压位大法

Claris

Hangzhou Dianzi University

2017 年 7 月 25 日

Overview

- 在计算机中，数据是以二进制形式存储的。

Overview

- 在计算机中，数据是以二进制形式存储的。
- 机器字长 w 一般为 32 位或 64 位，这意味着我们可以 w 位一起运算，将程序运行时间除以 w。

Overview

- 在计算机中，数据是以二进制形式存储的。
- 机器字长 w 一般为 32 位或 64 位，这意味着我们可以 w 位一起运算，将程序运行时间除以 w。
- 最经典的应用就是 bitset。

Overview

- 在计算机中，数据是以二进制形式存储的。
- 机器字长 w 一般为 32 位或 64 位，这意味着我们可以 w 位一起运算，将程序运行时间除以 w。
- 最经典的应用就是 bitset。
- bitset 中每个位置要么是 0，要么是 1，单点操作复杂度 $O(1)$，整体操作复杂度 $O(\frac{n}{w})$。

求 1 的个数

- 在 bitset 中，我们常常需要对一个 32 位无符号整数求 1 的个数。

求 1 的个数

- 在 bitset 中，我们常常需要对一个 32 位无符号整数求 1 的个数。
- 直接暴力计算是 $O(w)$ 的，不能体现优势。

求 1 的个数

- 在 bitset 中，我们常常需要对一个 32 位无符号整数求 1 的个数。
- 直接暴力计算是 $O(w)$ 的，不能体现优势。
- 查表法：预处理出 $[0, 2^{16})$ 内每个数的 1 的个数 cnt_i，$cnt_i = cnt_{i>>1} + (i\&1)$。

求 1 的个数

- 在 bitset 中，我们常常需要对一个 32 位无符号整数求 1 的个数。
- 直接暴力计算是 $O(w)$ 的，不能体现优势。
- 查表法：预处理出 $[0, 2^{16})$ 内每个数的 1 的个数 cnt_i，$cnt_i = cnt_{i>>1} + (i\&1)$。
- x 里 1 的个数 $= cnt_{x>>16} + cnt_{x\&65535}$。

求 1 的个数

- 在 bitset 中，我们常常需要对一个 32 位无符号整数求 1 的个数。
- 直接暴力计算是 $O(w)$ 的，不能体现优势。
- 查表法：预处理出 $[0, 2^{16})$ 内每个数的 1 的个数 cnt_i，$cnt_i = cnt_{i>>1} + (i\&1)$。
- x 里 1 的个数 $= cnt_{x>>16} + cnt_{x\&65535}$。
- 内建函数：__builtin_popcount(unsigned int x)

求 1 的个数

- 在 bitset 中，我们常常需要对一个 32 位无符号整数求 1 的个数。
- 直接暴力计算是 $O(w)$ 的，不能体现优势。
- 查表法：预处理出 $[0, 2^{16})$ 内每个数的 1 的个数 cnt_i，$cnt_i = cnt_{i>>1} + (i\&1)$。
- x 里 1 的个数 $= cnt_{x>>16} + cnt_{x\&65535}$。
- 内建函数：__builtin_popcount(unsigned int x)
- 内建函数 ll：__builtin_popcountll(unsigned long long x)

01 背包

给定 n 个物品，每个物品体积为 v_i，m 次询问是否存在一种选法满足体积之和为 k。

01 背包

给定 n 个物品，每个物品体积为 v_i，m 次询问是否存在一种选法满足体积之和为 k。

- $1 \leq n \leq 50000$。

01 背包

给定 n 个物品，每个物品体积为 v_i，m 次询问是否存在一种选法满足体积之和为 k。

- $1 \le n \le 50000$。
- $0 \le m \le 50000$。

01 背包

给定 n 个物品，每个物品体积为 v_i，m 次询问是否存在一种选法满足体积之和为 k。

- $1 \leq n \leq 50000$。
- $0 \leq m \leq 50000$。
- $1 \leq k \leq 50000$。

Solution

- 设 f_i 表示是否存在一个子集满足和为 i。

Solution

- 设 f_i 表示是否存在一个子集满足和为 i。
- 对于当前考虑的物品 v，有 $newf_i = f_i | f_{i-v}$。

Solution

- 设 f_i 表示是否存在一个子集满足和为 i。
- 对于当前考虑的物品 v，有 $newf_i = f_i | f_{i-v}$。
- 若使用 bitset 表示 f，则有 $f| = f << v$。

