2018 Multi-University Training Contest 3
HDU Contest

Claris

Hangzhou Dianzi University

2018 年 7 月 30 日

A. Ascending Rating

给定一个序列 $a[1..n]$，对于每个长度为 m 的连续子区间，求出区间 a 的最大值以及从左往右扫描该区间时 a 的最大值的变化次数。

A. Ascending Rating

给定一个序列 $a[1..n]$，对于每个长度为 m 的连续子区间，求出区间 a 的最大值以及从左往右扫描该区间时 a 的最大值的变化次数。

- $1 \leq m \leq n \leq 10^7$。

A. Ascending Rating

给定一个序列 $a[1..n]$，对于每个长度为 m 的连续子区间，求出区间 a 的最大值以及从左往右扫描该区间时 a 的最大值的变化次数。

- $1 \leq m \leq n \leq 10^7$。
- Shortest judge solution: 534 bytes

A. Ascending Rating

- 按照 r 从 m 到 n 的顺序很难解决这个问题。

A. Ascending Rating

- 按照 r 从 m 到 n 的顺序很难解决这个问题。
- 考虑按照 r 从 n 到 m 的顺序倒着求出每个区间的答案。

A. Ascending Rating

- 按照 r 从 m 到 n 的顺序很难解决这个问题。
- 考虑按照 r 从 n 到 m 的顺序倒着求出每个区间的答案。
- 按照滑窗最大值的经典方法维护 a 的单调队列，那么队列中的元素个数就是最大值的变化次数。

A. Ascending Rating

- 按照 r 从 m 到 n 的顺序很难解决这个问题。
- 考虑按照 r 从 n 到 m 的顺序倒着求出每个区间的答案。
- 按照滑窗最大值的经典方法维护 a 的单调队列，那么队列中的元素个数就是最大值的变化次数。
- 时间复杂度 $O(n)$。

B. Cut The String

给定一个字符串 $S[1..n]$，m 次询问指定它的一个连续子串。

B. Cut The String

给定一个字符串 $S[1..n]$，m 次询问指定它的一个连续子串。

你需要统计该子串有多少条分割线满足这条分割线前后两个字符串都是回文串。

B. Cut The String

给定一个字符串 $S[1..n]$，m 次询问指定它的一个连续子串。

你需要统计该子串有多少条分割线满足这条分割线前后两个字符串都是回文串。

- $1 \leq n, m \leq 10^5$。

B. Cut The String

给定一个字符串 $S[1..n]$，m 次询问指定它的一个连续子串。

你需要统计该子串有多少条分割线满足这条分割线前后两个字符串都是回文串。

- $1 \leq n, m \leq 10^5$。
- Shortest judge solution: 2928 bytes

B. Cut The String

- 一个字符串的所有回文后缀一定是它的最长回文后缀的所有border。

B. Cut The String

- 一个字符串的所有回文后缀一定是它的最长回文后缀的所有 border。
- 一个字符串的所有 border 按照长度排序后可以用 $O(\log n)$ 个等差数列表示。

B. Cut The String

- 一个字符串的所有回文后缀一定是它的最长回文后缀的所有 border。
- 一个字符串的所有 border 按照长度排序后可以用 $O(\log n)$ 个等差数列表示。
- 利用回文树可以直接得到每个前缀的最长回文后缀，稍加修改即可得到 $O(\log n)$ 个等差数列表示的所有回文后缀。

B. Cut The String

- 用两棵回文树预处理出每个前缀的回文后缀集合 f_i 以及每个后缀的回文前缀集合 g_i。

B. Cut The String

- 用两棵回文树预处理出每个前缀的回文后缀集合 f_i 以及每个后缀的回文前缀集合 g_i。
- 对于一个询问 l, r，枚举 g_l 以及 f_r 的两个等差数列，扩展欧几里得求出方案数，时间复杂度 $O(m \log^3 n)$，不能接受。

B. Cut The String

- 用两棵回文树预处理出每个前缀的回文后缀集合 f_i 以及每个后缀的回文前缀集合 g_i。
- 对于一个询问 l, r，枚举 g_l 以及 f_r 的两个等差数列，扩展欧几里得求出方案数，时间复杂度 $O(m \log^3 n)$，不能接受。
- 在扩展欧几里得之前根据上下界粗判即可去掉一个 $O(\log n)$，因为只有 $O(\log n)$ 对等差数列的上下界相交。

B. Cut The String

- 用两棵回文树预处理出每个前缀的回文后缀集合 f_i 以及每个后缀的回文前缀集合 g_i。
- 对于一个询问 l, r，枚举 g_l 以及 f_r 的两个等差数列，扩展欧几里得求出方案数，时间复杂度 $O(m \log^3 n)$，不能接受。
- 在扩展欧几里得之前根据上下界粗判即可去掉一个 $O(\log n)$，因为只有 $O(\log n)$ 对等差数列的上下界相交。
- 时间复杂度 $O(m \log^2 n)$。

C. Dynamic Graph Matching

给定一个 n 个点的无向图，m 次加边或者删边操作。

C. Dynamic Graph Matching

给定一个 n 个点的无向图，m 次加边或者删边操作。

在每次操作后统计有多少个匹配包含 $k = 1, 2, ..., \frac{n}{2}$ 条边。

C. Dynamic Graph Matching

给定一个 n 个点的无向图，m 次加边或者删边操作。

在每次操作后统计有多少个匹配包含 $k = 1, 2, ..., \frac{n}{2}$ 条边。

- $2 \leq n \leq 10, 1 \leq m \leq 30000$。

C. Dynamic Graph Matching

给定一个 n 个点的无向图，m 次加边或者删边操作。

在每次操作后统计有多少个匹配包含 $k = 1, 2, ..., \frac{n}{2}$ 条边。

- $2 \leq n \leq 10, 1 \leq m \leq 30000$。
- Shortest judge solution: 770 bytes

C. Dynamic Graph Matching

- 设 $f[i][S]$ 表示前 i 次操作之后，S 集合的点已经匹配的方案数。

C. Dynamic Graph Matching

- 设 $f[i][S]$ 表示前 i 次操作之后，S 集合的点已经匹配的方案数。
- 对于加边操作，显然 $f[i][S] = f[i-1][S] + f[i-1][S-u-v]$。

