模拟赛与作业讲评

基础-提高衔接计划

览遍千秋

2025-03-22

洛谷

www.luogu.com.cn

模拟赛

模拟赛概览

- 第一题
简单语法
- 第二题
简单搜索
- 第三题
大模拟 / STL
- 第四题
动态规划

题号	平均分	最高分
A		
B		
C		
D		

试卷分发

简单语法题。

根据题目给出的分配规则，将每一个 a_i 表示为 $30k + p(0 \le p < 30)$ 的形式。

k,p 很容易算出，$k = \frac{a_i}{30}, p = a_i \bmod 30$。

依照题意模拟计数即可。

- 若 $a_i = 30k$，其中 k 为非负整数，则分配 k 袋 30 份的试卷。
- 若 $a_i = 30k + p$，其中 k 为非负整数且 $0 < p \le 10$，则分配 k 袋 30 份的试卷与 1 袋 10 份的试卷。
- 若 $a_i = 30k + p$，其中 k 为非负整数且 $10 < p \le 20$，则分配 k 袋 30 份的试卷与 2 袋 10 份的试卷。
- 若 $a_i = 30k + p$，其中 k 为非负整数且 $20 < p < 30$，则分配 $k + 1$ 袋 30 份的试卷。

卡牌游戏

- 子任务 1： $1 \leq n \leq 15$，30pts

这个数据范围是给 $O(n!)$、$O(2^n)$ 等等指数级复杂度做法的，命题时并没有想到类似的做法。

卡牌游戏

- 子任务 2：$p_i = i \bmod n + 1$，20pts

即，$p_1 = 2, p_2 = 3, \cdots, p_n = 1$。

当到第 n 轮时，所有人将同时重新拿到起始的卡牌。

答案为 n。

卡牌游戏

- 子任务 3：n 为偶数。
- $p_i = i + 1$（i 为奇数）
- $p_i = i - 1$（i 为偶数）
- 20pts

相邻的两个人交换卡牌，在第 2 轮就会拿到初始卡牌，答案为 2。

卡牌游戏

由于 p 是一个排列，用图论的语言描述，存在 i 指向 p_i 的 n 条边，每个点都有一条边指出，一条边指入。

最终，会构成若干个环。该环中所有的点，都将在 [环大小] 轮第一次拿到初始卡牌。

通过若干次 dfs，我们可以标记出每一个环，答案即为环大小的最大值。

时间复杂度 $O(n)$。

驿站管理

考察 STL 水平。

一个快递存放的位置编号，按照 $x-y-z$ 表示。

题目给定了编号的确定原则，即：

- x 尽可能小
- 在满足 x 尽可能小的前提下，y 尽可能小
- 在满足 x,y 尽可能小的前提下，z 尽可能小

两个编号使用的先后关系，是可以严格比较的。

用结构体存储 x,y,z，可以根据上述的规则重载相关比较符号。

驿站管理

```cpp
struct Info {
    int x, y, z;
    Info(int x, int y, int z): x(x), y(y), z(z) {}
    bool operator < (const Info &a) const {
        if(x != a.x) return x > a.x;
        if(y != a.y) return y > a.y;
        return z > a.z;
    }
};
```

上面的代码中，定义了 Info 结构体，并重载了其小于号。

驿站管理

在之前，我们接触了 STL 中的 priority_queue。

```
priority_queue <T> varname;
```

其中，`T` 为类型名。

T 可以是 int, long long 等基本数据类型

亦可以是 pair<int, int> 等复合数据类型

还可以是用 struct,class 等关键字自定义的类型

自定义的类型须重载小于号

驿站管理

用 3 个 priority_queue 维护每类货物还剩余的可用编码。

- 加入货物

查看优先队列是否为空，若为空，则 Reject

否则取出 top，并输出

- 取走货物

对应编码放入优先队列

童话镇

- 子任务 1：$|S| = 1$，25pts

字符串 S 长度为 1，有且仅有几种情况：

- S 为 _
- S 为 %
- S 为小写英文字母

分类考虑

童话镇

设 $dp[i][j]$ 表示 S 的前 i 个字符和 T 的前 j 个字符是否可以匹配。

- $S[i]=T[j]$，且不为通配符

用这两个字符匹配，dp[i][j]|=dp[i-1][j-1]