Solution

- 设 f_i 表示是否存在一个子集满足和为 i。
- 对于当前考虑的物品 v，有 $newf_i = f_i | f_{i-v}$。
- 若使用 bitset 表示 f，则有 $f| = f << v$。
- 时间复杂度 $O(\frac{nk}{w} + m)$。

由乃的玉米田

给定一个非负整数序列 $a_1, a_2, \ldots, a_n$ ， m 次询问：

由乃的玉米田

给定一个非负整数序列 $a_1, a_2, \ldots, a_n$，m 次询问：

(1) 给定 x，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j，满足 $a_i - a_j = x$。

由乃的玉米田

给定一个非负整数序列 $a_1, a_2, \ldots, a_n$ ，m 次询问：

(1) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i - a_j = x$。

(2) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i + a_j = x$。

由乃的玉米田

给定一个非负整数序列 $a_1, a_2, \ldots, a_n$ ，m 次询问：

(1) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i - a_j = x$。

(2) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i + a_j = x$。

(3) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i \times a_j = x$。

由乃的玉米田

给定一个非负整数序列 $a_1, a_2, \ldots, a_n$ ，m 次询问：

(1) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i - a_j = x$。

(2) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i + a_j = x$。

(3) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i \times a_j = x$。

- $1 \leq n, m \leq 100000$。

由乃的玉米田

给定一个非负整数序列 $a_1, a_2, \ldots, a_n$，m 次询问：

(1) 给定 x，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j，满足 $a_i - a_j = x$。

(2) 给定 x，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j，满足 $a_i + a_j = x$。

(3) 给定 x，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j，满足 $a_i \times a_j = x$。

- $1 \leq n, m \leq 100000$。
- $0 \leq a_i, x \leq 100000$。

由乃的玉米田

给定一个非负整数序列 $a_1, a_2, \dots, a_n$ ，m 次询问：

(1) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i - a_j = x$。

(2) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i + a_j = x$。

(3) 给定 x ，询问区间 $[l, r]$ 内能否选出两个可重叠位置 i, j ，满足 $a_i \times a_j = x$。

- $1 \le n, m \le 100000$。
- $0 \le a_i, x \le 100000$。
- Source：BZOJ 4810

Solution

- 首先通过莫队算法，我们可以很轻松地维护 f_i 表示数字 i 是否在区间 $[l, r]$ 出现过。

Solution

- 首先通过莫队算法，我们可以很轻松地维护 f_i 表示数字 i 是否在区间 $[l, r]$ 出现过。
- 对于 $a_i \times a_j = x$ 的询问，枚举 x 的约数判断即可。

Solution

- 首先通过莫队算法，我们可以很轻松地维护 f_i 表示数字 i 是否在区间 $[l, r]$ 出现过。
- 对于 $a_i \times a_j = x$ 的询问，枚举 x 的约数判断即可。
- 对于 $a_i - a_j = x$ 的询问，等价于判断是否存在 i 满足 $f_i \& f_{i+x} = 1$。

Solution

- 首先通过莫队算法，我们可以很轻松地维护 f_i 表示数字 i 是否在区间 $[l, r]$ 出现过。
- 对于 $a_i \times a_j = x$ 的询问，枚举 x 的约数判断即可。
- 对于 $a_i - a_j = x$ 的询问，等价于判断是否存在 i 满足 $f_i \& f_{i+x} = 1$。
- 对于 $a_i + a_j = x$ 的询问，令 $g_{n-i} = f_i$，则等价于判断是否存在 i 满足 $f_i \& g_{n-x+i} = 1$。

Solution

- 首先通过莫队算法，我们可以很轻松地维护 f_i 表示数字 i 是否在区间 $[l, r]$ 出现过。
- 对于 $a_i \times a_j = x$ 的询问，枚举 x 的约数判断即可。
- 对于 $a_i - a_j = x$ 的询问，等价于判断是否存在 i 满足 $f_i \& f_{i+x} = 1$。
- 对于 $a_i + a_j = x$ 的询问，令 $g_{n-i} = f_i$，则等价于判断是否存在 i 满足 $f_i \& g_{n-x+i} = 1$。
- 用 bitset 表示 f, g 即可压位判断。