C. Dynamic Graph Matching

- 设 $f[i][S]$ 表示前 i 次操作之后，S 集合的点已经匹配的方案数。
- 对于加边操作，显然 $f[i][S] = f[i-1][S] + f[i-1][S-u-v]$。
- i 这一维可以省略，从大到小遍历 S，$f[S] += f[S-u-v]$。

C. Dynamic Graph Matching

- 设 $f[i][S]$ 表示前 i 次操作之后，S 集合的点已经匹配的方案数。
- 对于加边操作，显然 $f[i][S] = f[i-1][S] + f[i-1][S-u-v]$。
- i 这一维可以省略，从大到小遍历 S，$f[S] += f[S-u-v]$。
- 对于删边操作，注意到加边操作的顺序不影响结果，可以假设第 $i-1$ 次操作是加入要删除的边。

C. Dynamic Graph Matching

- 设 $f[i][S]$ 表示前 i 次操作之后，S 集合的点已经匹配的方案数。
- 对于加边操作，显然 $f[i][S] = f[i-1][S] + f[i-1][S-u-v]$。
- i 这一维可以省略，从大到小遍历 S，$f[S] += f[S-u-v]$。
- 对于删边操作，注意到加边操作的顺序不影响结果，可以假设第 $i-1$ 次操作是加入要删除的边。
- 将加边操作的更新倒过来，得到：从小到大遍历 S，$f[S] -= f[S-u-v]$。

C. Dynamic Graph Matching

- 设 $f[i][S]$ 表示前 i 次操作之后，S 集合的点已经匹配的方案数。
- 对于加边操作，显然 $f[i][S] = f[i-1][S] + f[i-1][S-u-v]$。
- i 这一维可以省略，从大到小遍历 S，$f[S] += f[S-u-v]$。
- 对于删边操作，注意到加边操作的顺序不影响结果，可以假设第 $i-1$ 次操作是加入要删除的边。
- 将加边操作的更新倒过来，得到：从小到大遍历 S，$f[S] -= f[S-u-v]$。
- 时间复杂度 $O(m2^n)$。

D. Euler Function

给定 k，求第 k 小的数 n，满足 $\varphi(n)$ 是合数。

D. Euler Function

给定 k，求第 k 小的数 n，满足 $\varphi(n)$ 是合数。

- $1 \leq k \leq 10^9$。

D. Euler Function

给定 k，求第 k 小的数 n，满足 $\varphi(n)$ 是合数。

- $1 \leq k \leq 10^9$。
- Shortest judge solution: 150 bytes

D. Euler Function

- 显然 $\varphi(1) = 1$ 不是合数，只考虑 $n \geq 2$ 的情况。

D. Euler Function

- 显然 $\varphi(1) = 1$ 不是合数，只考虑 $n \geq 2$ 的情况。
- 设 $n = p_1^{k_1} p_2^{k_2} ... p_m^{k_m}$ ，则
  $\varphi(n) = p_1^{k_1-1}(p_1 - 1)p_2^{k_2-1}(p_2 - 1)...p_m^{k_m-1}(p_m - 1)$。

D. Euler Function

- 显然 $\varphi(1) = 1$ 不是合数，只考虑 $n \geq 2$ 的情况。
- 设 $n = p_1^{k_1} p_2^{k_2} ... p_m^{k_m}$ ，则
  $\varphi(n) = p_1^{k_1-1}(p_1 - 1)p_2^{k_2-1}(p_2 - 1)...p_m^{k_m-1}(p_m - 1)$。
- 若 2 的指数不超过 1，那么 2 不影响 $\varphi(n)$ 的素性。

D. Euler Function

- 显然 $\varphi(1) = 1$ 不是合数，只考虑 $n \geq 2$ 的情况。
- 设 $n = p_1^{k_1} p_2^{k_2} ... p_m^{k_m}$ ，则
  $\varphi(n) = p_1^{k_1-1}(p_1 - 1)p_2^{k_2-1}(p_2 - 1)...p_m^{k_m-1}(p_m - 1)$。
- 若 2 的指数不超过 1，那么 2 不影响 $\varphi(n)$ 的素性。
- 若 2 的指数为 2，那么对 $\varphi(n)$ 贡献一个 2。

D. Euler Function

- 显然 $\varphi(1) = 1$ 不是合数，只考虑 $n \geq 2$ 的情况。
- 设 $n = p_1^{k_1} p_2^{k_2} ... p_m^{k_m}$ ，则
  $\varphi(n) = p_1^{k_1-1}(p_1 - 1)p_2^{k_2-1}(p_2 - 1)...p_m^{k_m-1}(p_m - 1)$。
- 若 2 的指数不超过 1，那么 2 不影响 $\varphi(n)$ 的素性。
- 若 2 的指数为 2，那么对 $\varphi(n)$ 贡献一个 2。
- 若 2 的指数 ≥ 3，那么对 $\varphi(n)$ 至少贡献一个 4，此时 $\varphi(n)$ 一定是合数。

D. Euler Function

- 对于除 2 之外的质数 p，如果指数 ≥ 2，那么至少贡献了 $(p-1)p$。

D. Euler Function

- 对于除 2 之外的质数 p，如果指数 ≥ 2，那么至少贡献了 $(p-1)p$。
- 如果除 2 之外还有至少两个质数 p, q，那么至少贡献了 $(p-1)(q-1)$。

D. Euler Function

- 对于除 2 之外的质数 p，如果指数 ≥ 2，那么至少贡献了 $(p-1)p$。
- 如果除 2 之外还有至少两个质数 p, q，那么至少贡献了 $(p-1)(q-1)$。
- 所以这两种情况下 $\varphi(n)$ 一定是合数。

