PHarr XCPC ex Templates

PHarr

SMU

November 3, 2024

Contents

编程技巧和基础算法

Linux 下运行脚本

```bash
#!/bin/bash
g++ $1.cpp -o $1 -g -O2 -std=c++20 -Wl,--stack=268435456 -Wall -fsanitize=undefined -fsanitize=address && echo compile_successfully >&2 && ./$1
```

数据结构

图论

差分约束系统

定义

差分约束系统是一种特殊的 n 元一次不等式组，它包含 n 个变量 $x_1, x_2, \dots, x_n$ 以及 m 个约束条件，每个约束条件是由两个其中的变量做差构成的，形如 $x_i - x_j \le c_k$，其中 $1 \le i, j \le n, i \ne j, 1 \le k \le m$ 并且 c_k 是常数（可以是非负数，也可以是负数）。我们要解决的问题是：求一组解 $x_1 = a_1, x_2 = a_2, \dots, x_n = a_n$，使得所有的约束条件得到满足，否则判断出无解。

差分约束系统中的每个约束条件 $x_i - x_j \le c_k$ 都可以变形成 $x_i \le x_j + c_k$，这与单源最短路中的三角形不等式 $dist[y] \le dist[x] + z$ 非常相似。因此，我们可以把每个变量 x_i 看做图中的一个结点，对于每个约束条件 $x_i - x_j \le c_k$，从结点 j 向结点 i 连一条长度为 c_k 的有向边。

注意到，如果 $\{a_1, a_2, \dots, a_n\}$ 是该差分约束系统的一组解，那么对于任意的常数 d，$\{a_1 + d, a_2 + d, \dots, a_n + d\}$ 显然也是该差分约束系统的一组解，因为这样做差后 d 刚好被消掉。

过程

设 $dist[0] = 0$ 并向每一个点连一条权重为 0 边，跑单源最短路，若图中存在负环，则给定的差分约束系统无解，否则，$x_i = dist[i]$ 为该差分约束系统的一组解。

性质

一般使用 Bellman–Ford 或队列优化的 Bellman–Ford（俗称 SPFA，在某些随机图跑得很快）判断图中是否存在负环，最坏时间复杂度为 $O(nm)$。

如果题目给定了一个源点，则不需要建立超级源点。

```cpp
// luogu P1993
// 有三种约束条件
// x[a] >= x[b] + c -> x[b] - x[a] <= -c -> add(a, b, -c)
// x[a] <= x[b] + c -> x[a] - x[b] <= c -> add(b, a, c)
// x[a] == x[b] -> x[a] - x[b] <= 0 and x[b] - x[a] <= 0 -> add(a, b, 0), add(b, a, 0)
#include <bits/stdc++.h>

using namespace std;

const int inf = INT_MAX / 2;

using vi = vector<int>;
using pii = pair<int, int>;

int main() {
    ios::sync_with_stdio(false), cin.tie(nullptr);
    int n, m;
    cin >> n >> m;

    vector<vector<pii>> e(n + 1);
    for (int op, a, b, c; m; m--) {
        cin >> op;
        if (op == 1) {
            cin >> a >> b >> c;
            e[a].emplace_back(b, -c);
        } else if (op == 2) {
            cin >> a >> b >> c;
            e[b].emplace_back(a, c);
        } else {
```

```
            cin >> a >> b;
            e[a].emplace_back(b, 0);
            e[b].emplace_back(a, 0);
        }
    }

    for (int i = 1; i <= n; i++)
        e[0].emplace_back(i, 0);

    vector<int> dis(n + 1, inf), vis(n + 1), tot(n + 1);
    dis[0] = 0, vis[0] = 1, tot[0]++;
    bool ok = true;
    queue<int> q;
    q.push(0);
    while (not q.empty() and ok) {
        int x = q.front();
        q.pop();
        vis[x] = 0;
        for (auto [y, w]: e[x]) {
            if (dis[y] <= dis[x] + w) continue;
            dis[y] = dis[x] + w;
            if (vis[y] != 0) continue;
            vis[y] = 1;
            q.push(y);
            tot[y]++;
            if (tot[y] > n) {
                ok = false;
                break;
            }

        }
    }

    if (ok) cout << "Yes\n";
    else cout << "No\n";
    return 0;
}
```

数学知识

计算几何

已知正方形对角线两点坐标，求另外两点坐标

按照顺时针方向，正方形上四点 A, B, C, D，两对角线交点 O。现在已知 $A(ax, ay), C(cx, cy)$ 求另外两点坐标。

令 $\vec{v} = \frac{C-A}{2} = \left(\frac{cx-ax}{2}, \frac{cy-ay}{2}\right)$，则有 $O = A + \vec{v} = C - \vec{v} = (ax + vx, ay + vy) = (cx - vx, cy - vy)$