Solution

- 首先通过莫队算法，我们可以很轻松地维护 f_i 表示数字 i 是否在区间 $[l, r]$ 出现过。
- 对于 $a_i \times a_j = x$ 的询问，枚举 x 的约数判断即可。
- 对于 $a_i - a_j = x$ 的询问，等价于判断是否存在 i 满足 $f_i \& f_{i+x} = 1$。
- 对于 $a_i + a_j = x$ 的询问，令 $g_{n-i} = f_i$，则等价于判断是否存在 i 满足 $f_i \& g_{n-x+i} = 1$。
- 用 bitset 表示 f, g 即可压位判断。
- 时间复杂度 $O(n\sqrt{n} + \frac{nm}{w})$。

掉进兔子洞

给定一个整数序列 $a_1, a_2, \ldots, a_n$ ， m 次询问：

掉进兔子洞

给定一个整数序列 $a_1, a_2, \ldots, a_n$，m 次询问：

每次给定三个区间 $[l_1, r_1], [l_2, r_2], [l_3, r_3]$，设 $cnt_{i,j}$ 表示数字 i 在第 j 个区间的出现次数，求 $\sum \min(cnt_{i,1}, cnt_{i,2}, cnt_{i,3})$。

掉进兔子洞

给定一个整数序列 $a_1, a_2, ..., a_n$ ， m 次询问：

每次给定三个区间 $[l_1, r_1], [l_2, r_2], [l_3, r_3]$ ，设 $cnt_{i,j}$ 表示数字 i 在第 j 个区间的出现次数，求 $\sum \min(cnt_{i,1}, cnt_{i,2}, cnt_{i,3})$。

- $1 \leq n, m \leq 100000$。

掉进兔子洞

给定一个整数序列 $a_1, a_2, ..., a_n$，m 次询问：

每次给定三个区间 $[l_1, r_1], [l_2, r_2], [l_3, r_3]$，设 $cnt_{i,j}$ 表示数字 i 在第 j 个区间的出现次数，求 $\sum \min(cnt_{i,1}, cnt_{i,2}, cnt_{i,3})$。

- $1 \leq n, m \leq 100000$。
- $1 \leq a_i \leq 10^9$。

掉进兔子洞

给定一个整数序列 $a_1, a_2, ..., a_n$ ，m 次询问：

每次给定三个区间 $[l_1, r_1], [l_2, r_2], [l_3, r_3]$ ，设 $cnt_{i,j}$ 表示数字 i 在第 j 个区间的出现次数，求 $\sum \min(cnt_{i,1}, cnt_{i,2}, cnt_{i,3})$。

- $1 \le n, m \le 100000$。
- $1 \le a_i \le 10^9$。
- Source：BZOJ 4939

Solution

- 显然 a_i 可以离散化，使得 $1 \le a_i \le n$。

Solution

- 显然 a_i 可以离散化，使得 $1 \leq a_i \leq n$。
- 若没有重复数字，那么设 $cnt_{i,j} \leq 1$，用 bitset 即可表示。

Solution

- 显然 a_i 可以离散化，使得 $1 \le a_i \le n$。
- 若没有重复数字，那么设 $cnt_{i,j} \le 1$，用 bitset 即可表示。
- 将每个询问拆成 3 个询问，分别用莫队算法求出 cnt 数组，然后求交即可。

Solution

- 显然 a_i 可以离散化，使得 $1 \le a_i \le n$。
- 若没有重复数字，那么设 $cnt_{i,j} \le 1$，用 bitset 即可表示。
- 将每个询问拆成 3 个询问，分别用莫队算法求出 cnt 数组，然后求交即可。
- 时间复杂度 $O(n\sqrt{n} + \frac{nm}{w})$。

Solution

- 显然 a_i 可以离散化，使得 $1 \le a_i \le n$。
- 若没有重复数字，那么设 $cnt_{i,j} \le 1$，用 bitset 即可表示。
- 将每个询问拆成 3 个询问，分别用莫队算法求出 cnt 数组，然后求交即可。
- 时间复杂度 $O(n\sqrt{n} + \frac{nm}{w})$。
- 空间复杂度 $O(\frac{nm}{w})$，存不下？

Solution

- 显然 a_i 可以离散化，使得 $1 \le a_i \le n$。
- 若没有重复数字，那么设 $cnt_{i,j} \le 1$，用 bitset 即可表示。
- 将每个询问拆成 3 个询问，分别用莫队算法求出 cnt 数组，然后求交即可。
- 时间复杂度 $O(n\sqrt{n} + \frac{nm}{w})$。
- 空间复杂度 $O(\frac{nm}{w})$，存不下？
- 将询问分批，每次只做 25000 个即可。

