

# PHarr XCPC ex Templates

PHarr

SMU

November 3, 2024

# Contents

|                       |          |
|-----------------------|----------|
| <b>编程技巧和基础算法</b>      | <b>2</b> |
| Linux 下运行脚本 . . . . . | 2        |
| <b>数据结构</b>           | <b>2</b> |
| <b>图论</b>             | <b>2</b> |
| 差分约束系统 . . . . .      | 2        |
| <b>数学知识</b>           | <b>3</b> |
| 计算几何 . . . . .        | 3        |
| <b>字符串</b>            | <b>4</b> |
| 后缀自动机 . . . . .       | 4        |
| <b>动态规划</b>           | <b>4</b> |

# 编程技巧和基础算法

## Linux 下运行脚本

```
1 #!/bin/bash
2 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  $x_i - x_j \leq c_k$ ，其中  $1 \leq i, j \leq n, i \neq j, 1 \leq k \leq m$  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  $x_i - x_j \leq c_k$  都可以变形为  $x_i \leq x_j + c_k$ ，这与单源最短路中的三角形不等式  $dist[y] \leq dist[x] + z$  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  $x_i - x_j \leq c_k$ ，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。

```
1 // luogu P1993
2 // 有三种约束条件
3 // x[a] >= x[b] + c -> x[b] - x[a] <= -c -> add(a, b, -c)
4 // x[a] <= x[b] + c -> x[a] - x[b] <= c -> add(b, a, c)
5 // x[a] == x[b] -> x[a] - x[b] <= 0 and x[b] - x[a] <= 0 -> add(a, b, 0), add(b, a, 0)
6 #include <bits/stdc++.h>
7
8 using namespace std;
9
10 const int inf = INT_MAX / 2;
11
12 using vi = vector<int>;
13 using pii = pair<int, int>;
14
15 int main() {
16     ios::sync_with_stdio(false), cin.tie(nullptr);
17     int n, m;
18     cin >> n >> m;
19
20     vector<vector<pii>> e(n + 1);
21     for (int op, a, b, c; m--;) {
22         cin >> op;
23         if (op == 1) {
24             cin >> a >> b >> c;
25             e[a].emplace_back(b, -c);
26         } else if (op == 2) {
27             cin >> a >> b >> c;
28             e[b].emplace_back(a, c);
29         } else {
```

```

30         cin >> a >> b;
31         e[a].emplace_back(b, 0);
32         e[b].emplace_back(a, 0);
33     }
34 }
35
36 for (int i = 1; i <= n; i++)
37     e[0].emplace_back(i, 0);
38
39 vector<int> dis(n + 1, inf), vis(n + 1), tot(n + 1);
40 dis[0] = 0, vis[0] = 1, tot[0]++;
41 bool ok = true;
42 queue<int> q;
43 q.push(0);
44 while (not q.empty() and ok) {
45     int x = q.front();
46     q.pop();
47     vis[x] = 0;
48     for (auto [y, w]: e[x]) {
49         if (dis[y] <= dis[x] + w) continue;
50         dis[y] = dis[x] + w;
51         if (vis[y] != 0) continue;
52         vis[y] = 1;
53         q.push(y);
54         tot[y]++;
55         if (tot[y] > n) {
56             ok = false;
57             break;
58         }
59     }
60 }
61
62
63 if (ok) cout << "Yes\n";
64 else cout << "No\n";
65 return 0;
66 }

```

## 数学知识

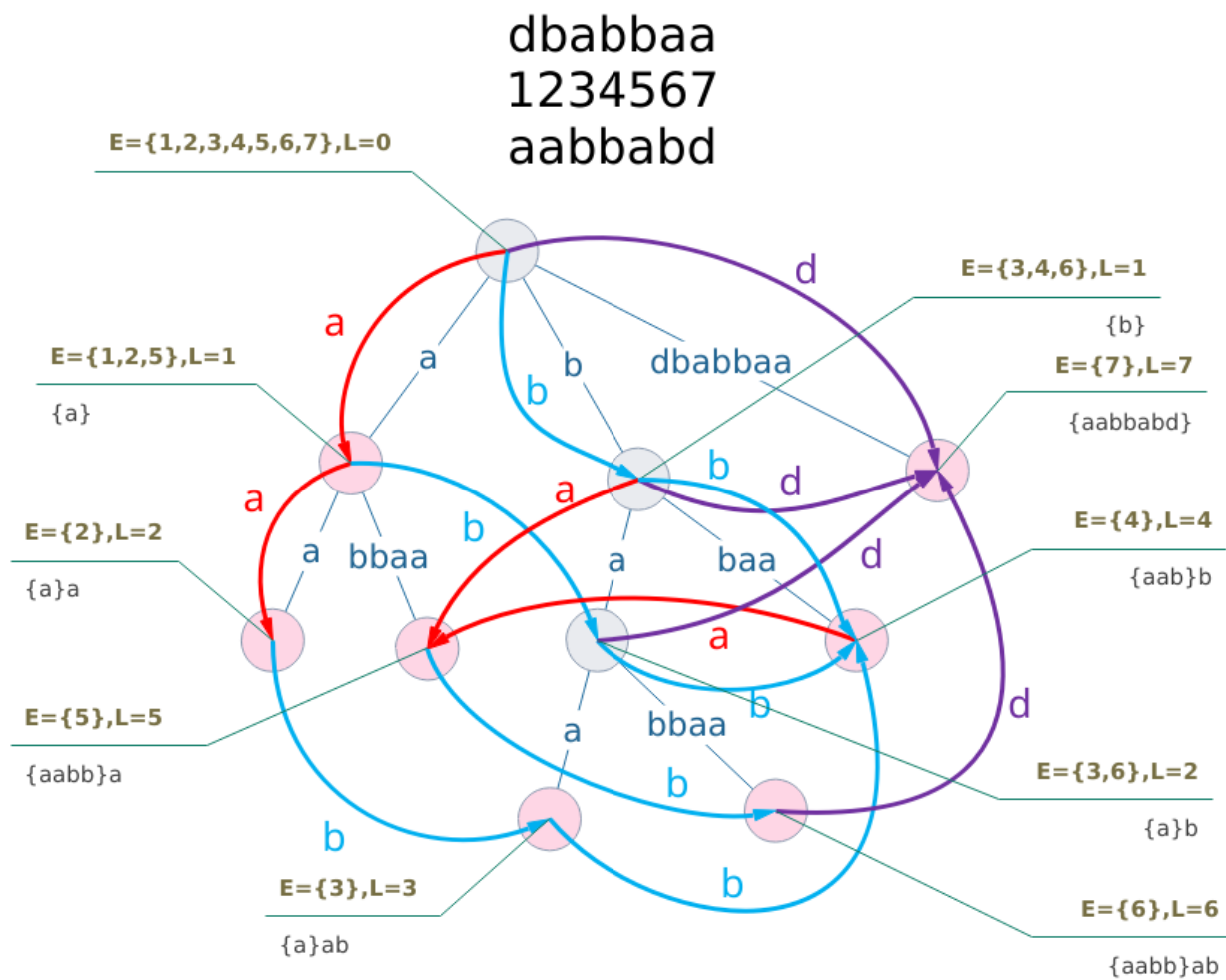
### 计算几何

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。

## 字符串

### 后缀自动机



### 动态规划