D. Euler Function

- 剩下的情况只能是以下几种之一，其中 p 是大于 2 的质数：

D. Euler Function

- 剩下的情况只能是以下几种之一，其中 p 是大于 2 的质数：
- $n = 2, 4$，显然是素数。

D. Euler Function

- 剩下的情况只能是以下几种之一，其中 p 是大于 2 的质数：
- $n = 2, 4$，显然是素数。
- $n = p$，当 $p > 3$ 时一定是偶合数，而当 $p = 3, n = 3$ 时是偶素数。

D. Euler Function

- 剩下的情况只能是以下几种之一，其中 p 是大于 2 的质数：
- $n = 2, 4$，显然是素数。
- $n = p$，当 $p > 3$ 时一定是偶合数，而当 $p = 3, n = 3$ 时是偶素数。
- $n = 2p$，同理当 $p > 3$ 时一定是偶合数，而当 $p = 3, n = 6$ 时是偶素数。

D. Euler Function

- 剩下的情况只能是以下几种之一，其中 p 是大于 2 的质数：
- $n = 2, 4$，显然是素数。
- $n = p$，当 $p > 3$ 时一定是偶合数，而当 $p = 3, n = 3$ 时是偶素数。
- $n = 2p$，同理当 $p > 3$ 时一定是偶合数，而当 $p = 3, n = 6$ 时是偶素数。
- $n = 4p$，那么 $\varphi(n)$ 中至少有两个 2，所以一定是合数。

D. Euler Function

- 综上所述，当且仅当 $n = 1, 2, 3, 4, 6$ 时，$\varphi(n)$ 不是合数。

D. Euler Function

- 综上所述，当且仅当 $n = 1, 2, 3, 4, 6$ 时，$\varphi(n)$ 不是合数。
- $ans(k) = k + 4 + [k > 1]$。

D. Euler Function

- 综上所述，当且仅当 $n = 1, 2, 3, 4, 6$ 时，$\varphi(n)$ 不是合数。
- $ans(k) = k + 4 + [k > 1]$。
- 时间复杂度 $O(1)$。

E. Find The Submatrix

给定一个 $n \times m$ 的矩阵，可以将其中不超过 A 个位置的值设为 0。

E. Find The Submatrix

给定一个 $n \times m$ 的矩阵，可以将其中不超过 A 个位置的值设为 0。

找到不超过 B 个不相交的列数都为 m 的连续子矩阵，使得和最大。

E. Find The Submatrix

给定一个 $n \times m$ 的矩阵，可以将其中不超过 A 个位置的值设为 0。

找到不超过 B 个不相交的列数都为 m 的连续子矩阵，使得和最大。

- $1 \leq n \leq 100, 1 \leq m \leq 3000, 0 \leq A \leq 10000, 1 \leq B \leq 3$。

E. Find The Submatrix

给定一个 $n \times m$ 的矩阵，可以将其中不超过 A 个位置的值设为 0。

找到不超过 B 个不相交的列数都为 m 的连续子矩阵，使得和最大。

- $1 \leq n \leq 100, 1 \leq m \leq 3000, 0 \leq A \leq 10000, 1 \leq B \leq 3$。
- Shortest judge solution: 1746 bytes

E. Find The Submatrix

- 对于每一行来说，最优方案一定是将最小的若干个数修改为 0，设 $a[i][j]$ 表示第 i 行修改前 j 小的数的和，则 $a[i][j] - a[i][j+1] \leq a[i][j+1] - a[i][j+2]$。

E. Find The Submatrix

- 对于每一行来说，最优方案一定是将最小的若干个数修改为 0，设 $a[i][j]$ 表示第 i 行修改前 j 小的数的和，则 $a[i][j] - a[i][j+1] \leq a[i][j+1] - a[i][j+2]$。
- 设 $f[i][j][0][k]$ 表示考虑前 i 行，选了完整的 j 个连续子矩阵，修改了 k 次，且选择第 i 行时的最大和。
  设 $f[i][j][1][k]$ 表示考虑前 i 行，选了完整的 j 个连续子矩阵，修改了 k 次，且不选择第 i 行时的最大和。

E. Find The Submatrix

- 对于每一行来说，最优方案一定是将最小的若干个数修改为 0，设 $a[i][j]$ 表示第 i 行修改前 j 小的数的和，则 $a[i][j] - a[i][j+1] \leq a[i][j+1] - a[i][j+2]$。
- 设 $f[i][j][0][k]$ 表示考虑前 i 行，选了完整的 j 个连续子矩阵，修改了 k 次，且选择第 i 行时的最大和。
  设 $f[i][j][1][k]$ 表示考虑前 i 行，选了完整的 j 个连续子矩阵，修改了 k 次，且不选择第 i 行时的最大和。
- 初始状态：$f[0][0][1][0] = 0$。

E. Find The Submatrix

- 对于某个 $f[i][j][0][k]$，要么和上一行连起来，要么上一行不选并新开一个连续子矩阵，故 $f[i][j][0][k] = \max(f[i-1][j][0][x] + a[i][k-x], f[i-1][j-1][1][x] + a[i][k-x])$。

E. Find The Submatrix

- 对于某个 $f[i][j][0][k]$，要么和上一行连起来，要么上一行不选并新开一个连续子矩阵，故 $f[i][j][0][k] = \max(f[i-1][j][0][x] + a[i][k-x], f[i-1][j-1][1][x] + a[i][k-x])$。
- 对于某个 $f[i][j][1][k]$，上一行要么选要么不选，故 $f[i][j][1][k] = \max(f[i-1][j][0][k], f[i-1][j][1][k])$。