Solution

- a_i 有重复应该如何解决？

Solution

- a_i 有重复应该如何解决？
- 假设原序列是 1 1 1 8 8 9，离散化后会得到 1 1 1 4 4 6。

Solution

- a_i 有重复应该如何解决？
- 假设原序列是 1 1 1 8 8 9，离散化后会得到 1 1 1 4 4 6。
- 假设 x 出现了 y 次，那么显然 $[x+1, x+y-1]$ 都是空位，我们把 $[x, x+y-1]$ 都填上 1 即可。

Solution

- a_i 有重复应该如何解决？
- 假设原序列是 1 1 1 8 8 9，离散化后会得到 1 1 1 4 4 6。
- 假设 x 出现了 y 次，那么显然 $[x+1, x+y-1]$ 都是空位，我们把 $[x, x+y-1]$ 都填上 1 即可。
- 利用这个小技巧，我们依然可以使用 bitset 来保存 cnt。

集合求和

给定一个 n 个点，m 条边的有向图，点 i 的权值为 v_i。

集合求和

给定一个 n 个点，m 条边的有向图，点 i 的权值为 v_i。
对于每个点 i，求从 i 出发能到达的所有点的权值之和。

集合求和

给定一个 n 个点，m 条边的有向图，点 i 的权值为 v_i。对于每个点 i，求从 i 出发能到达的所有点的权值之和。

- $1 \leq n \leq 30000$。

集合求和

给定一个 n 个点，m 条边的有向图，点 i 的权值为 v_i。
对于每个点 i，求从 i 出发能到达的所有点的权值之和。

- $1 \le n \le 30000$。
- $0 \le m \le 30000$。

集合求和

给定一个 n 个点，m 条边的有向图，点 i 的权值为 v_i。
对于每个点 i，求从 i 出发能到达的所有点的权值之和。

- $1 \leq n \leq 30000$。
- $0 \leq m \leq 30000$。
- $0 \leq v_i \leq 10000$。

Solution

- 先求出 SCC，将图缩点为 DAG。

Solution

- 先求出 SCC，将图缩点为 DAG。
- 设 $f_{i,j}$ 表示 i 能否到达 j，则 $f_{i,j} = OR(f_{next,j})$。

Solution

- 先求出 SCC，将图缩点为 DAG。
- 设 $f_{i,j}$ 表示 i 能否到达 j，则 $f_{i,j} = OR(f_{next,j})$。
- 若使用 bitset 则可以在 $O(\frac{nm}{w})$ 的时间内求出 f。

Solution

- 先求出 SCC，将图缩点为 DAG。
- 设 $f_{i,j}$ 表示 i 能否到达 j，则 $f_{i,j} = OR(f_{next,j})$。
- 若使用 bitset 则可以在 $O(\frac{nm}{w})$ 的时间内求出 f。
- 如何根据 f 求出点权之和？

Solution

- 先求出 SCC，将图缩点为 DAG。
- 设 $f_{i,j}$ 表示 i 能否到达 j，则 $f_{i,j} = OR(f_{next,j})$。
- 若使用 bitset 则可以在 $O(\frac{nm}{w})$ 的时间内求出 f。
- 如何根据 f 求出点权之和？
- Method of Four Russians：将 1 到 n 分成 k 块，每块通过 $O(2^k)$ 的预处理求出每一个集合的答案，那么整体的答案等于每块答案之和。

Solution

- 先求出 SCC，将图缩点为 DAG。
- 设 $f_{i,j}$ 表示 i 能否到达 j，则 $f_{i,j} = OR(f_{next,j})$。
- 若使用 bitset 则可以在 $O(\frac{nm}{w})$ 的时间内求出 f。
- 如何根据 f 求出点权之和？
- Method of Four Russians：将 1 到 n 分成 k 块，每块通过 $O(2^k)$ 的预处理求出每一个集合的答案，那么整体的答案等于每块答案之和。
- 取 $k = 13$，则时间复杂度为 $O(\frac{n(n+m+2^{13})}{13})$。

