {
        int u = -1;

        while (!pq.empty())
        {
            u = pq.top().second;
            pq.pop();
            if (visited[u]) break;
        }
        if (u == -1 || dis[u] == INT_MAX) break;
        visited[u] = 1;
        for (auto edge : adj1[u])
        {
            int v = edge.first;
            int w = edge.second;

            if (dis[u] + w < dis[v])
            {
                dis[v] = dis[u] + w;
                pq.push({dis[v], v});
            }
        }
    }
    int main()
    {
        cin >> n >> m;
        c.resize(n + 1, 0);
        d.resize(n + 1, 0);
        dp.resize(n + 1, 0);
        visited.resize(n + 1, 0);
        adj1.resize(n + 1, vector<int>());
        adj1.resize(n + 1, vector<pair<int, int>>());
        dis.resize(n + 1, INT_MAX);
        for (int i = 1; i <= n; i++)
        {
            cin >> c[i] >> d[i];
        }
        for (int i = 1; i <= m; i++)
        {
            int u, v;
            cin >> u >> v;
            adj1[u].push_back(v);
            adj1[v].push_back(u);
        }
        for (int i = 1; i <= n; i++)
        {
            bfs(i, d[i]);
        }
        fill(visited.begin(), visited.end(), 0);
        dijkstra(1);
        cout << dis[n] << "\n";
        return 0;
    }
}

amount.resize(n, vector<int>(X + 1, INT_MAX - 1));
for (int i = 0; i < n; i++) amount[i][0] = 0;
//base case i = 0, Money-exchange
//knapsack inf
for (int j = 1; j <= X; j++)
{
    if ((j / d[0]) * d[0] == j)
    {
        amount[0][j] = j / d[0];
    }
}
for (int i = 1; i < n; i++)
{
    for (int j = 1; j <= X; j++)
    {
        amount[i][j] = amount[i - 1][j];
        if (j == d[i])
        {
            amount[i][j] = min(amount[i][j],
                                amount[i][j - d[i]] + 1);
        }
    }
}
cout << (amount[n - 1][X] != INT_MAX - 1 ?
         amount[n - 1][X] : -1) << "\n";
return 0;
```

Make-span-schedule (topo sort and DP)

```
int n, m;
vector<vector<int>> adj;
vector<int> d;
vector<int> visited;
vector<int> order;
int hasCycle = 0;
int ans = 0;
vector<int> f;
void topo(int u)
{
    if (visited[u] == 2) return; // done
    if (visited[u] == 1)
    {
        hasCycle = 1;
        // cycle
        return;
    }
    visited[u] = 1;
    for (auto v : adj[u])
    {
        topo(v);
        if (hasCycle) return;
    }
    visited[u] = 2;
    order.push_back(u);
}
int main()
{
    cin >> n >> m;
    d.resize(n + 1, 0);
    adj.resize(n + 1, vector<int>());
    for (int i = 1; i <= n; i++)
    {
        cin >> d[i];
    }
    adj.resize(n + 1, vector<int>());
    visited.resize(n + 1, 0);
    for (int i = 1; i <= n; i++)
    {
        if (visited[i]) topo(i);
    }
    reverse(order.begin(), order.end());
    for (auto u : order)
    {
        int v = edge.first;
        int w = edge.second;
        if (dis[u] + w < dis[v])
        {
            dis[v] = dis[u] + w;
            pq.push({dis[v], v});
        }
    }
}
int main()
{
    cin >> n >> m;
    vector<vector<int>> adj;
    // adj[task] = list of staffs
    // that can do this task
    vector<int> matchTask;
    // matchTask[staff] = task
    // assigned to this staff (0 if none)
    adj.resize(n + 1, INT_MAX);
    for (int i = 1; i <= n; i++)
    {
        cin >> c[i] >> d[i];
    }
    for (int i = 1; i <= m; i++)
    {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    for (int i = 1; i <= n; i++)
    {
        bfs(i, d[i]);
    }
    fill(visited.begin(), visited.end(), 0);
    dijkstra(1);
    cout << dis[n] << "\n";
    return 0;
}

amount.resize(n, vector<int>(X + 1, INT_MAX - 1));
for (int i = 0; i < n; i++) amount[i][0] = 0;
//base case i = 0, Money-exchange
//knapsack inf
for (int j = 1; j <= X; j++)
{
    if ((j / d[0]) * d[0] == j)
    {
        amount[0][j] = j / d[0];
    }
}
for (int i = 1; i < n; i++)
{
    for (int j = 1; j <= X; j++)
    {
        amount[i][j] = amount[i - 1][j];
        if (j == d[i])
        {
            amount[i][j] = min(amount[i][j],
                                amount[i][j - d[i]] + 1);
        }
    }
}
cout << (amount[n - 1][X] != INT_MAX - 1 ?
         amount[n - 1][X] : -1) << "\n";
return 0;
```

SCC