E. Find The Submatrix

- 对于某个 $f[i][j][0][k]$，要么和上一行连起来，要么上一行不选并新开一个连续子矩阵，故 $f[i][j][0][k] = \max(f[i-1][j][0][x] + a[i][k-x], f[i-1][j-1][1][x] + a[i][k-x])$。
- 对于某个 $f[i][j][1][k]$，上一行要么选要么不选，故 $f[i][j][1][k] = \max(f[i-1][j][0][k], f[i-1][j][1][k])$。
- 直接转移时间复杂度为 $O(nmAB)$，不能接受。

E. Find The Submatrix

- 将 $f[i][j][0][k]$ 两部分转移分别求出最优值然后取 max。

E. Find The Submatrix

- 将 $f[i][j][0][k]$ 两部分转移分别求出最优值然后取 max。
- 需要优化的部分形如 $g[k] = \max(f[x] + w[k-x])$，其中 $w[j] - w[j+1] \leq w[j+1] - w[j+2]$。

E. Find The Submatrix

- 将 $f[i][j][0][k]$ 两部分转移分别求出最优值然后取 max。
- 需要优化的部分形如 $g[k] = \max(f[x] + w[k-x])$，其中 $w[j] - w[j+1] \leq w[j+1] - w[j+2]$。
- 考虑对于固定的 k，有两个决策 i, j 作为 x，其中 $i < j$ 且 i 比 j 劣，则 $f[i] + w[k-i] \leq f[j] + w[k-j]$。

E. Find The Submatrix

- 将 $f[i][j][0][k]$ 两部分转移分别求出最优值然后取 max。
- 需要优化的部分形如 $g[k] = \max(f[x] + w[k-x])$，其中 $w[j] - w[j+1] \leq w[j+1] - w[j+2]$。
- 考虑对于固定的 k，有两个决策 i, j 作为 x，其中 $i < j$ 且 i 比 j 劣，则 $f[i] + w[k-i] \leq f[j] + w[k-j]$。
- 若 k 增加 1，因为 $k-i > k-j$，所以 $w[k-i]$ 的增量小于 $w[k-j]$ 的增量，i 将永远比 j 劣。

E. Find The Submatrix

- 将 $f[i][j][0][k]$ 两部分转移分别求出最优值然后取 max。
- 需要优化的部分形如 $g[k] = \max(f[x] + w[k-x])$，其中 $w[j] - w[j+1] \leq w[j+1] - w[j+2]$。
- 考虑对于固定的 k，有两个决策 i, j 作为 x，其中 $i < j$ 且 i 比 j 劣，则 $f[i] + w[k-i] \leq f[j] + w[k-j]$。
- 若 k 增加 1，因为 $k-i > k-j$，所以 $w[k-i]$ 的增量小于 $w[k-j]$ 的增量，i 将永远比 j 劣。
- 所以最优决策 x 具有决策单调性，分治求解即可。

E. Find The Submatrix

- 将 $f[i][j][0][k]$ 两部分转移分别求出最优值然后取 max。
- 需要优化的部分形如 $g[k] = \max(f[x] + w[k-x])$，其中 $w[j] - w[j+1] \leq w[j+1] - w[j+2]$。
- 考虑对于固定的 k，有两个决策 i, j 作为 x，其中 $i < j$ 且 i 比 j 劣，则 $f[i] + w[k-i] \leq f[j] + w[k-j]$。
- 若 k 增加 1，因为 $k-i > k-j$，所以 $w[k-i]$ 的增量小于 $w[k-j]$ 的增量，i 将永远比 j 劣。
- 所以最优决策 x 具有决策单调性，分治求解即可。
- 时间复杂度 $O(nm \log m + nAB \log A)$。

F. Grab The Tree

给定一棵 n 个点的树，每个点有权值。两个人玩游戏，先手需要占领若干不相邻的点，然后后手占领剩下所有点。

F. Grab The Tree

给定一棵 n 个点的树，每个点有权值。两个人玩游戏，先手需要占领若干不相邻的点，然后后手占领剩下所有点。

每个人的得分为占领的点权异或和，分高的获胜。问最优策略下游戏的结果。

F. Grab The Tree

给定一棵 n 个点的树，每个点有权值。两个人玩游戏，先手需要占领若干不相邻的点，然后后手占领剩下所有点。

每个人的得分为占领的点权异或和，分高的获胜。问最优策略下游戏的结果。

- $1 \leq n \leq 10^5$。

F. Grab The Tree

给定一棵 n 个点的树，每个点有权值。两个人玩游戏，先手需要占领若干不相邻的点，然后后手占领剩下所有点。

每个人的得分为占领的点权异或和，分高的获胜。问最优策略下游戏的结果。

- $1 \leq n \leq 10^5$。
- Shortest judge solution: 235 bytes

F. Grab The Tree

- 设 sum 为所有点权的异或和，A 为先手得分，B 为后手得分。

F. Grab The Tree

- 设 sum 为所有点权的异或和，A 为先手得分，B 为后手得分。
- 若 $sum = 0$，则 $A = B$，故无论如何都是平局。

F. Grab The Tree

- 设 sum 为所有点权的异或和，A 为先手得分，B 为后手得分。
- 若 $sum = 0$，则 $A = B$，故无论如何都是平局。
- 否则考虑 sum 二进制下最高的 1 所在那位，一定有奇数个点那一位为 1。

F. Grab The Tree

- 设 sum 为所有点权的异或和，A 为先手得分，B 为后手得分。
- 若 $sum = 0$，则 $A = B$，故无论如何都是平局。
- 否则考虑 sum 二进制下最高的 1 所在那位，一定有奇数个点那一位为 1。
- 若先手拿走任意一个那一位为 1 的点，则 B 该位为 0，故先手必胜。

F. Grab The Tree

- 设 sum 为所有点权的异或和，A 为先手得分，B 为后手得分。
- 若 $sum = 0$，则 $A = B$，故无论如何都是平局。
- 否则考虑 sum 二进制下最高的 1 所在那位，一定有奇数个点那一位为 1。
- 若先手拿走任意一个那一位为 1 的点，则 B 该位为 0，故先手必胜。
- 时间复杂度 $O(n)$。