Associated Vertices

给定一个 n 个点，m 条边的有向图。

Associated Vertices

给定一个 n 个点，m 条边的有向图。

对于两个不同的点 a, b，若存在一个点 c 满足 c 可以同时到达 a 和 b，则称 a, b 是关联的。

Associated Vertices

给定一个 n 个点，m 条边的有向图。

对于两个不同的点 a, b，若存在一个点 c 满足 c 可以同时到达 a 和 b，则称 a, b 是关联的。

请统计有多少对 a, b 是关联的。

Associated Vertices

给定一个 n 个点，m 条边的有向图。

对于两个不同的点 a, b，若存在一个点 c 满足 c 可以同时到达 a 和 b，则称 a, b 是关联的。

请统计有多少对 a, b 是关联的。

- $1 \leq n \leq 50000$。

Associated Vertices

给定一个 n 个点，m 条边的有向图。

对于两个不同的点 a, b，若存在一个点 c 满足 c 可以同时到达 a 和 b，则称 a, b 是关联的。

请统计有多少对 a, b 是关联的。

- $1 \leq n \leq 50000$。
- $0 \leq m \leq 50000$。

Associated Vertices

给定一个 n 个点，m 条边的有向图。

对于两个不同的点 a, b，若存在一个点 c 满足 c 可以同时到达 a 和 b，则称 a, b 是关联的。

请统计有多少对 a, b 是关联的。

- $1 \le n \le 50000$。
- $0 \le m \le 50000$。
- Source：XVI Open Cup named after E.V. Pankratiev. GP of Ukraine

Solution

- 先求出 SCC，将图缩点为 DAG。

Solution

- 先求出 SCC，将图缩点为 DAG。
- 设 f_i 为 i 能到达的点集，则 $f_i = OR(f_{next}, i)$。

Solution

- 先求出 SCC，将图缩点为 DAG。
- 设 f_i 为 i 能到达的点集，则 $f_i = OR(f_{next}, i)$。
- 设 g_i 表示和 i 关联的点集，则 $g_i = OR(g_{pre}, f_i)$。

Solution

- 先求出 SCC，将图缩点为 DAG。
- 设 f_i 为 i 能到达的点集，则 $f_i = OR(f_{next}, i)$。
- 设 g_i 表示和 i 关联的点集，则 $g_i = OR(g_{pre}, f_i)$。
- 时间复杂度 $O(\frac{nm}{w})$。

Graph Optimization

有一张 n 个点的有向图和 m 条信息，信息有 2 种形式：

Graph Optimization

有一张 n 个点的有向图和 m 条信息，信息有 2 种形式：

(1)a 点能到达 b 点。

Graph Optimization

有一张 n 个点的有向图和 m 条信息，信息有 2 种形式：

(1)a 点能到达 b 点。

(2)a 点不能到达 b 点。

Graph Optimization

有一张 n 个点的有向图和 m 条信息，信息有 2 种形式：

(1)a 点能到达 b 点。

(2)a 点不能到达 b 点。

请构造一张符合条件的图，或判断无解，要求边数不能超过 300000。

Graph Optimization

有一张 n 个点的有向图和 m 条信息，信息有 2 种形式：

(1)a 点能到达 b 点。

(2)a 点不能到达 b 点。

请构造一张符合条件的图，或判断无解，要求边数不能超过 300000。

- $1 \leq n \leq 100000$。

Graph Optimization

有一张 n 个点的有向图和 m 条信息，信息有 2 种形式：

(1)a 点能到达 b 点。

(2)a 点不能到达 b 点。

请构造一张符合条件的图，或判断无解，要求边数不能超过 300000。

- $1 \le n \le 100000$。
- $0 \le m \le 100000$。

Graph Optimization

有一张 n 个点的有向图和 m 条信息，信息有 2 种形式：

(1)a 点能到达 b 点。

(2)a 点不能到达 b 点。

请构造一张符合条件的图，或判断无解，要求边数不能超过 300000。

- $1 \le n \le 100000$。
- $0 \le m \le 100000$。
- Source：XVI Open Cup named after E.V. Pankratiev. GP of Siberia

Solution

- 对于 (1)，直接加入 $a \to b$ 的边可以最小化对其它点对的影响。

Solution

- 对于 (1)，直接加入 $a \to b$ 的边可以最小化对其它点对的影响。
- 那么只要检查是否有 (2) 不满足即可。

Solution

- 对于 (1)，直接加入 $a \to b$ 的边可以最小化对其它点对的影响。
- 那么只要检查是否有 (2) 不满足即可。
- 缩点之后 bitset 统计即可。

Solution

- 对于 (1)，直接加入 $a \to b$ 的边可以最小化对其它点对的影响。
- 那么只要检查是否有 (2) 不满足即可。
- 缩点之后 bitset 统计即可。
- 时间复杂度 $O(\frac{nm}{w})$。

