Paradigmas de Resolução de Problemas

Programação Dinâmica – Problema do Caixeiro Viajante: Exercícios Resolvidos

Prof. Edson Alves - UnB/FGA

2020

Sumário

OJ 11284 – Shopping Trip

Problema

For some reason, Daniel loves to collect and watch operas on DVD. He can find and order all the operas he wants from Amazon, and they will even deliver them right to his door, but he can usually find a better price at one of his favourite stores. However, with the cost of gas nowadays, it is hard to tell whether or not one would actually save money by driving to the stores to purchase the DVDs.

Daniel would like to buy some operas today. For each of the operas he wants, he knows exactly one store that is selling it for a lower cost than the Amazon price. He would like to know if it would actually be worth it to go out and buy the operas from the stores.

Daniel only knows the road system connecting his favourite stores, and will only use those roads to get around. He knows at least one route, if only an indirect one, to every store.

Problema

In his shopping trip, Daniel begins at his house, drives from store to store in any order to purchase his operas, then drives back to his house. For any particular opera, he can opt not to drive to the store to buy it, since he can just order it from Amazon.

For convenience, Daniel assigned his house the integer 0, and numbered each of his favourite stores with integers starting at 1. You are given a description of the road system and the exact amount it would cost for Daniel to drive each road. For each opera Daniel wants, you are given the number of the store it is available at, and the amount he would save if he bought that particular opera at that store. Your task is to determine the greatest amount of money Daniel can save by making the shopping trip.

Entrada e saída

Input

The first line of input contains a single number indicating the number of scenarios to process. A blank line precedes each scenario.

Each scenario begins with line containing two numbers: N $(1 \leq N \leq 50)$, the number of stores, and M $(1 \leq M \leq 1000)$, the number of roads. The following M lines each contain a description of a road. Each road is described by two integers indicating the house or stores it connects, and a real number with two decimal digits indicating the cost in dollars to drive that road. All roads are two-way.

The next line in the scenario contains a number P $(1 \leq P \leq 12)$, the number of opera DVDs Daniel wants to buy. For each of the P operas, a line follows containing an integer indicating the store number at which the opera is available, and a real number with two decimal digits indicating the difference between the Amazon price and the price at that store in dollars.

Entrada e saída

Output

For each scenario in the input, write one line of output indicating the largest amount of money, in dollars and cents, that Daniel can save by making his shopping trip. Follow the format of the sample output; there should always be two digits after the decimal point to indicate the number of cents. If Daniel cannot save any money by going to the stores, output a single line saying ‘`Don't leave the house`’.

Exemplo de entradas e saídas

Sample Input

```
2

4 5
0 1 1.00
1 2 3.00
1 3 2.00
2 4 4.00
3 4 3.25
3
2 1.50
3 7.00
4 9.00

1 1
0 1 1.50
1
1 2.99
```

Sample Output

```
Daniel can save $3.50
Don’t leave the house
```

Solução

- O problema pode ser modelado como um TSP
- A travessia deve ser formada pelas lojas que vendem os DVDs desejados
- Como o grafo da entrada não é completo, as distâncias mínimas entre quaisquer duas lojas podem ser computada por meio do algoritmo de Floyd-Warshall
- Dois pontos importantes: primeiramente, existe a possibilidade de se adquirir apenas alguns, ou até mesmo nenhum DVD
- O custo de viagem entre as lojas e o ganho na compra dos DVDs devem ter sinais opostos, e a escolha destes sinais pode modificar a atualização dos estados

Solução

- Há uma série de cuidados a serem tomados para obter uma solução AC
- Embora não fique claro na descrição da entrada, o grafo do problema não é simples, podendo acontecer *loops* e multiarestas
- Assim, para cada par de vértices deve ser mantida apenas o menor custo possível
- Os *loops* devem ser eliminados, isto é, $dist(u, u) = 0$ para todo $u \in [0, N]$
- Para evitar erros de precisão, o ideal é trabalhar com múltiplos de centavos, postergando a conversão para dólares para o momento da impressão

