Week 24: Dynamic Programming on Graphs

Topics: - Shortest Path Algorithms: Dijkstra, Bellman-Ford, Floyd-Warshall - DP on DAGs (Longest Path, Counting Paths) - Traveling Salesman Problem (TSP) with Bitmask DP - DP on Trees (Subtree DP, Tree Diameter, Path DP) - DP with Memoization on Graphs

Weekly Tips: - DP on DAGs: topological sorting ensures correct DP order. - TSP Bitmask DP is suitable for small n (<=20) vertices. - Tree DP often involves combining child DP values at each node. - Use memoization to avoid recomputation in DP on graphs. - Understand edge cases for negative weights (Bellman-Ford) and cycles.

Problem 1: Counting Paths in DAG Link: CSES Counting Paths **Difficulty:** Intermediate

C++ Solution with Explanation Comments:

```
#include <bits/stdc++.h>
using namespace std;
const int MOD=1e9+7;
vector<int> adj[100005];
long long dp[100005];
bool vis[100005];
long long countPaths(int u,int dest){
    if(u==dest) return 1;
    if(vis[u]) return dp[u];
    vis[u]=true; long long res=0;
    for(int v:adj[u]) res=(res+countPaths(v,dest))%MOD;
    return dp[u]=res;
}
int main(){
    int n,m; cin>>n>>m;
    for(int i=0;i<m;i++){
        int u,v; cin>>u>>v; adj[u].push_back(v);
    }
    cout<<countPaths(1,n)<<endl;
}
```

Explanation Comments: - Recursively count paths from node to destination. - Use memoization (dp array) to store already computed results. - Handles large DAG efficiently.

Problem 2: TSP with Bitmask DP Link: CSES TSP **Difficulty:** Advanced

C++ Solution with Explanation Comments:

```
#include <bits/stdc++.h>
using namespace std;
const long long INF=1e18;
int n;
vector<vector<long long>> cost;
vector<vector<long long>> dp;
long long tsp(int mask,int pos){
    if(mask==(1<<n)-1) return cost[pos][0];
    if(dp[mask][pos]!=-1) return dp[mask][pos];
    long long res=INF;
    for(int nxt=0;nxt<n;nxt++){
        if(!(mask&(1<<nxt))) res=min(res,cost[pos][nxt]+tsp(mask|(1<<nxt),nxt));
    }
    return dp[mask][pos]=res;
}
int main(){
    cin>>n; cost.assign(n,vector<long long>(n));
    for(int i=0;i<n;i++) for(int j=0;j<n;j++) cin>>cost[i][j];
    dp.assign(1<<n,vector<long long>(n,-1));
    cout<<tsp(1,0)<<endl;
}
```

Explanation Comments: - Bitmask represents visited nodes. - Recursively try all unvisited nodes and take minimum cost. - Memoization avoids recomputation. - Feasible for n <= 20.

End of Week 24 - Graph DP is crucial for counting paths, finding optimal tours, and solving DAG/tree problems. - Practice DAG DP, TSP Bitmask DP, and Tree DP extensively for ACM-ICPC contests.