The 19th Zhejiang Provincial Collegiate Programming Contest Editorial

Prepared by Zhejiang University

2022 年 4 月 16 日

Overview

	Easiest												Hardest
Idea	B	C	A	G	L	M	I	J	F	E	K	D	H
Coding	C	B	L	A	H	M	I	G	E	J	D	K	F
Summary	B	C	A	L	G	M	I	J	F	E	K	D	H

B. JB Loves Comma

Shortest Judge Solution: 403 Bytes

Description

给定字符串 S，在每个 “cjb” 子串后面添加一个逗号。

- $|S| \leq 10^5$。

Solution

- 签到题，读入 $\rightarrow$ 处理 $\rightarrow$ 输出。
- 时间复杂度 $O(|S|)$。

C. JB Wants to Earn Big Money

Shortest Judge Solution: 274 Bytes

Description

给定 n 个人的预期价格和股票交易价格，统计参与交易的人数。

- $n \leq 10^5$。

Solution

- 签到题，读入 $\rightarrow$ 处理 $\rightarrow$ 输出。
- 时间复杂度 $O(n)$。

A. JB Loves Math

Shortest Judge Solution: 596 Bytes

Description

给定两个正整数 a, b，你需要选定一个正奇数 x 和一个正偶数 y。

之后的每一步操作中，你可以将 a 增大 x 或者将 a 减小 y。

求把 a 变成 b 的最少操作次数。

- $1 \leq a, b \leq 10^6$。
- 测试数据组数 $\leq 10^5$。

Solution

- 答案不超过 3。
- 分类讨论。

L. Candy Machine

Shortest Judge Solution: 437 Bytes

Description

给定 N 个正整数，从中选择一个子集使得严格大于该集合平均数的数字个数尽可能多。

- $N \leq 10^6$。

Solution

- 假设最终选择的集合的平均数不超过 k。
- 为使平均数不超过 k，应将 $\leq k$ 的数全部选入，然后贪心选择 $> k$ 的部分中最小的若干个数。
- 因此将 N 个数从小到大排序后，最优解一定是一个前缀。
- 枚举每个前缀，双指针统计严格大于平均数的数字个数。
- 时间复杂度 $O(N \log N)$。

G. Easy Glide

Shortest Judge Solution: 1116 Bytes

Description

给定平面上 n 个滑行点以及起点 S 和终点 T。

行走速度为 V_1，每次经过某个滑行点后可以按 V_2 速度滑行 3 秒。

求从 S 滑行到 T 所需的最少时间。

- $n \leq 1000$。

Solution

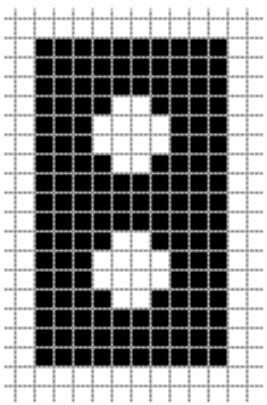
- 建立一张 $n+2$ 个点的有向图，分别表示 n 个滑行点以及起点 S 和终点 T。
- 由起点向每个点连单向边，边权为 S 按 V_1 走到至该点所需的时间。
- 由每个滑行点向其它滑行点以及终点连单向边，边权为先按 V_2 滑行至多 3 秒然后按 V_1 行走至目的地所需的时间。
- 朴素 Dijkstra 求 S 到 T 的最短路。
- 时间复杂度 $O(n^2)$。

M. BpbBppbpBB

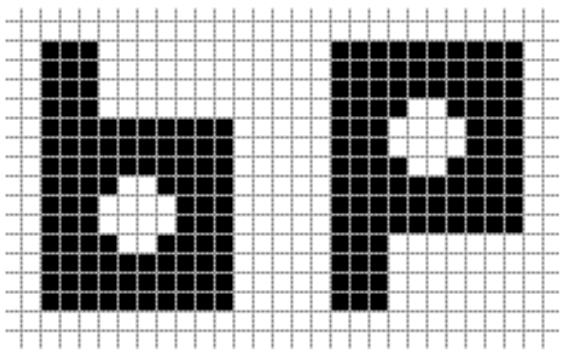
Shortest Judge Solution: 811 Bytes

Description

给定使用两种印章无重叠可旋转地打印出的字符画，统计每种印章的使用次数。



C type



S type

- $n, m \leq 1000$。

Solution

- 假设两种印章分别使用了 x 个和 y 个。
- 根据黑点总数可以列出 x 和 y 的一个等式。
- 根据洞的总数可以列出 x 和 y 的另一个等式。
- 联立两式解出 x 和 y。
- 时间复杂度 $O(nm)$。