Solução $O(N^3 + P^2 2^P)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ii = pair<int, int>;

const int MAXN { 60 }, MAXP { 15 }, oo { 1000000010 };

int dist[MAXN][MAXN];
int st[MAXN][1 << MAXP];

int tsp(int i, int mask, int N, const vector<ii>& xs)
{
    if (st[i][mask] != -1)
        return st[i][mask];

    int res = -dist[i][0];

    for (int j = 0; j < N; ++j)
    {
        if (mask & (1 << j))
            continue;
```

Solução $O(N^3 + P^2 2^P)$

```
        auto u = xs[j].first, c = xs[j].second;
        auto cost = -dist[i][u] + c;

        res = max(res, tsp(u, mask | (1 << j), N, xs) + cost);
    }

    st[i][mask] = res;
    return res;
}

void floyd_warshall(int N)
{
    for (int u = 0; u <= N; ++u)
        dist[u][u] = 0;

    for (int k = 0; k <= N; ++k)
        for (int u = 0; u <= N; ++u)
            for (int v = 0; v <= N; ++v)
                dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v]);
}
```

Solução $O(N^3 + P^2 2^P)$

```

int solve(int N, const vector<ii>& xs)
{
    floyd_warshall(N);

    memset(st, -1, sizeof st);

    return tsp(0, 0, (int) xs.size(), xs);
}

int main()
{
    ios::sync_with_stdio(false);

    int T;
    cin >> T;

    while (T--)
    {
        int N, M;
        cin >> N >> M;
```

Solução $O(N^3 + P^2 2^P)$

```cpp

    for (int u = 0; u <= N; ++u)
        for (int v = 0; v <= N; ++v)
            dist[u][v] = oo;

    while (M--)
    {
        int u, v, d, c;
        char sep;

        cin >> u >> v >> d >> sep >> c;

        dist[u][v] = min(dist[u][v], 100*d + c);
        dist[v][u] = min(dist[u][v], 100*d + c);
    }

    int P;
    cin >> P;

    vector<ii> xs;

```

Solução $O(N^3 + P^2 2^P)$

```cpp
        for (int i = 0; i < P; ++i)
        {
            int u, d, c;
            char sep;
            cin >> u >> d >> sep >> c;

            xs.push_back(ii(u, 100*d + c));
        }

        auto ans = solve(N, xs);

        if (ans)
            cout << "Daniel can save $" << ans / 100 << "."
                << setw(2) << setfill('0') << ans % 100 << endl;
        else
            cout << "Don't leave the house\n";
    }

    return 0;
}
```

SPOJ COURIER – The Courier

Problema

Byteland is a scarcely populated country, and residents of different cities seldom communicate with each other. There is no regular postal service and throughout most of the year a one-man courier establishment suffices to transport all freight. However, on Christmas Day there is somewhat more work for the courier than usual, and since he can only transport one parcel at a time on his bicycle, he finds himself riding back and forth among the cities of Byteland.

The courier needs to schedule a route which would allow him to leave his home city, perform the individual orders in arbitrary order (i.e. travel to the city of the sender and transport the parcel to the city of the recipient, carrying no more than one parcel at a time), and finally return home. All roads are bi-directional, but not all cities are connected by roads directly; some pairs of cities may be connected by more than one road. Knowing the lengths of all the roads and the errands to be performed, determine the length of the shortest possible cycling route for the courier.

Entrada e saída

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

Each test case begins with a line containing three integers: $n\ m\ b$, denoting the number of cities in Byteland, the number of roads, and the number of the courier's home city, respectively ($1 \leq n \leq 100$, $1 \leq b \leq n$, $1 \leq m \leq 10000$). The next m lines contain three integers each, the i-th being $u_i\ v_i\ d_i$, which means that cities u_i and v_i are connected by a road of length d_i ($1 \leq u_i, v_i \leq 100$, $1 \leq d_i \leq 10000$). The following line contains integer z – the number of transport requests the courier has received ($1 \leq z \leq 5$). After that, z lines with the description of the orders follow. Each consists of three integers, the j-th being $u_j\ v_j\ b_j$, which signifies that b_j parcels should be transported (individually) from city u_j to city v_j. The sum of all b_j does not exceed 12.

Entrada e saída

Output

For each test case output a line with a single integer – the length of the shortest possible bicycle route for the courier.

Exemplo de entradas e saídas

Sample Input