- $S[i]$ 为 % 或 $T[j]$ 为 %

用这两个字符匹配，dp[i][j]|=dp[i-1][j-1]

- $S[i]$ 为 _

若 $T[j]$ 是和 $S[i]$ 匹配的第一个字符，则取 dp[i-1][j-1]

若 $T[j]$ 是和 $S[i]$ 匹配的第 2+ 个字符，则取 dp[i][j-1]

dp[i][j]|=dp[i-1][j-1] | dp[i][j-1]

- $T[j]$ 为 _，与上一种情况类似

W8 作业讲评

课堂例题

- P3366 【模板】最小生成树
- P4779 【模板】单源最短路径
- P4568 [JLOI2011] 飞行路线

二叉搜索树

- **题意简述**

实现一个 BST，支持插入、删除、排名、前驱、后继操作。

二叉搜索树

- 思路分析

模板题

排名等操作，给每一棵子树加一个 size 数组即可

地铁路线

- **题意简述**

给定一张 n 个节点、m 条边的无向图，边有边权。

1. 地铁线路能够连接全部高校（直接或间接）
2. 在满足要求 1 的情况下，改造的线路尽可能少
3. 在满足要求 1、2 的情况下，改造的线路中拥挤程度的最大值尽可能小

地铁路线

- **思路分析**

要求 1：选出的边能够使得图联通

要求 2：改造的边尽可能少——$n-1$ 条，树

要求 3：拥挤程度的最大值尽可能小：最小生成树

为什么要求 3 推导出最小生成树？

地铁路线

考虑 Kruskal 算法过程：

- 按照边权从小到大排序
- 对于边 $e = (u, v, w)$，如果 u, v 在同一个并查集内，则不选，否则选入生成树

e 没有选入最小生成树，原因是 $u \to v$，存在一条路径 $u \to x_1 \to x_2 \to \cdots \to v$，其中每一条边的边权都小于 e。

因此，满足条件 3。

CAPOOS

- **题意简述**

有 N 只 CAPOO 在地上蠕动，第 i 只 CAPOO 的颜色为 C_i。

七海想要将这 N 只 CAPOO 全部打包带走。七海每次可以在地上选择两只 CAPOO x, y，获得 $\left(C_x^{C_y} + C_y^{C_x}\right) \bmod M$ 的快乐值。