I. Barbecue

Shortest Judge Solution: 864 Bytes

Description

给定一个长度为 n 的字符串 S，q 次询问，每次询问指定 S 的一个子串，两个人在该子串上进行博弈。

博弈双方轮流删去当前串开头或结尾的一个字符，碰到回文串的人输。

预测两人都按最优策略操作时最终谁会获胜。

- $n, q \leq 10^6$。

Solution

- 首先通过 Hash 等方式 $O(1)$ 特判起始串为回文串的情况。
- 对于接下来任意一个局面，先手操作前一定不是回文串。
- 若先手无法进行任何操作，则说明无论删去开头还是结尾都会得到回文串。
- 容易发现满足条件的串只能形如 `ab`, `abab`, `ababab`, ...
- 这说明终止态的长度一定是偶数，因此输赢只和起始串长度的奇偶性有关。
- 时间复杂度 $O(n+q)$。

J. Frog

Shortest Judge Solution: 1463 Bytes

Description

给定圆心为 $(0,0)$ 且半径为 1 的圆的圆周上的两个点 S, T，用最少的步数从 S 跳到 T。

需保证每一步的跳跃距离恰好为 1，且任意时刻都不能进入圆内。

- 测试数据组数 $\leq 10^4$。

Solution

- 不妨设 $S = (1, 0)$，且 T 的纵坐标非负，令 S 与 T 的圆心角为 $d°$。
- 若 $d = 0$，则需要 0 步。
- 若 $0 < d \leq 90$，则显然需要 2 步。
- 若 $d > 90$，那么如果 3 步可以到达，则 S、T 关于圆长度为 1 的切线之间的距离不能超过 1，解得 $d \leq 131$。
- 若 $d > 131$，则 $S \to (1, 1) \to (0, 1)$ 再加额外的 2 步可以 4 步到达 T。

F. Easy Fix

Shortest Judge Solution: 2197 Bytes

Description

给定长度为 n 的排列 p，令 A_i 表示 i 左边比 p_i 小的数字个数，B_i 表示 i 右边比 p_i 小的数字个数，$C_i = \min(A_i, B_i)$。

有 m 次独立的询问，每次询问给定 u 和 v，问如果交换 p_u 和 p_v，$\sum_{i=1}^{n} C_i$ 的值将会是多少。

- $n \leq 100000$。
- $m \leq 200000$。

Solution

- 使用树状数组预处理出每个位置初始的 A 值和 B 值。
- 不妨设 $u < v$，则 C_u 和 C_v 可以直接重新计算。
- 对于 u 左边以及 v 右边的位置，它们的 C 值不变。
- 对于 $[u+1, v-1]$ 这些位置，它们的 A 值和 B 值变化量为常数，每种类别的贡献都是二维数点问题。
- 离线询问后，扫描线 $+$ 树状数组即可。
- 时间复杂度 $O((n+m)\log n)$。

E. Easy Jump

Shortest Judge Solution: 1369 Bytes

Description

痛苦之路包含 n 个关卡，给定游戏规则以及每个关卡通过的概率，求按照最优策略保证角色不死的情况下所需的最小期望通关时间。

- $n \leq 1000$。
- 血量上限 H 满足 $2 \leq H \leq 9$。
- 蓝量上限 S 满足 $0 \leq S \leq 6$。

Solution

- 首先考虑 $S = 0$ 或者 $T1 \geq T2$ 的情况，此时不需要考虑耗蓝来回血。
- 为了使得期望通关时间最小，我们贪心地挑战关卡直至血量只有 1，然后回 1 点血。
- 令 $g_{i,j}$ 表示玩家位于第 i 关，血量为 j 时所需的最小期望通关时间，p 表示这一关的通过概率：
  - 如果 $j > 2$，则 $g_{i,j} = 1 + p \cdot g_{i+1,j} + (1-p)g_{i,j-1}$。
  - 如果 $j = 2$，则 $g_{i,j} = 1 + p \cdot g_{i+1,j} + (1-p)(g_{i,j} + T2)$。
- 注意部分转移是一个关于 $g_{i,j}$ 的等式，需要从中解出 $g_{i,j}$ 的值。