G. Interstellar Travel

给定平面上 n 个点，起点横坐标最小，终点横坐标最大。

G. Interstellar Travel

给定平面上 n 个点，起点横坐标最小，终点横坐标最大。

每次可以飞到一个横坐标严格更大的点，代价为两个坐标的叉积。

G. Interstellar Travel

给定平面上 n 个点，起点横坐标最小，终点横坐标最大。

每次可以飞到一个横坐标严格更大的点，代价为两个坐标的叉积。

求起点到终点总代价最小的飞行路线，并输出字典序最小的路线。

G. Interstellar Travel

给定平面上 n 个点，起点横坐标最小，终点横坐标最大。

每次可以飞到一个横坐标严格更大的点，代价为两个坐标的叉积。

求起点到终点总代价最小的飞行路线，并输出字典序最小的路线。

- $2 \leq n \leq 200000$。

G. Interstellar Travel

给定平面上 n 个点，起点横坐标最小，终点横坐标最大。

每次可以飞到一个横坐标严格更大的点，代价为两个坐标的叉积。

求起点到终点总代价最小的飞行路线，并输出字典序最小的路线。

- $2 \leq n \leq 200000$。
- Shortest judge solution: 979 bytes

G. Interstellar Travel

- 显然坐标相同的点里只保留编号最小的点最优。

G. Interstellar Travel

- 显然坐标相同的点里只保留编号最小的点最优。
- 将起点到终点的路径补全为终点往下走到无穷远处，再往左走到起点正下方，再往上回到起点。

G. Interstellar Travel

- 显然坐标相同的点里只保留编号最小的点最优。
- 将起点到终点的路径补全为终点往下走到无穷远处，再往左走到起点正下方，再往上回到起点。
- 任意路径中回到起点部分的代价相同，观察代价和的几何意义，就是走过部分的面积的相反数。

G. Interstellar Travel

- 显然坐标相同的点里只保留编号最小的点最优。
- 将起点到终点的路径补全为终点往下走到无穷远处，再往左走到起点正下方，再往上回到起点。
- 任意路径中回到起点部分的代价相同，观察代价和的几何意义，就是走过部分的面积的相反数。
- 代价和最小等价于面积最大，故一定是沿着上凸壳行走。

G. Interstellar Travel

- 显然坐标相同的点里只保留编号最小的点最优。
- 将起点到终点的路径补全为终点往下走到无穷远处，再往左走到起点正下方，再往上回到起点。
- 任意路径中回到起点部分的代价相同，观察代价和的几何意义，就是走过部分的面积的相反数。
- 代价和最小等价于面积最大，故一定是沿着上凸壳行走。
- 显然起点、终点、凸壳的拐点必须要作为降落点。

G. Interstellar Travel

- 显然坐标相同的点里只保留编号最小的点最优。
- 将起点到终点的路径补全为终点往下走到无穷远处，再往左走到起点正下方，再往上回到起点。
- 任意路径中回到起点部分的代价相同，观察代价和的几何意义，就是走过部分的面积的相反数。
- 代价和最小等价于面积最大，故一定是沿着上凸壳行走。
- 显然起点、终点、凸壳的拐点必须要作为降落点。
- 对于共线的点 $a_1, a_2, \ldots, a_m$，若一个点 i 的编号是 $[i, m]$ 中最小的，那么在此处降落可以最小化字典序。

G. Interstellar Travel

- 显然坐标相同的点里只保留编号最小的点最优。
- 将起点到终点的路径补全为终点往下走到无穷远处，再往左走到起点正下方，再往上回到起点。
- 任意路径中回到起点部分的代价相同，观察代价和的几何意义，就是走过部分的面积的相反数。
- 代价和最小等价于面积最大，故一定是沿着上凸壳行走。
- 显然起点、终点、凸壳的拐点必须要作为降落点。
- 对于共线的点 $a_1, a_2, \ldots, a_m$，若一个点 i 的编号是 $[i, m]$ 中最小的，那么在此处降落可以最小化字典序。
- 时间复杂度 $O(n \log n)$。

H. Monster Hunter

给定一棵 n 个点的树，除 1 外每个点有一只怪兽，打败它需要先消耗 a_i 点 HP，再恢复 b_i 点 HP。

H. Monster Hunter

给定一棵 n 个点的树，除 1 外每个点有一只怪兽，打败它需要先消耗 a_i 点 HP，再恢复 b_i 点 HP。

求从 1 号点出发按照最优策略打败所有怪兽一开始所需的最少 HP。

H. Monster Hunter

给定一棵 n 个点的树，除 1 外每个点有一只怪兽，打败它需要先消耗 a_i 点 HP，再恢复 b_i 点 HP。

求从 1 号点出发按照最优策略打败所有怪兽一开始所需的最少 HP。

- $2 \leq n \leq 100000$。

H. Monster Hunter

给定一棵 n 个点的树，除 1 外每个点有一只怪兽，打败它需要先消耗 a_i 点 HP，再恢复 b_i 点 HP。

求从 1 号点出发按照最优策略打败所有怪兽一开始所需的最少 HP。

- $2 \leq n \leq 100000$。
- Shortest judge solution: 1459 bytes

H. Monster Hunter

- 以 1 为根将树转化成有根树，那么每只怪兽要在父亲怪兽被击败后才能被击败。

H. Monster Hunter

- 以 1 为根将树转化成有根树，那么每只怪兽要在父亲怪兽被击败后才能被击败。
- 考虑简化版问题：忽略父亲的限制，求最优的攻击顺序。

H. Monster Hunter

- 以 1 为根将树转化成有根树，那么每只怪兽要在父亲怪兽被击败后才能被击败。
- 考虑简化版问题：忽略父亲的限制，求最优的攻击顺序。
- 将怪兽分成两类：$a < b$ 的和 $a \geq b$ 的，前一类打完会加血，后一类打完会扣血，显然最优策略下应该先打第一类再打第二类。