```
1
5 7 2
1 2 7
1 3 5
1 5 2
2 4 10
2 5 1
3 4 3
3 5 4
3
1 4 2
5 3 1
5 1 1
```

Sample Output

```
43
```

Solução

- Este é um problema bastante interessante, que pode ser modelado como um TSP, porém de forma não óbvia
- A travessia será composta pelas B entregas, sendo que cada pacote consiste em uma entrega individual
- Seja $dp(i, mask)$ a distância mínima para fazer as entregas partindo do vértice i e já tendo entregue os pacotes sinalizados pela máscara binária $mask$
- O custo das "arestas" entre as entregas é dado pela soma da distância entre i e o vértice u onde está o pacote mais a distância entre u e o destino do pacote v
- As distâncias mínimas entre quaisquer dois vértices pode ser computada por meio do algoritmo de Floyd-Warshall
- Serão $O(N \times 2^K)$ estados distintos, onde $K = \sum_i b_i$

Solução $O(N^3 + K^2 2^K)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ii = pair<int, int>;

const int MAXN { 110 }, MAXB { 13 }, oo { 1000000010 };

int dist[MAXN][MAXN];
int st[MAXN][1 << MAXB];

int tsp(int i, int mask, int N, int B, const vector<ii>& xs)
{
    if (mask == (1 << N) - 1)
        return dist[i][B];

    if (st[i][mask] != -1)
        return st[i][mask];

    int res = oo;

```

Solução $O(N^3 + K^2 2^K)$

```
    for (int j = 0; j < N; ++j)
    {
        if (mask & (1 << j))
            continue;

        auto u = xs[j].first, v = xs[j].second;
        auto cost = dist[i][u] + dist[u][v];

        res = min(res, tsp(v, mask | (1 << j), N, B, xs) + cost);
    }

    st[i][mask] = res;
    return res;
}

void floyd_warshall(int N, const vector<vector<ii>>& adj)
{
    for (int u = 1; u <= N; ++u)
        for (int v = 1; v <= N; ++v)
            dist[u][v] = oo;

```

Solução $O(N^3 + K^2 2^K)$

```
    for (int u = 1; u <= N; ++u)
        dist[u][u] = 0;

    for (int u = 1; u <= N; ++u)
        for (const auto& [v, w] : adj[u])
            dist[u][v] = w;

    for (int k = 1; k <= N; ++k)
        for (int u = 1; u <= N; ++u)
            for (int v = 1; v <= N; ++v)
                dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v]);
}

int
solve(int N, int B, const vector<vector<ii>>& adj, const vector<ii>& xs)
{
    floyd_warshall(N, adj);

    memset(st, -1, sizeof st);
    return tsp(B, 0, (int) xs.size(), B, xs);
}
```

Solução $O(N^3 + K^2 2^K)$

```cpp
int main()
{
    ios::sync_with_stdio(false);

    int T;
    cin >> T;

    while (T--)
    {
        int N, M, B;
        cin >> N >> M >> B;

        vector<vector<ii>> adj(N + 1);

        while (M--)
        {
            int u, v, d;
            cin >> u >> v >> d;

            adj[u].push_back(ii(v, d));
            adj[v].push_back(ii(u, d));
```

Solução $O(N^3 + K^2 2^K)$

```
        }

        int Z;
        cin >> Z;

        vector<ii> xs;

        for (int i = 0; i < Z; ++i)
        {
            int u, v, b;
            cin >> u >> v >> b;

            while (b--)
                xs.push_back(ii(u, v));
        }

        cout << solve(N, B, adj, xs) << '\n';
    }

    return 0;
}
```

Referências

1. OJ 11284 – Shopping Trip
2. SPOJ COURIER - The Courier