

A

不妨采用贪心策略：能喝则喝。

从左往右扫描，维护当前手中有几杯咖啡即可。

B

模拟即可

C

定义 $f[i][j]$ ：在第 j 列、以第 i 行为结尾的“向上连续相等段”的起始行

则有等价条件：列 j 在 $[l, r]$ 内全相同 $\Leftrightarrow f[r][j] \leq l$

(等价说法：段长 $len[r][j] = r - f[r][j] + 1 \geq r - l + 1$)

1. 预处理 f

- 初始化： $f[1][j] = 1$
- 递推规则：若 $s[i][j] == s[i-1][j]$ 则 $f[i][j] = f[i-1][j]$ ，否则 $f[i][j] = i$
- 时间复杂度： $O(nm)$

2. 行内排序

- 对每个行 r ，将 $f[r][1..m]$ 排序为 F_r
- 时间复杂度： $O(n \cdot m \log m)$

3. 回答询问

- 给定区间 $[l, r]$ ，统计满足 $f[r][j] \leq l$ 的列数 j
- 等价于在排序数组 F_r 中二分查找 $upper_bound(l)$ 的索引
- 单次查询时间复杂度： $O(\log m)$

D

设 $f(S)$ 表示按照拓扑序从前往后的顺序加点，初始为空集，到已经加入 S 这个点集，这一段的加点方案数。设 $g(S)$ 表示按照拓扑序从后往前的顺序删点，初始为全集，到现在还剩下 S 这个点集，此时删除顺序的方案数。 f, g 都可以在 $O(2^n n)$ 时间内简单地 dp 求出。

对于给定的 u, v ， u 在 v 前，就是要：拓扑序加入 v 那一刻， u 已经在拓扑序里了。所以，

$$ans_{u,v} = \sum_{\substack{u \in S \\ v \notin S}} f(S) g(S \cup \{v\})$$

(枚举加入 v 之前一刻, 有哪些点加入了拓扑序)

直接实现该算法的时间复杂度为 $O(2^n n^2)$, 但 S 的枚举有 $1/4$ 的常数, 已经可以过。

当然, 上述算法还可以继续优化到 $O(2^n n)$: 枚举 S, v , 相当于 $\forall u \in S, ans_{u,v}$ 都加上一个定值。因此瓶颈在于: “给出 $a_0 \sim a_{2^n-1}$, 对所有 u 求出 $\sum_{u \in S} a_S$ ”。可以用下面的方法:

```
for i from n-1 downto 0:
  for j from 2^i to (2^{i+1}-1):
    ans[i] += a[j]
    a[j-2^i] += a[j]
```

最后 ans_u 就是答案。因此我们把 $O(2^n n)$ 的循环优化到了 $O(2^n)$ (但是实际上速度仅仅变快了不到一倍)

E

题意

平面上 n 个点, 坐标为 (x_i, y_i) , 有权值 c_i 。需要选 k 个点, 使得这些点的权值和加上这些点 x 坐标最大值与最小值的差加上这些点 y 坐标最大值与最小值的差尽量大。

经典思路

最大化 $\max - \min$, 显然是越大越好。可以弱化问题: 选 k 个点, 选某个点的 $-x_i$, 某个点的 x_j , 某个点的 $-y_k$, 某个点的 y_l , 以及所有点的权值和。不优的情况不会选, 最优决策自动选择最大值和最小值做差。

动态规划解法

设 $f_{i,j,s}$ 表示前 i 个物品, 选了 j 个, 当前四种特殊权值 $(-x_i, x_j, -y_k, y_l)$ 是否使用的最大价值。预处理 $val_{i,s}$ 表示点 i 在状态 s 下的贡献。转移方程: $f_{i,j,s} = \max(f_{i-1,j-1,t} + val_{i,s \oplus t})$

优化思路

j 个点中只有最多 4 个有特殊权值, 其余是普通点 (体积为 1, 价值为 $val_{i,0}$)。固定 k 时:

- 权值前 $k - 4$ 的元素必选, 可能贡献特殊权值
- 不在前 $k - 4$ 的元素最多选 4 个, 且必须贡献特殊权值

按权值降序排序，分别做前缀/后缀背包：

- 前缀背包忽略体积（普通点）
- 后缀背包体积 ≤ 4 （特殊点）

复杂度优化为 $O(n \times 4 \times 3^4)$ 。

查询处理

枚举特殊点数量 t ($0 \leq t \leq \min(4, k)$)，合并前缀/后缀背包： $\text{ans} = \max_{t=0}^{\min(4,k)} \{f_{k-t,s} + g_{k-t+1,t,s \oplus 2^4}\}$

总复杂度 $O(n \times 3^4 + n \times 4 \times 3^4 + q \times 4 \times 2^4)$ 。

F

讲下我的做法，实测 118 次操作，能过原题。

考虑二进制序列（仅含0和1），将连续段分组处理，序列形态如：10101010...

- 单次操作可使0的连续段数量减半，分段模式示例：1|0|10|1|0|10...
- 最终可能形成 101 结构，再通过一次操作 1|01 完成排序

考虑分治

- 将 $\leq n/2$ 的元素视为0， $> n/2$ 的视为1，应用0/1专用算法
- 递归处理左右子区间，同一层可合并处理
- 若最终顺序异常，整体执行reverse操作

复杂度分析

操作次数约为 $0.5 \cdot \log^2 n$ ，满足题目要求