Solution

- 对于 (1)，直接加入 $a \to b$ 的边可以最小化对其它点对的影响。
- 那么只要检查是否有 (2) 不满足即可。
- 缩点之后 bitset 统计即可。
- 时间复杂度 $O(\frac{nm}{w})$。
- 空间复杂度 $O(\frac{n^2}{w})$，开不下？

Solution

- 对于 (1)，直接加入 $a \to b$ 的边可以最小化对其它点对的影响。
- 那么只要检查是否有 (2) 不满足即可。
- 缩点之后 bitset 统计即可。
- 时间复杂度 $O(\frac{nm}{w})$。
- 空间复杂度 $O(\frac{n^2}{w})$，开不下？
- 将 1 到 n 分成 3 组，每组 33334 个然后计算即可。

Solution

- 对于 (1)，直接加入 $a \to b$ 的边可以最小化对其它点对的影响。
- 那么只要检查是否有 (2) 不满足即可。
- 缩点之后 bitset 统计即可。
- 时间复杂度 $O(\frac{nm}{w})$。
- 空间复杂度 $O(\frac{n^2}{w})$，开不下？
- 将 1 到 n 分成 3 组，每组 33334 个然后计算即可。
- 要尽量减少组数，因为赋值操作很慢。

Dual Sim Phone

给定一个 n 个点，m 条边的无向图，每条边有一个代价 c_i。

Dual Sim Phone

给定一个 n 个点，m 条边的无向图，每条边有一个代价 c_i。

请选择两个点，并从它们的边表中选择一些边，使得这些边覆盖了所有 n 个点。

Dual Sim Phone

给定一个 n 个点，m 条边的无向图，每条边有一个代价 c_i。

请选择两个点，并从它们的边表中选择一些边，使得这些边覆盖了所有 n 个点。

求选择的边的代价最大值的最小值。

Dual Sim Phone

给定一个 n 个点，m 条边的无向图，每条边有一个代价 c_i。

请选择两个点，并从它们的边表中选择一些边，使得这些边覆盖了所有 n 个点。

求选择的边的代价最大值的最小值。

- $2 \leq n \leq 10000$。

Dual Sim Phone

给定一个 n 个点，m 条边的无向图，每条边有一个代价 c_i。

请选择两个点，并从它们的边表中选择一些边，使得这些边覆盖了所有 n 个点。

求选择的边的代价最大值的最小值。

- $2 \leq n \leq 10000$。
- $0 \leq m \leq 100000$。

Dual Sim Phone

给定一个 n 个点，m 条边的无向图，每条边有一个代价 c_i。

请选择两个点，并从它们的边表中选择一些边，使得这些边覆盖了所有 n 个点。

求选择的边的代价最大值的最小值。

- $2 \le n \le 10000$。
- $0 \le m \le 100000$。
- $0 \le c_i \le 10^9$。

Dual Sim Phone

给定一个 n 个点，m 条边的无向图，每条边有一个代价 c_i。

请选择两个点，并从它们的边表中选择一些边，使得这些边覆盖了所有 n 个点。

求选择的边的代价最大值的最小值。

- $2 \le n \le 10000$。
- $0 \le m \le 100000$。
- $0 \le c_i \le 10^9$。
- Source： Ural 1811

Solution

- 二分答案，转化为判定是否存在两个点的边表并集为全集。

Solution

- 二分答案，转化为判定是否存在两个点的边表并集为全集。
- 这两个点必然满足 $deg_x + deg_y \geq n$。

Solution

- 二分答案，转化为判定是否存在两个点的边表并集为全集。
- 这两个点必然满足 $deg_x + deg_y \geq n$。
- 不妨设 $deg_x \geq deg_y$，那么有 $deg_x \times 2 \geq n$。

Solution

- 二分答案，转化为判定是否存在两个点的边表并集为全集。
- 这两个点必然满足 $deg_x + deg_y \geq n$。
- 不妨设 $deg_x \geq deg_y$，那么有 $deg_x \times 2 \geq n$。
- 考虑枚举 x，最多只会有 $O(\frac{m}{n})$ 个 x。

Solution

- 二分答案，转化为判定是否存在两个点的边表并集为全集。
- 这两个点必然满足 $deg_x + deg_y \geq n$。
- 不妨设 $deg_x \geq deg_y$，那么有 $deg_x \times 2 \geq n$。
- 考虑枚举 x，最多只会有 $O(\frac{m}{n})$ 个 x。
- 设 $f_{i,j}$ 表示 i 是否没有指向 j，那么只要存在 $f_{x,j} \& f_{y,j} = 1$ 即不可行。