接着，七海将其中一只 CAPOO 放进包里，另外一只 CAPOO 放回地上。

七海想知道，当地上仅剩一只 CAPOO 时，她的快乐值最大为多少。

CAPOOS

- **思路分析**

选取第 x, y 只 CAPOO，相当于在点 x, y 之间连边。

每次选取减少 1 只，共连出 $n-1$ 条边。

每一只 CAPOO 至少被选取了一次（才能只剩下一只），因此 $n-1$ 条边构成一个连通图（树）。

枚举 x, y，在 x, y 之间连边权为 $\left(C_x^{C_y}+C_y^{C_x}\right) \bmod M$ 的边，做最大生成树。

紧急救助

- **题意简述**

n 座城市，第 i 座城市的坐标为 (x_i, y_i)。

两座城市的距离为欧几里得距离的平方。

小于 p 的边不能使用，求最小生成树。

紧急救助

- **思路分析**

对于任意两座城市，建立边，共 n^2 条，边权小于 p 的不建。

Kruskal 模板。

基础最短路练习题

- **题意简述**

给定 n 个点 m 条边的简单无向连通图 G，边有边权。保证没有重边和自环。

定义一条简单路径的权值为路径上所有边边权的异或和。

保证 G 中不存在简单环使得边权异或和不为 0。

Q 次询问 x 到 y 的最短简单路径。

基础最短路练习题

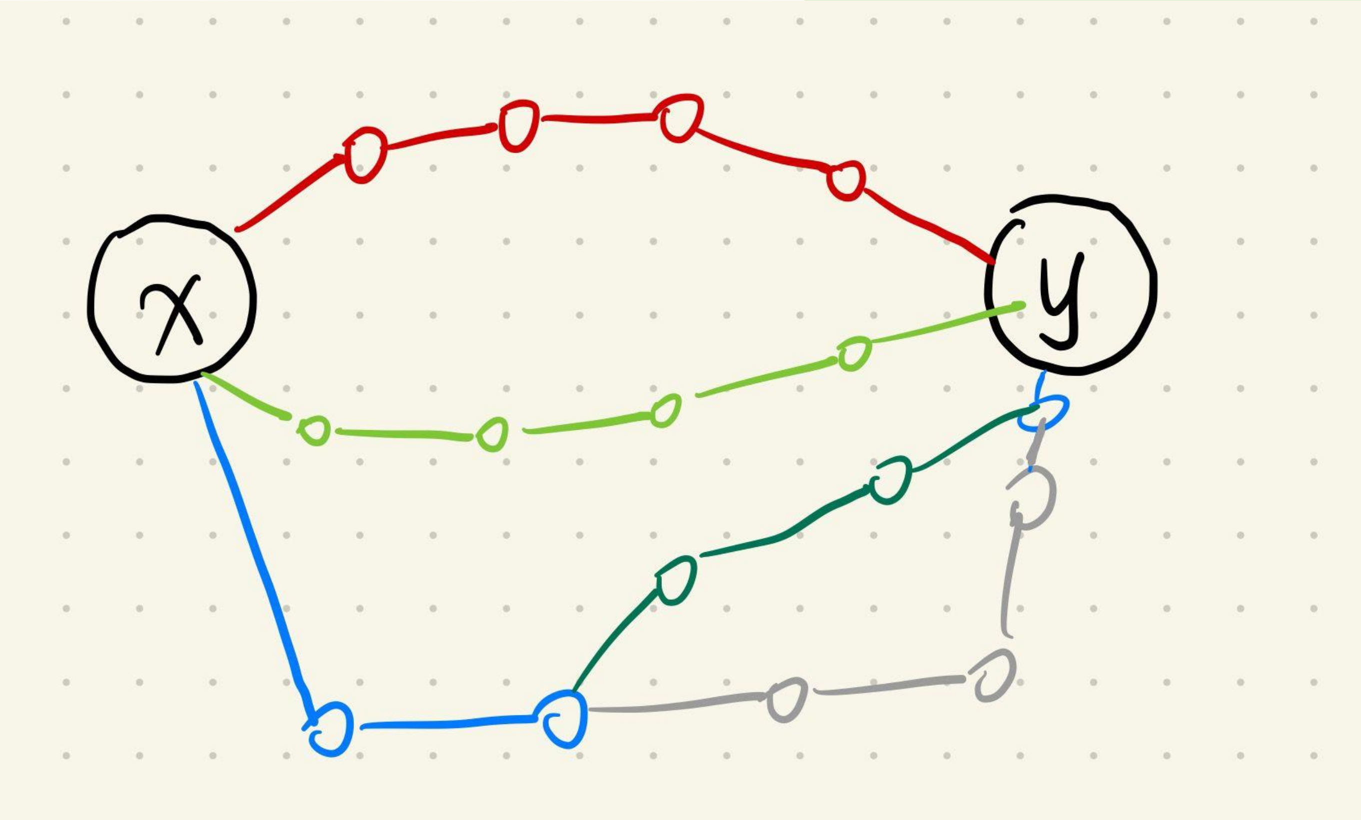