H. Monster Hunter

- 以 1 为根将树转化成有根树，那么每只怪兽要在父亲怪兽被击败后才能被击败。
- 考虑简化版问题：忽略父亲的限制，求最优的攻击顺序。
- 将怪兽分成两类：$a < b$ 的和 $a \geq b$ 的，前一类打完会加血，后一类打完会扣血，显然最优策略下应该先打第一类再打第二类。
- 对于 $a < b$ 的怪兽，显然最优策略下应该按照 a 从小到大打。

H. Monster Hunter

- 对于 $a \geq b$ 的怪兽，考虑两只怪兽 i, j，先打 i 再打 j 的过程中血量会减少到 $HP + \min(-a_i, -a_i + b_i - a_j)$，因为 $a \geq b$，所以这等于 $HP - a_i - a_j + b_i$。同理先打 j 再打 i 的过程中血量会减少到 $HP - a_i - a_j + b_j$。

H. Monster Hunter

- 对于 $a \geq b$ 的怪兽，考虑两只怪兽 i, j，先打 i 再打 j 的过程中血量会减少到 $HP + \min(-a_i, -a_i + b_i - a_j)$，因为 $a \geq b$，所以这等于 $HP - a_i - a_j + b_i$。同理先打 j 再打 i 的过程中血量会减少到 $HP - a_i - a_j + b_j$。
- 可以发现按照任何顺序都只和 b 有关，最优策略下需要让血量尽可能多，因此要按照 b 从大到小打。

H. Monster Hunter

- 对于 $a \geq b$ 的怪兽，考虑两只怪兽 i, j，先打 i 再打 j 的过程中血量会减少到 $HP + \min(-a_i, -a_i + b_i - a_j)$，因为 $a \geq b$，所以这等于 $HP - a_i - a_j + b_i$。同理先打 j 再打 i 的过程中血量会减少到 $HP - a_i - a_j + b_j$。
- 可以发现按照任何顺序都只和 b 有关，最优策略下需要让血量尽可能多，因此要按照 b 从大到小打。
- 如此可以 $O(1)$ 比较任意两只怪兽应该先打谁，从而得到最优攻击顺序。

H. Monster Hunter

- 考虑原问题，求出忽略父亲限制后最优攻击顺序 $p_1, p_2, \ldots, p_n$。

H. Monster Hunter

- 考虑原问题，求出忽略父亲限制后最优攻击顺序 $p_1, p_2, \ldots, p_n$。
- 若 $p_1 = 1$，那么第一步打 p_1 一定最优。

H. Monster Hunter

- 考虑原问题，求出忽略父亲限制后最优攻击顺序 $p_1, p_2, \ldots, p_n$。
- 若 $p_1 = 1$，那么第一步打 p_1 一定最优。
- 若 $p_1 \neq 1$，那么在打完 p_1 的父亲 x 后，紧接着打 p_1 一定最优。直接将 p_1 和 x 两只怪兽合并，依然用 (a, b) 表示，并将 p_1 所有儿子的父亲改为 x 即可消除 p_1 的影响。

H. Monster Hunter

- 考虑原问题，求出忽略父亲限制后最优攻击顺序 $p_1, p_2, \ldots, p_n$。
- 若 $p_1 = 1$，那么第一步打 p_1 一定最优。
- 若 $p_1 \neq 1$，那么在打完 p_1 的父亲 x 后，紧接着打 p_1 一定最优。直接将 p_1 和 x 两只怪兽合并，依然用 (a, b) 表示，并将 p_1 所有儿子的父亲改为 x 即可消除 p_1 的影响。
- 上面两步将规模为 n 的问题化成了规模为 $n-1$ 的问题，重复操作直到只剩下一只怪兽即可求出最少所需血量。

H. Monster Hunter

- 需要高效支持修改一个怪兽，删除最优怪兽，以及修改父亲的操作。

H. Monster Hunter

- 需要高效支持修改一个怪兽，删除最优怪兽，以及修改父亲的操作。
- 对于最优怪兽，可以用堆维护。

H. Monster Hunter

- 需要高效支持修改一个怪兽，删除最优怪兽，以及修改父亲的操作。
- 对于最优怪兽，可以用堆维护。
- 对于修改父亲，可以用并查集维护。

H. Monster Hunter

- 需要高效支持修改一个怪兽，删除最优怪兽，以及修改父亲的操作。
- 对于最优怪兽，可以用堆维护。
- 对于修改父亲，可以用并查集维护。
- 时间复杂度 $O(n \log n)$。

I. Random Sequence

给定一个正整数序列 $a[1..n]$，每个数在 $[1, m]$ 之间，有些数已知，有些数未知。

I. Random Sequence

给定一个正整数序列 $a[1..n]$，每个数在 $[1, m]$ 之间，有些数已知，有些数未知。

求未知数随机填的情况下以下值的期望：

$$\prod_{i=1}^{n-3} v[\gcd(a_i, a_{i+1}, a_{i+2}, a_{i+3})]$$

I. Random Sequence

给定一个正整数序列 $a[1..n]$，每个数在 $[1, m]$ 之间，有些数已知，有些数未知。

求未知数随机填的情况下以下值的期望：

$$\prod_{i=1}^{n-3} v[\gcd(a_i, a_{i+1}, a_{i+2}, a_{i+3})]$$

- $4 \leq n \leq 100, 1 \leq m \leq 100$。

I. Random Sequence

给定一个正整数序列 $a[1..n]$，每个数在 $[1, m]$ 之间，有些数已知，有些数未知。

求未知数随机填的情况下以下值的期望：

$$\prod_{i=1}^{n-3} v[\gcd(a_i, a_{i+1}, a_{i+2}, a_{i+3})]$$

- $4 \le n \le 100, 1 \le m \le 100$。
- Shortest judge solution: 1414 bytes

I. Random Sequence

- 设 $f[i][x][y][z]$ 表示考虑前 i 个位置，
  $a_i = x, \gcd(a_i, a_{i-1}) = y, \gcd(a_i, a_{i-1}, a_{i-2}) = z$ 的期望。