Solution

- 对于 $S > 0$ 且 $T1 < T2$ 的情况，需要考虑耗蓝来回血。
- 如果我们位于一个无法无限回蓝的关卡，我们贪心地挑战关卡直至血量只有 1，然后回 1 点血，回血时优先耗蓝。
- 令 $f_{i,j,k}$ 表示玩家位于第 i 关，蓝量为 j 且血量为 k 时所需的最小期望通关时间，p 表示这一关的通过概率：
  - 如果 $k > 2$，则 $f_{i,j,k} = 1 + p \cdot f_{i+1,j,k} + (1-p) f_{i,j,k-1}$。
  - 如果 $k = 2$ 且 $j > 0$，则

    $f_{i,j,k} = 1 + p \cdot f_{i+1,j,k} + (1-p)(f_{i,j-1,k} + T1)$。
  - 如果 $k = 2$ 且 $j = 0$，则

    $f_{i,j,k} = 1 + p \cdot f_{i+1,j,k} + (1-p)(f_{i,j,k} + T2)$。

Solution

- 如果我们位于一个可以无限回蓝的关卡，在这里提前回复若干点血可以帮助我们更好地攻略后面的关卡。
- 枚举在这一关提前回复了多少血来进行转移。
- 由于此时可以无限回蓝，因此该关卡任何状态的蓝量都是满的，状态数只有 $O(H)$。
- 时间复杂度 $O(nH(H+S))$。

K. Dynamic Reachability

Shortest Judge Solution: 2146 Bytes

Description

给定 n 个点 m 条边的 DAG，每条边的颜色是黑色或白色。

q 次操作，每次要么反转一条边的颜色，要么询问从 u 点出发只沿着黑边走能否到达 v 点。

- $n \leq 50000$。
- $m, q \leq 100000$。

Solution

- 离线处理操作，将操作按执行顺序分组，每组 k 个操作，依次处理每一组。
- 假设处理到了当前组，则接下来至多只有 k 条边的颜色会发生变化，令这些边为关键边。
- 令 $O(k)$ 条关键边的两端点以及该组所有询问涉及的两个点为关键点，总共不超过 $2k$ 个关键点。

Solution

- 在 DAG 上递推求出 $f_{i,j}$ 表示 i 点沿着黑色非关键边能否到达第 j 个关键点，通过位运算加速至 $O(n+m+\frac{(n+m)k}{w})$。
- 建立一张 $O(k)$ 个点的新图 G，G 中 $i \to j$ 有边当且仅当第 i 个关键点根据 f 的信息能到达第 j 个关键点。
- 得到缩小版的新图 G 后，对于每个询问，暴力将 $O(k)$ 条黑色关键边加入 G 中，然后 BFS 判断能否从 u 走到 v。
- BFS 的过程同样可以使用位运算加速至 $O(\frac{k^2}{w})$。

Solution

- 总时间复杂度
  $O\left(\frac{q}{k}\left(n+m+\frac{(n+m)k}{w}\right)+q\frac{k^2}{w}\right)=O(\frac{q(n+m+k^2)}{w}+\frac{q(n+m)}{k})$。
- 当 $k=32=\frac{w}{2}$ 时，总复杂度为 $O(\frac{q(n+m)}{w}+qw)$。
- 当 k 取得更大一些时，遍历 DAG 递推的次数将大幅减少，实际运行效果更好。

D. The Profiteer

Shortest Judge Solution: 1866 Bytes

Description

给定 n 个商品的原价、涨价后的价格以及价值，令 $f(x)$ 表示花不超过 x 元钱最多能买走总价值多少的商品。

给定 E 和 k，统计有多少区间 $[l, r]$ 满足将编号在该区间内的商品涨价出售时，期望收益 $\frac{f(1)+f(2)+\cdots+f(k)}{k} \leq E$。