- **思路分析**

注意题目限制：保证 G 中不存在简单环使得边权异或和不为 0。

换言之，若存在简单环，则其边权异或值为 0。

基础最短路练习题

如图所示，$x \to y$ 可能存在多条路径，但是，任意两条路径将构成一个简单环。



基础最短路练习题

完全不重合的路径（红色、草绿色），构成了一个大的环，因此，红色路径和草绿色路径的权值异或为 0。

根据自反性，红色路径的权值等于草绿色路径的权值。

部分重合的路径（灰色、深绿色，重合部分蓝色），构成了一个小环，灰色与深绿色部分异或值为 0，权值相同，因此两条路径权值也相同。

基础最短路练习题

结论：$x \to y$ 任意一条路径权值相同，任取一条即可。

如何实现任取一条呢？取原图的任意一棵生成树即可，树上路径唯一。

求出一棵生成树后，如何维护 $x \to y$ 路径的异或值呢？

树上倍增？可以，但没必要。

基础最短路练习题

技巧：点边互化

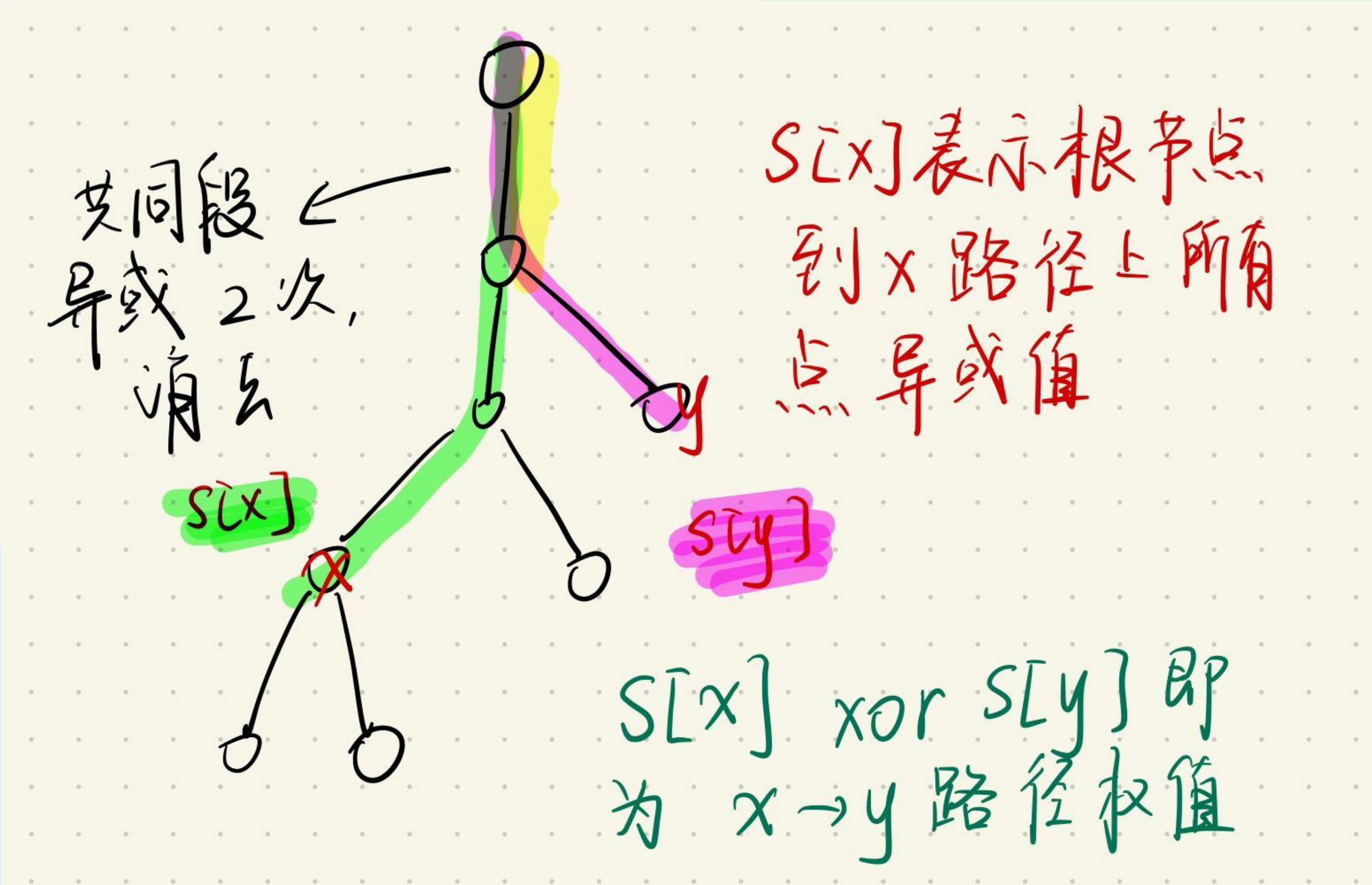
在有根树中，如何将边权挂载到点上？

一条边连接父节点、子节点

边权放在子节点中，只有根结点没有点权

基础最短路练习题

树上前缀异或



基础最短路练习题

总结:

- 求原图的任意生成树
- 一次 dfs 计算树上前缀异或
- 输出 s[x] xor s[y]