```
int n, m;
vector<vector<int>> adj;
vector<int> num;
vector<int> low;
stack<int> SCC;
vector<int> onStack;
int current_num = 0, cnt = 0;
vector<vector<int>> res;
void dfs(int u)
{
    current_num++;
    num[u] = current_num;
    low[u] = current_num;
    SCC.push(u);
    onStack[u] = 1;
    for (auto v : adj[u])
    {
        if (num[v] == 0)
        {
            dfs(v);
            low[u] = min(low[u], low[v]);
        }
        else if (onStack[v] == 1)
        {
            low[u] = min(low[u], num[v]);
        }
    }
    if (low[u] == num[u])
    {
        cnt++;
        vector<int> res1;
        while (!SCC.empty())
        {
            int v = SCC.top();
            SCC.pop();
            onStack[v] = false;
            res1.push_back(v);
            if (v == u)
            {
                sort(res1.begin(), res1.end());
                res.push_back(res1);
                break;
            }
        }
    }
}
int main()
{
    if (!!(cin >> n >> m >> k)) return 0;

    sz = max(n, m);
    a.assign(sz + 1, vector<long long>(sz + 1, 0));
    for (int i = 0; i < k; i++)
    {
        int x, y, w;
        cin >> x >> y >> w;
        if (a[x][y] == 0) a[x][y] = -w;
        else a[x][y] = min(a[x][y], (long long)-w);
    }

    u.assign(sz + 1, 0);
    v.assign(sz + 1, 0);
    p.assign(sz + 1, 0);
    way.assign(sz + 1, 0);
    for (int i = 1; i <= sz; i++)
    {
        p[i] = i;
        int j0 = 0;
        minv.assign(sz + 1, INF);
        vector<bool> used(sz + 1, false);
        do
        {
            used[j0] = true;
            int i0 = p[j0];
            long long delta = INF;
            int j1;
            for (int j = 1; j <= sz; j++)
            {
                if (!used[j])
                {
                    if (v == u)
                    {
                        long long cur = a[i0][j] - u[i0] - v[j];
                        if (cur < minv[j])
                        {
                            minv[j] = cur;
                            way[j] = j0;
                        }
                    }
                    if (minv[j] < delta)
                    {
                        delta = minv[j];
                        j1 = j;
                    }
                }
            }
            j0 = j1;
        } while (j0 != 0);
        cout << p[0] << "\n";
        for (int j = 0; j <= sz; j++)
        {
            if (used[j])
            {
                u[p[j]] += delta;
                v[j] -= delta;
            }
            else
            {
                minv[j] -= delta;
            }
        }
        j0 = j1;
        while (p[j0] != 0);
        do
        {
            int j1 = way[j0];
            p[j0] = p[j1];
            j0 = j1;
        } while (j0 != 0);
        cout << v[0] << "\n";
    }
    return 0;
}
```

Hungary

```
const long long INF = 1e18; // Giá trị vô cùng lớn cho long long

int n, m, k;
vector<vector<int>> adj;
vector<int> num;
vector<int> low;
vector<long long> u, v, minv;
vector<int> p, way; // p: mang matching, way: mang truy vết

int main()
{
    if (!!(cin >> n >> m >> k)) return 0;

    sz = max(n, m);
    a.assign(sz + 1, vector<long long>(sz + 1, 0));
    for (int i = 0; i < k; i++)
    {
        int x, y, w;
        cin >> x >> y >> w;
        if (a[x][y] == 0) a[x][y] = -w;
        else a[x][y] = min(a[x][y], (long long)-w);
    }

    u.assign(sz + 1, 0);
    v.assign(sz + 1, 0);
    p.assign(sz + 1, 0);
    way.assign(sz + 1, 0);
    for (int i = 1; i <= sz; i++)
    {
        p[i] = i;
        int j0 = 0;
        minv.assign(sz + 1, INF);
        vector<bool> used(sz + 1, false);
        do
        {
            used[j0] = true;
            int i0 = p[j0];
            long long delta = INF;
            int j1;
            for (int j = 1; j <= sz; j++)
            {
                if (!used[j])
                {
                    if (v == u)
                    {
                        long long cur = a[i0][j] - u[i0] - v[j];
                        if (cur < minv[j])
                        {
                            minv[j] = cur;
                            way[j] = j0;
                        }
                    }
                    if (minv[j] < delta)
                    {
                        delta = minv[j];
                        j1 = j;
                    }
                }
            }
            j0 = j1;
        } while (j0 != 0);
        cout << p[0] << "\n";
        for (int j = 0; j <= sz; j++)
        {
            if (used[j])
            {
                u[p[j]] += delta;
                v[j] -= delta;
            }
            else
            {
                minv[j] -= delta;
            }
        }
        j0 = j1;
        while (p[j0] != 0);
        do
        {
            int j1 = way[j0];
            p[j0] = p[j1];
            j0 = j1;
        } while (j0 != 0);
        cout << v[0] << "\n";
    }
    return 0;
}
```

Edmond-karp (max flow)