Solution

- 二分答案，转化为判定是否存在两个点的边表并集为全集。
- 这两个点必然满足 $deg_x + deg_y \geq n$。
- 不妨设 $deg_x \geq deg_y$，那么有 $deg_x \times 2 \geq n$。
- 考虑枚举 x，最多只会有 $O(\frac{m}{n})$ 个 x。
- 设 $f_{i,j}$ 表示 i 是否没有指向 j，那么只要存在 $f_{x,j} \& f_{y,j} = 1$ 即不可行。
- 可以压位计算，时间复杂度 $O(\frac{nm}{w})$。

Solution

- 二分答案，转化为判定是否存在两个点的边表并集为全集。
- 这两个点必然满足 $deg_x + deg_y \geq n$。
- 不妨设 $deg_x \geq deg_y$，那么有 $deg_x \times 2 \geq n$。
- 考虑枚举 x，最多只会有 $O(\frac{m}{n})$ 个 x。
- 设 $f_{i,j}$ 表示 i 是否没有指向 j，那么只要存在 $f_{x,j} \& f_{y,j} = 1$ 即不可行。
- 可以压位计算，时间复杂度 $O(\frac{nm}{w})$。
- 当 n 比较大时该算法并不能在规定时限内出解。

Solution

- 算法 2：枚举 y 的所有出边，通过时间戳判定是否出现在 x 中。

Solution

- 算法 2：枚举 y 的所有出边，通过时间戳判定是否出现在 x 中。
- 时间复杂度 $O(\frac{m^2}{n})$。

Solution

- 算法 2：枚举 y 的所有出边，通过时间戳判定是否出现在 x 中。
- 时间复杂度 $O(\frac{m^2}{n})$。
- 显然重边只需保留权值最小的，去重后有 $n \geq \sqrt{m}$。

Solution

- 算法 2：枚举 y 的所有出边，通过时间戳判定是否出现在 x 中。
- 时间复杂度 $O(\frac{m^2}{n})$。
- 显然重边只需保留权值最小的，去重后有 $n \geq \sqrt{m}$。
- 则该算法时间复杂度为 $O(m\sqrt{m})$，在 m 较大时也不能在规定时限内出解。

Solution

- 注意到算法 1 在 n 较小时比较快，而算法 2 在 n 比较大时比较快，故考虑平衡复杂度。

Solution

- 注意到算法 1 在 n 较小时比较快，而算法 2 在 n 比较大时比较快，故考虑平衡复杂度。
- 设 S 为阈值，当 $n \le S$ 时用算法 1，否则用算法 2，则有 $\frac{Sm}{w} \le \frac{m^2}{S}$。

Solution

- 注意到算法 1 在 n 较小时比较快，而算法 2 在 n 比较大时比较快，故考虑平衡复杂度。
- 设 S 为阈值，当 $n \le S$ 时用算法 1，否则用算法 2，则有 $\frac{Sm}{w} \le \frac{m^2}{S}$。
- 当 S 取 $\sqrt{wm}$ 时，取得最优复杂度 $O(m\sqrt{\frac{m}{w}})$。

Solution

- 注意到算法 1 在 n 较小时比较快，而算法 2 在 n 比较大时比较快，故考虑平衡复杂度。
- 设 S 为阈值，当 $n \leq S$ 时用算法 1，否则用算法 2，则有 $\frac{Sm}{w} \leq \frac{m^2}{S}$。
- 当 S 取 $\sqrt{wm}$ 时，取得最优复杂度 $O(m\sqrt{\frac{m}{w}})$。
- 总时间复杂度 $O(m \log m\sqrt{\frac{m}{w}})$。