认为并查集近似线性，时间复杂度为 $O(n+Q)$。

邮递员送信

- **题意简述**

n 个结点，m 条边的有向图，边有边权。

对于所有的 $2 \sim n$ 号节点，由 1 号节点出发访问结点 x，再返回 1 号节点，求最短距离。

邮递员送信

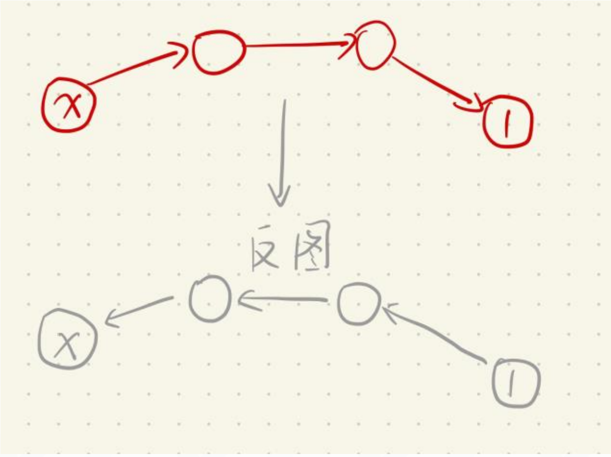
- **思路分析**

由 1 出发，到节点 x 的最短距离较为简单，以 1 节点为出发点，做 SSSP 问题即可，Dijkstra 算法模板。

从节点 x 到节点 1 的最短距离，应该如何计算呢？

建立反图，$x \to 1$ 的最短路即反图中 $1 \to x$ 的最短路。

在原图、反图中进行两次 Dijkstra 即可。



道路重建

- **题意简述**

n 个点 m 条边的无向图，边有边权。

其中 d 条边损毁，从 A 到 B 连通，求最小代价。

道路重建

- **思路分析**

经过损毁的边，要付出边权的代价。

经过未损毁的边，不需要付出代价，边权设为 0 即可。

注意 Dijkstra 算法不能处理负权。

由于 n 非常小，可以用 Floyd 算法解决最短路问题。

Floyd 算法将在后面讲解。

道路重建

思想：缩点

未损毁的边相连的点距离一律为 0，可以用并查集维护相连情况。

把一个并查集视作一个新点，消除了全部的 0 权边。

可以使用 Dijkstra 算法。

逐个击破

- **题意简述**

给一棵 n 个结点的树，边有边权。

其中有 K 个关键点，删去一些边，使得关键点不连通。

最小化代价。

逐个击破

- **思路分析**

正难则反，选出尽可能多的边，使得 K 个关键点不连通。

依照边权排序，逐个尝试选取。

并查集合并时需要标记当前集合内是否有关键点，两个有关键点的并查集不能合并。

口袋的天空

- **题意分析**

给出 n 个节点 m 条边组成的无向图，边有边权。

选出一些边，使得节点构成 K 个连通块，最小化代价。

口袋的天空

- **思路分析**

回顾 Kruskal 算法，每选择一条边，将有两个并查集合并。

并查集的数目即为连通块的数目。

构成 1 个连通块，即构成生成树，选择 $n-1$ 条边。

构成 K 个连通块，选择 $n-K$ 条边。

按照 Kruskal 算法流程，对边按边权从小到大排序，选出 $n-K$ 条边即停止。

[Cnoi2020] 雷雨

- **题意简述**

给定 $N \times M$ 的矩阵，每个格子有一个数。

从 O 出发到 A、B，代价为路径上所有格子中的数之和。

求最小代价。

[Cnoi2020] 雷雨

- **思路分析**

由 $O \to A, O \to B$ 的路径，一定存在一个分叉点 X。

枚举点 X，$O \to X, X \to A, X \to B$ 的最短路径长度，可以通过以 O, A, B 为起点进行三次 SSSP 问题解决。

小明的游戏

- **题意简述**

给 $n \times m$ 矩阵，有两类格子。