```
int n, m;
vector<vector<int>> capacity;
vector<vector<int>> adj;
int bfs(int source, int sink, vector<int>& parent)
{
    fill(parent.begin(), parent.end(), -1);
    parent[source] = source;
    queue<pair<int, int>> q;
    q.push({source, INT_MAX});
    while (!q.empty())
    {
        int u = q.front().first;
        int flow = q.front().second;
        q.pop();
        for (int v : adj[u])
        {
            if (parent[v] == -1 && capacity[u][v] > 0)
            {
                parent[v] = u;
                int new_flow = min(flow, capacity[u][v]);
                q.push({v, new_flow});
            }
        }
    }
    return 0;
}

int edmond_karp(int source, int sink)
{
    int max_flow = 0;
    vector<int> parent(n + 1);
    int new_flow;
    while ((new_flow = bfs(source, sink, parent)) > 0)
    {
        max_flow += new_flow;
        int cur = sink;
        while (cur != source)
        {
            int prev = parent[cur];
            capacity[prev][cur] -= new_flow;
            capacity[cur][prev] += new_flow;
            cur = prev;
        }
    }
    return max_flow;
}

int main()
{
    cin >> n >> m;
    capacity.resize(n + 1, vector<int>(n + 1, 0));
    adj.resize(n + 1);
    for (int i = 1; i <= m; i++)
    {
        int u, v, c;
        cin >> u >> v >> c;
        capacity[u][v] += c;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    int source, sink;
    cin >> source >> sink;
    cout << edmond_karp(source, sink) << "\n";
    return 0;
}
```

Max-matching-bipartite

```
int n, m;
vector<vector<int>> adj;
// adj[task] = list of staffs
// that can do this task
vector<int> matchTask;
// matchTask[staff] = task
// assigned to this staff (0 if none)
adj.resize(n + 1, INT_MAX);
for (int i = 1; i <= n; i++)
{
    cin >> c[i] >> d[i];
}
for (int i = 1; i <= m; i++)
{
    int u, v;
    cin >> u >> v;
    adj[u].push_back(v);
    adj[v].push_back(u);
}
for (int i = 1; i <= n; i++)
{
    bfs(i, d[i]);
}
fill(visited.begin(), visited.end(), 0);
dijkstra(1);
cout << dis[n] << "\n";
return 0;
}

amount.resize(n, vector<int>(X + 1, INT_MAX - 1));
for (int i = 0; i < n; i++) amount[i][0] = 0;
//base case i = 0, Money-exchange
//knapsack inf
for (int j = 1; j <= X; j++)
{
    if ((j / d[0]) * d[0] == j)
    {
        amount[0][j] = j / d[0];
    }
}
for (int i = 1; i < n; i++)
{
    for (int j = 1; j <= X; j++)
    {
        amount[i][j] = amount[i - 1][j];
        if (j == d[i])
        {
            amount[i][j] = min(amount[i][j],
                                amount[i][j - d[i]] + 1);
        }
    }
}
cout << (amount[n - 1][X] != INT_MAX - 1 ?
         amount[n - 1][X] : -1) << "\n";
return 0;
```

Total-path-lengths

```
int n;
vector<vector<pair<int, int>> a(MAX + 1);
vector<int> subtree_size(MAX + 1, 1);
vector<int> visited(MAX + 1, 0);
vector<long long> answer(MAX + 1, 0);

void dfs(int u, int depth)
{
    visited[u] = 1;
    answer[1] += depth;
    for (auto path : a[u])
    {
        int v = path.first;
        int w = path.second;
        if (visited[v] == 0)
        {
            dfs(v, depth + subtree_size[v] * w);
            subtree_size[u] += subtree_size[v];
        }
    }
}

void dfs2(int u)
{
    visited[u] = 1;
    for (auto path : a[u])
    {
        int v = path.first;
        int w = path.second;
        if (visited[v] == 0)
        {
            answer[v] = answer[u] - w * subtree_size[v] * w;
            dfs2(v);
        }
    }
}

int main()
{
    cin >> n;
    for (int i = 1; i < n; i++)
    {
        int u, v, w;
        cin >> u >> v >> w;
        a[u].push_back({v, w});
        a[v].push_back({u, w});
    }
    dfs(1, 0);
    for (int i = 1; i <= n; i++)
    {
        cout << answer[i] << "\n";
    }
    return 0;
}
```

Longest-path

```
int n;
vector<int> visited(MAX + 1, 0);
vector<vector<int>> a(MAX + 1);
vector<int> distances(MAX + 1);

void bfs(int i)
{
    queue<int> q;
    if (visited[i] == 1) return;
    q.push(i);
    visited[i] = 1;
    while (!q.empty())
    {
        int u = q.front();
        q.pop();
        for (auto path : a[u])
        {
            int v = path.first;
            if (visited[v] == 0)
            {
                q.push(v);
                visited[v] = 1;
                distances[v] = distances[u] + w;
            }
        }
    }
}

int main()
{
    cin >> n;
    for (int i = 1; i < n; i++)
    {
        int u, v, w;
        cin >> u >> v >> w;
        a[u].push_back({v, w});
        a[v].push_back({u, w});
    }
    bfs(1);
    int Max = 0;
    int start = 1;
    for (int i = 1; i <= n; i++)
    {
        if (Max < distances[i])
        {
            Max = distances[i];
            start = i;
        }
    }
    cout << Max << "\n";
    return 0;
}
```