- $n, k \leq 200000$。
- $n \times k \leq 10^7$。

Solution

- 随着区间的扩大，期望收益只会逐渐变小。
- 对于每个 l 找到 f_l，表示最小的 r 满足将 $[l, r]$ 涨价时期望收益不超过 E。
- $ans = \sum_{i=1}^{n} n - f_i + 1$。
- 不难发现 $f_1 \leq f_2 \leq \cdots \leq f_{n-1} \leq f_n$。
- 双指针枚举区间，0-1 背包求出期望收益，时间复杂度 $O(n^2 k)$，不能接受。

Solution

- 利用整体二分求出所有 f 的值。
- 令 $solve(l, r, dl, dr, V)$ 表示要求出 $f_l, f_{l+1}, \ldots, f_r$ 的值，每一项的取值范围是 $[dl, dr]$，且 $[l, r] \cup [dl, dr]$ 之外的所有商品都已经加入背包 V 中。
- 取 $dm = \lfloor \frac{dl+dr}{2} \rfloor$。
- 根据 $f_i \leq f_{i+1}$ 的性质，在 $[l, r]$ 二分找到最大的 m 满足 $f_m \leq dm$。
- 即二分一个 m 的值，判断涨价 $[m, dm]$ 后的期望收益是否不超过 E。

Solution

- 由于 V 中只缺失 $[l, r] \cup [dl, dr]$ 这些商品，将这些商品暴力加入背包即可。
- 朴素的实现共需要 $O((r - l + dr - dl) \log n)$ 次背包加入操作。
- 注意到 $[dl, dr]$ 中不在 $[l, r]$ 的那些商品价格恒定，可以在二分开始之前直接加入背包。
- 在二分的过程中，如果发现 m 偏小，即要往 $[m + 1, r]$ 继续二分，那么 $[l, m]$ 这些商品的价格恒定，可以直接加入背包。
- 同理当 m 偏大时，$[m, r]$ 也可以直接加入背包。
- 优化后总计需要 $O(r - l + dr - dl)$ 次背包加入操作。

Solution

- 找到最大的 m 满足 $f_m \le dm$ 后，得到子问题 $solve(l, m, dl, dm-1, V_1)$ 和 $solve(m+1, r, dm+1, dr, V_2)$。
- 对于 $[l, r] \cup [dl, dr]$ 的每个商品，如果它不在某个子问题的范围内，那么它的价格一定恒定，直接加入对应的背包即可。
- $solve$ 一共递归 $O(\log n)$ 层，每层的 $[l, r]$ 以及 $[dl, dr]$ 均不相交，因此每层的 $r - l + dr - dl$ 之和为 $O(n)$。
- 总计 $O(n \log n)$ 次背包加入操作，总时间复杂度 $O(nk \log n)$。

H. A=B

Shortest Judge Solution: 770 Bytes

Description

用 A=B 语言编写程序实现下述功能：

给两个以 “s” 分割的字符串，识别第二个串是否为第一个串的子串。

$O(L^3)$ Solution

- 每一轮开始把 s 串和 t 串复制到整个串后面。
- 判断 t 串是否为 s 串的前缀。
- 这一轮结束时消去 s 串开头的字符。
- 单次将 s 串复制到整个串末尾耗费 $O(|s|L)$ 代价，共 $|s|$ 轮，总复杂度 $O(L^3)$。

Intended Solution

- 仔细观察指令的性质：只允许对一个串的局部进行修改，几乎不允许随机访问。
- 频繁地复制串是需要规避的，因为单次将 s 串复制或移动到末尾的 “距离势能” 变化使得复杂度至少为 $O(|s|L)$。
- 提示我们可以将要比较的位安排在相邻的位置，例如 `abcdefSxyz` 可以安排成 `xaybzcdefS`，让 “`x`”,“`y`”,“`z`” 后面紧跟 “`a`”,“`b`”,“`c`”。