I. Random Sequence

- 设 $f[i][x][y][z]$ 表示考虑前 i 个位置，
  $a_i = x, \gcd(a_i, a_{i-1}) = y, \gcd(a_i, a_{i-1}, a_{i-2}) = z$ 的期望。
- 枚举 a_{i+1} 的值转移即可。

I. Random Sequence

- 设 $f[i][x][y][z]$ 表示考虑前 i 个位置，
  $a_i = x, \gcd(a_i, a_{i-1}) = y, \gcd(a_i, a_{i-1}, a_{i-2}) = z$ 的期望。
- 枚举 a_{i+1} 的值转移即可。
- 时间复杂度 $O(nmS)$，其中 S 表示 (x, y, z) 的状态数。

I. Random Sequence

- 设 $f[i][x][y][z]$ 表示考虑前 i 个位置，$a_i = x, \gcd(a_i, a_{i-1}) = y, \gcd(a_i, a_{i-1}, a_{i-2}) = z$ 的期望。
- 枚举 a_{i+1} 的值转移即可。
- 时间复杂度 $O(nmS)$，其中 S 表示 (x, y, z) 的状态数。
- 显然合法状态中 $y|x, z|y$，当 $m = 100$ 时 $S = 1471$。

J. Rectangle Radar Scanner

给定平面上 n 个带信息的点，m 次询问一个平行坐标轴的矩形内所有点的不可减信息和。

J. Rectangle Radar Scanner

给定平面上 n 个带信息的点，m 次询问一个平行坐标轴的矩形内所有点的不可减信息和。

- $1 \leq n \leq 10^5, 1 \leq m \leq 10^6$。

J. Rectangle Radar Scanner

给定平面上 n 个带信息的点，m 次询问一个平行坐标轴的矩形内所有点的不可减信息和。

- $1 \leq n \leq 10^5, 1 \leq m \leq 10^6$。
- Shortest judge solution: 2778 bytes

J. Rectangle Radar Scanner

- 因为信息不可减，所以不能通过二维数点的方法差分得到答案。

J. Rectangle Radar Scanner

- 因为信息不可减，所以不能通过二维数点的方法差分得到答案。
- 将平面上的点按横坐标进行分治，在两侧递归处理所有没有覆盖中线的矩形询问。

J. Rectangle Radar Scanner

- 因为信息不可减，所以不能通过二维数点的方法差分得到答案。
- 将平面上的点按横坐标进行分治，在两侧递归处理所有没有覆盖中线的矩形询问。
- 对于覆盖当前中线的询问，拆成中线左右两个询问，那么两侧的信息没有重叠，而且矩形 4 个边界中有一个边界的限制消失了。

J. Rectangle Radar Scanner

- 问题转化为：求 $x \leq A, B \leq y \leq C$ 的所有点的信息和。

J. Rectangle Radar Scanner

- 问题转化为：求 $x \leq A, B \leq y \leq C$ 的所有点的信息和。
- 按横坐标将询问和点排序，依次加入每个点或者回答每个询问。

J. Rectangle Radar Scanner

- 问题转化为：求 $x \leq A, B \leq y \leq C$ 的所有点的信息和。
- 按横坐标将询问和点排序，依次加入每个点或者回答每个询问。
- 用线段树维护纵维度每个区间的信息和。

J. Rectangle Radar Scanner

- 问题转化为：求 $x \leq A, B \leq y \leq C$ 的所有点的信息和。
- 按横坐标将询问和点排序，依次加入每个点或者回答每个询问。
- 用线段树维护纵维度每个区间的信息和。
- 时间复杂度 $O((n \log n + m) \log n)$。

K. Transport Construction

给定第一象限内 n 个点，任意两点间连边，权值为两点坐标的点积。

K. Transport Construction

给定第一象限内 n 个点，任意两点间连边，权值为两点坐标的点积。

求这个完全图的最小生成树。

K. Transport Construction

给定第一象限内 n 个点，任意两点间连边，权值为两点坐标的点积。

求这个完全图的最小生成树。

- $2 \leq n \leq 100000$。

K. Transport Construction

给定第一象限内 n 个点，任意两点间连边，权值为两点坐标的点积。

求这个完全图的最小生成树。

- $2 \leq n \leq 100000$。
- Shortest judge solution: 3498 bytes

K. Transport Construction

- 考虑 Boruvka 算法求最小生成树的过程。

K. Transport Construction

- 考虑 Boruvka 算法求最小生成树的过程。
- 对于每个点寻找与它相连的边权最小的边，若边权相同则取另一个端点编号最小的。

K. Transport Construction

- 考虑 Boruvka 算法求最小生成树的过程。
- 对于每个点寻找与它相连的边权最小的边，若边权相同则取另一个端点编号最小的。
- 那么每个连通块要么是树，要么是二元环，最多 $\frac{n}{2}$ 个连通块。

K. Transport Construction

- 考虑 Boruvka 算法求最小生成树的过程。
- 对于每个点寻找与它相连的边权最小的边，若边权相同则取另一个端点编号最小的。
- 那么每个连通块要么是树，要么是二元环，最多 $\frac{n}{2}$ 个连通块。
- 将每个连通块缩点，反复迭代 $O(\log n)$ 轮，直至剩下一个点。

K. Transport Construction

- 问题转化为，给定 n 个带颜色的点，对于每个点询问和它异色的点中和它点积最小的点。

K. Transport Construction

- 问题转化为，给定 n 个带颜色的点，对于每个点询问和它异色的点中和它点积最小的点。
- 对颜色进行分治，假设当前考虑 $[l, r]$ 的所有颜色，递归分治 $[l, mid]$ 和 $[mid + 1, r]$，然后用 $[l, mid]$ 询问 $[mid + 1, r]$，再用 $[mid + 1, r]$ 询问 $[l, mid]$。