同类格子移动不需要代价，不同类格子移动需要 1 代价。

求从起点到终点的最小代价。

小明的游戏

- **思路分析**

可以使用 01-BFS 完成。

也可以建图完成，对于 (i,j)，编号为 $(i-1)\times m+j$。

(i,j) 可以到 $(i-1,j),(i,j-1),(i,j+1),(i+1,j)$，依照格子类型赋予不同边权。

逃离僵尸岛

- **题意简述**

给 n 个节点 m 条边的无向图，点有点权。

K 个关键点，距离关键点距离不超过 S 的点权为 Q，其他点点权为 P。

逃离僵尸岛

- **思路分析**

首先，需要处理出所有危险城市。

以 K 个关键点作为起点进行 BFS，距离不超过 S 的全部标记。

即，一开始将 K 个点全部压入队列，而不是进行 K 次 BFS。

此时，就可以得到每个点的点权，但是 Dijkstra 算法是解决边权的，怎么转化呢？

逃离僵尸岛

在存储图时，无向边 $x - y$，本质上被拆分为了 $x \to y$ 和 $y \to x$ 两条有向边。

用 $val(x)$ 表示节点 x 的点权。

进入点 x，需要付出 x 点权的代价。

因此，$y \to x$ 的边权可以记为 $val(x)$。

同理，$x \to y$ 的边权可以记为 $val(y)$。

至此，可以使用 Dijkstra 计算最短路。

Floyd 算法

全源最短路径

全源最短路问题

Dijkstra 算法可以解决 SSSP 问题，即单源最短路径问题。

执行一次 Dijkstra 算法，可以计算出从源点 S 到任意节点的最短路。

然而，我们有时关心任意两个节点 i, j 之间的最短路径。

这称为全源最短路问题。

Floyd 算法

Floyd 算法基于动态规划思想。

设 $f[k][i][j]$ 表示由 i 到 j，只经过编号不超过 k 的点的最短路径。

对于 $f[k][i][j]$，有两种情况：

- 不经过第 k 个点，$f[k][i][j] = f[k-1][i][j]$
- 经过第 k 个点，$f[k][i][j] = f[k][i][k] + f[k][k][j]$

因此，$f[k][i][j] = \min(f[k-1][i][j], f[k][i][k] + f[k][k][j])$

Floyd

类似于背包的优化方法，可以压掉 f 数组的第一维。

初始化 f 数组为 INF，$f[i][i] = 0$ 除外。

如果有 $i \to j$ 的边，$f[i][j] = w$。

```
for(int k = 1; k <= n; k++) {
    for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= n; j++) {
            f[i][j] = min(f[i][j], f[i][k] + f[k][j]);
        }
    }
}
```

Floyd

Floyd 的动态规划思想非常重要，我们在第九周的作业中将为大家提供一些相关的试题。