Intended Solution

- 观察两轮循环中位置的差别：
  - 第一轮：`Txaybzcdefs`
  - 第二轮：`aTxbyczdefs`
- 这里“`T`”表示类似开始状态的字符，“`T`”左边的字符表示 s 串已经被消耗掉的字符。
- 可以发现两轮的差别相当于“`x`”,“`y`”,“`z`”都分别和其后面的字符进行了交换，然后“`T`”再右移代表消耗一个字符。
- 复杂度分析：判断前缀 $O(|t|)$，两轮循环的状态转换 $O(|t|)$。

Intended Solution

- 先将读入串进行预处理，变成 s 串和 t 串交错的形式，耗费 $O(|t|L)$ 的代价。
- 下一页的程序会使用 “#” 符号来代表注释。
- 大致过程：

  abcabcScab→ZPabcabcYcab→ZPabcabcCYab→

  ZPCabcabcYab→Zfpabcabcyab→ZfaPbcabcYab→ $\cdots$ →

  ZfadbecPabcY→Zfadbecabcy

Intended Solution

```
S=XY
aX=Xa
bX=Xb
cX=Xc           # 把一个X字符放在开头
X=ZP            # X变为ZP，Z是后面程序的开始字符，P用于挡住从t复制过来的字符的前进
PA=dp
PB=ep
PC=fp           # P（大写P）成功挡住t过来的字符之后，右移并变为状态p（小写p）
pa=aP
pb=bP
pc=cP           # 状态p发现后面有s串的字符之后，右移并变为状态P
p=(return)0     # 状态p发现后面没有s串的字符，这说明s串比t串短，应该直接退出
aA=Aa
bA=Ab
cA=Ac
aB=Ba
bB=Bb
cB=Bc
aC=Ca
bC=Cb
cC=Cc           # 这九行是t串变身后的字符往左移的程序
Ya=AY
Yb=BY
Yc=CY           # 这三行是Y字符将t串开头的字符变身，配合上面九行往左丢。
P=              # 垃圾回收
```

Intended Solution

- 接下来，字符 “z” 启动一轮从左至右扫一遍的判断。
- 一边判断一边对字符进行交换，从而变成下一轮的状态。
- 总共 $|s|$ 轮，每轮 $O(|t|)$ 次指令执行。

Intended Solution

```
Z=T1          # 循环启动，1代表目前还没有遇见前缀不匹配的情况，0代表已经不是前缀了
1da=ad1
1eb=be1
1fc=cf1       # 字符匹配，1状态保持并顺便将字符swap
1a=(return)1
1b=(return)1
1c=(return)1
1Y=(return)1 # 1状态发现后面的字符是s串的字符或者s串结尾，已经能确定t串是目前s串的前缀，直接退出
             # 有可能1状态后面全是t串字符，即目前的s串已经比t串小了，但是这种可能不被包含在上述4行

1=0           # 剩余情况代表有字符不匹配，变成0状态
0da=ad0
0db=bd0
0dc=cd0
0ea=ae0
0eb=be0
0ec=ce0
0fa=af0
0fb=bf0
0fc=cf0       # 枚举后面9种情况 swap并右移
0=            # 0右边再无需要交换的字符，直接销毁。（可能会遇到0右边全剩t串字符的情况，但不需要管）
Ta=Z
Tb=Z
Tc=Z          # 消耗一个s串字符并返回循环开始状态
T=(return)0  # s串字符全部消耗完都没有return 1过，说明t串不是s串子串
```

完整 A=B 程序

```
S=XY
aX=Xa
bX=Xb
cX=Xc
X=ZP
PA=dp
PB=ep
PC=fp
pa=aP
pb=bP
pc=cP
p=(return)0
aA=Aa
bA=Ab
cA=Ac
aB=Ba
bB=Bb
cB=Bc
aC=Ca
bC=Cb
cC=Cc
Ya=AY
Yb=BY
Yc=CY
P=
Z=T1
1da=ad1
1eb=be1
1fc=cf1
1a=(return)1
1b=(return)1
1c=(return)1
1Y=(return)1
1=0
0da=ad0
0db=bd0
0dc=cd0
0ea=ae0
0eb=be0
0ec=ce0
0fa=af0
0fb=bf0
0fc=cf0
0=
Ta=Z
Tb=Z
Tc=Z
T=(return)0
```

Thank you!