K. Transport Construction

- 问题转化为，给定 n 个带颜色的点，对于每个点询问和它异色的点中和它点积最小的点。
- 对颜色进行分治，假设当前考虑 $[l, r]$ 的所有颜色，递归分治 $[l, mid]$ 和 $[mid + 1, r]$，然后用 $[l, mid]$ 询问 $[mid + 1, r]$，再用 $[mid + 1, r]$ 询问 $[l, mid]$。
- 考虑点积的几何意义：向量投影的长度。将一条直线从原点开始往上移动，直到碰到一个点，此时碰到的点一定点积最小。

K. Transport Construction

- 只有下凸壳上的点会被直线碰到，将子区间的凸壳按照横坐标归并，同时将子区间的询问直线按照斜率归并，然后双指针即可。

K. Transport Construction

- 只有下凸壳上的点会被直线碰到，将子区间的凸壳按照横坐标归并，同时将子区间的询问直线按照斜率归并，然后双指针即可。
- 每轮分治时间复杂度 $O(n \log n)$，总时间复杂度 $O(n \log^2 n)$。

L. Visual Cube

按照一定格式画出一个 $a \times b \times c$ 的长方体。

L. Visual Cube

按照一定格式画出一个 $a \times b \times c$ 的长方体。

- $1 \leq a, b, c \leq 20$。

L. Visual Cube

按照一定格式画出一个 $a \times b \times c$ 的长方体。

- $1 \leq a, b, c \leq 20$。
- Shortest judge solution: 1015 bytes

L. Visual Cube

- 计算画布大小以及各个关键位置的坐标。

L. Visual Cube

- 计算画布大小以及各个关键位置的坐标。
- 按照格式将画布填充正确。

M. Walking Plan

给定一个 n 个点，m 条边的有向图，q 次询问 s 到 t 经过至少 k 条边的最短路。

M. Walking Plan

给定一个 n 个点，m 条边的有向图，q 次询问 s 到 t 经过至少 k 条边的最短路。

- $2 \leq n \leq 50, 1 \leq m, k \leq 10000, 1 \leq q \leq 100000$。

M. Walking Plan

给定一个 n 个点，m 条边的有向图，q 次询问 s 到 t 经过至少 k 条边的最短路。

- $2 \leq n \leq 50, 1 \leq m, k \leq 10000, 1 \leq q \leq 100000$。
- Shortest judge solution: 1220 bytes

M. Walking Plan

- 设 $G[i][j]$ 表示 i 一步到 j 的最短路，$f[t][i][j]$ 表示 i 经过恰好 t 条边到达 j 的最短路，则 $f[t][i][j] = \min(f[t-1][i][k] + G[k][j])$。

M. Walking Plan

- 设 $G[i][j]$ 表示 i 一步到 j 的最短路，$f[t][i][j]$ 表示 i 经过恰好 t 条边到达 j 的最短路，则
  $f[t][i][j] = \min(f[t-1][i][k] + G[k][j])$。
- f 具有更强的性质，即 $f[t][i][j] = \min(f[x][i][k] + f[t-x][k][j])$。

M. Walking Plan

- 设 $G[i][j]$ 表示 i 一步到 j 的最短路，$f[t][i][j]$ 表示 i 经过恰好 t 条边到达 j 的最短路，则 $f[t][i][j] = \min(f[t-1][i][k] + G[k][j])$。
- f 具有更强的性质，即 $f[t][i][j] = \min(f[x][i][k] + f[t-x][k][j])$。
- 将合并的过程看作乘法，则 $f[t] = f[x]f[t-x] = G^t$。

M. Walking Plan

- 注意到 $k \leq 10000$，设 $A = \lfloor \frac{k}{100} \rfloor, B = k \bmod 100$，将每个询问 s, t, k 拆成两步：
  1. 从 s 出发经过恰好 $100A$ 条边到达某个点 u。
  2. 从 u 出发经过至少 B 条边到达 t。

M. Walking Plan

- 注意到 $k \leq 10000$，设 $A = \lfloor \frac{k}{100} \rfloor, B = k \bmod 100$，将每个询问 s, t, k 拆成两步：
  1. 从 s 出发经过恰好 $100A$ 条边到达某个点 u。
  2. 从 u 出发经过至少 B 条边到达 t。
- 设 $a[i] = G^{100i}, b[i] = G^i$，那么 $a[i]$ 即为经过恰好 $100i$ 条边的最短路，对于 $b[i]$ 需要再进行一次 Floyd 得到至少 i 条边的最短路。

M. Walking Plan

- 注意到 $k \leq 10000$，设 $A = \lfloor \frac{k}{100} \rfloor, B = k \bmod 100$，将每个询问 s, t, k 拆成两步：
  1. 从 s 出发经过恰好 $100A$ 条边到达某个点 u。
  2. 从 u 出发经过至少 B 条边到达 t。
- 设 $a[i] = G^{100i}, b[i] = G^i$，那么 $a[i]$ 即为经过恰好 $100i$ 条边的最短路，对于 $b[i]$ 需要再进行一次 Floyd 得到至少 i 条边的最短路。
- 枚举中间点 u，则 $ans = \min(a[A][s][u] + b[B][u][t])$。

M. Walking Plan

- 注意到 $k \leq 10000$，设 $A = \lfloor \frac{k}{100} \rfloor, B = k \bmod 100$，将每个询问 s, t, k 拆成两步：
  1. 从 s 出发经过恰好 $100A$ 条边到达某个点 u。
  2. 从 u 出发经过至少 B 条边到达 t。
- 设 $a[i] = G^{100i}, b[i] = G^i$，那么 $a[i]$ 即为经过恰好 $100i$ 条边的最短路，对于 $b[i]$ 需要再进行一次 Floyd 得到至少 i 条边的最短路。
- 枚举中间点 u，则 $ans = \min(a[A][s][u] + b[B][u][t])$。
- 时间复杂度 $O(n^3\sqrt{k} + qn)$。

Thank you!