根据正方形的对称性可知

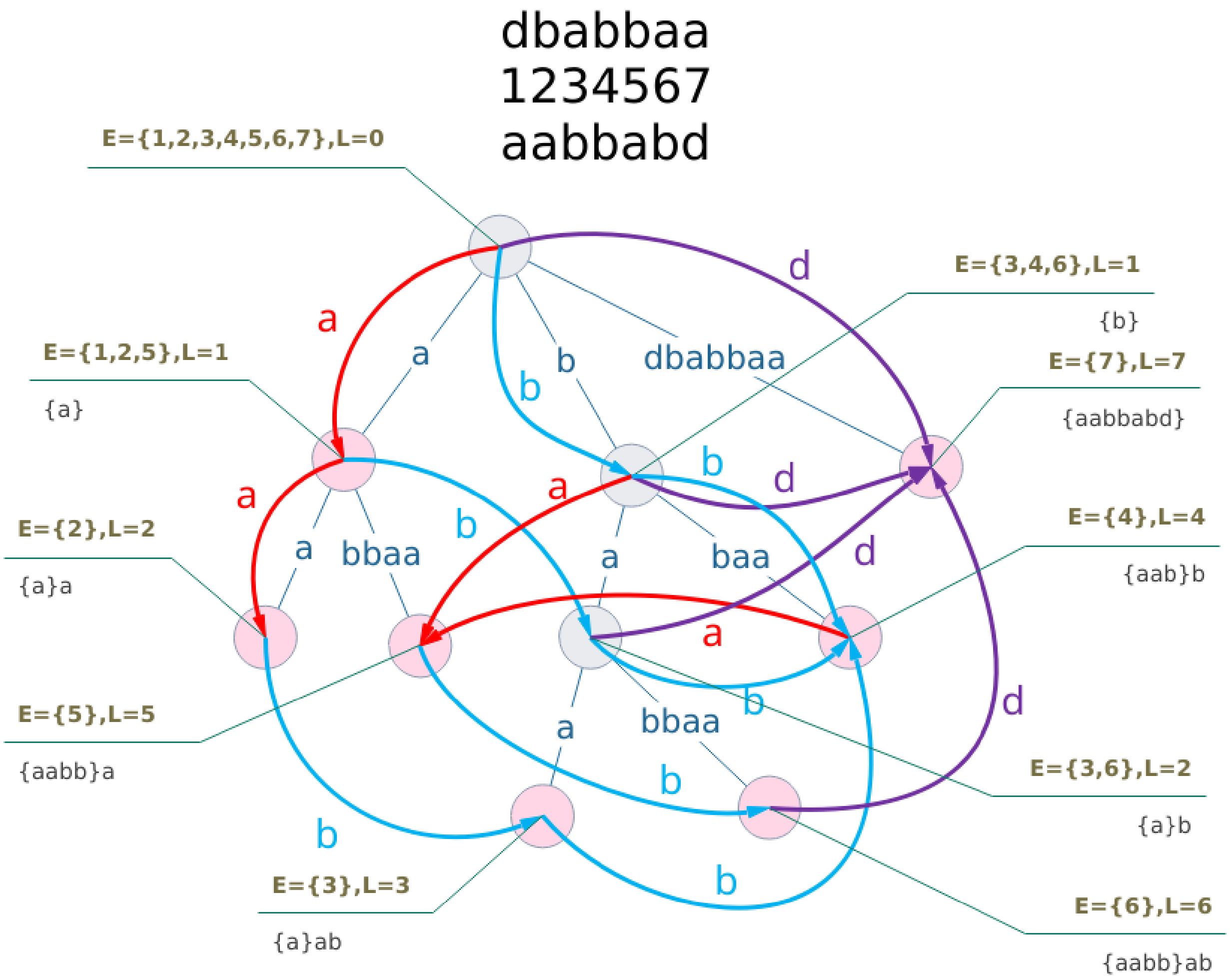
$$B = (ox - vy, oy + vx) = (ax + vx - vy, cy + vx - vy)D = (ox + vy, oy - vx) = (cx - vx + vy, ay - vx + vy)$$

令 $vp = vx - vy = \frac{cx-ax-cy+ay}{2}$ 分别代入上式子可得

$$B = (ax + vp, cy + vp)C = (cx - vp, ay - vp)$$

字符串

后缀自动机



动态规划