All Pair Shortest Path

给定一张 n 个点的有向图，边权都是 1。

All Pair Shortest Path

给定一张 n 个点的有向图，边权都是 1。

设 $dis(i,j)$ 为 i 到 j 的最短路，若不存在则为 n。请计算：

All Pair Shortest Path

给定一张 n 个点的有向图，边权都是 1。

设 $dis(i,j)$ 为 i 到 j 的最短路，若不存在则为 n。请计算：

$$\sum_{i=1}^{n}\sum_{j=1}^{n} dis(i,j)^2$$

All Pair Shortest Path

给定一张 n 个点的有向图，边权都是 1。
设 $dis(i,j)$ 为 i 到 j 的最短路，若不存在则为 n。请计算：

$$\sum_{i=1}^{n}\sum_{j=1}^{n} dis(i,j)^2$$

- $1 \le n \le 2000$。

All Pair Shortest Path

给定一张 n 个点的有向图，边权都是 1。

设 $dis(i,j)$ 为 i 到 j 的最短路，若不存在则为 n。请计算：

$$\sum_{i=1}^{n}\sum_{j=1}^{n} dis(i,j)^2$$

- $1 \le n \le 2000$。
- Source：ftiasch's Contest #4

Solution

- 直接枚举起点进行 BFS 的复杂度是 $O(n^3)$ 的，不能承受。考虑优化每次 BFS 的效率。

Solution

- 直接枚举起点进行 BFS 的复杂度是 $O(n^3)$ 的，不能承受。考虑优化每次 BFS 的效率。
- 每次取出队头的时候，要将所有与它相连且未访问过的点加入队尾，访问过的点如果再遍历就会有浪费。

Solution

- 直接枚举起点进行 BFS 的复杂度是 $O(n^3)$ 的，不能承受。考虑优化每次 BFS 的效率。
- 每次取出队头的时候，要将所有与它相连且未访问过的点加入队尾，访问过的点如果再遍历就会有浪费。
- 所以对于每个点 i，维护与它有边的点的集合 g_i，以及未访问过的点的集合 vis。

Solution

- 直接枚举起点进行 BFS 的复杂度是 $O(n^3)$ 的，不能承受。考虑优化每次 BFS 的效率。
- 每次取出队头的时候，要将所有与它相连且未访问过的点加入队尾，访问过的点如果再遍历就会有浪费。
- 所以对于每个点 i，维护与它有边的点的集合 g_i，以及未访问过的点的集合 vis。
- 那么只需要遍历 $g_i \& vis$ 集合中的所有点即可。

Solution

- 直接枚举起点进行 BFS 的复杂度是 $O(n^3)$ 的，不能承受。考虑优化每次 BFS 的效率。
- 每次取出队头的时候，要将所有与它相连且未访问过的点加入队尾，访问过的点如果再遍历就会有浪费。
- 所以对于每个点 i，维护与它有边的点的集合 g_i，以及未访问过的点的集合 vis。
- 那么只需要遍历 $g_i \& vis$ 集合中的所有点即可。
- bitset 维护。

Solution

- 直接枚举起点进行 BFS 的复杂度是 $O(n^3)$ 的，不能承受。考虑优化每次 BFS 的效率。
- 每次取出队头的时候，要将所有与它相连且未访问过的点加入队尾，访问过的点如果再遍历就会有浪费。
- 所以对于每个点 i，维护与它有边的点的集合 g_i，以及未访问过的点的集合 vis。
- 那么只需要遍历 $g_i \& vis$ 集合中的所有点即可。
- bitset 维护。
- 时间复杂度 $O(\frac{n^3}{w})$。

友好城市

给定一张 n 个点，m 条边的有向图。

友好城市

给定一张 n 个点，m 条边的有向图。

q 次询问，每次给定一个区间 $[l, r]$，问仅保留编号在这个区间内的边时，能互相到达的点对数。

友好城市

给定一张 n 个点，m 条边的有向图。

q 次询问，每次给定一个区间 $[l, r]$，问仅保留编号在这个区间内的边时，能互相到达的点对数。

强制在线。

友好城市

给定一张 n 个点，m 条边的有向图。

q 次询问，每次给定一个区间 $[l, r]$，问仅保留编号在这个区间内的边时，能互相到达的点对数。

强制在线。

- $2 \leq n \leq 150$。

友好城市

给定一张 n 个点，m 条边的有向图。

q 次询问，每次给定一个区间 $[l, r]$，问仅保留编号在这个区间内的边时，能互相到达的点对数。

强制在线。

- $2 \le n \le 150$。
- $1 \le m \le 300000$。

友好城市

给定一张 n 个点，m 条边的有向图。

q 次询问，每次给定一个区间 $[l, r]$，问仅保留编号在这个区间内的边时，能互相到达的点对数。

强制在线。

- $2 \leq n \leq 150$。
- $1 \leq m \leq 300000$。
- $1 \leq q \leq 50000$。

友好城市

给定一张 n 个点，m 条边的有向图。

q 次询问，每次给定一个区间 $[l, r]$，问仅保留编号在这个区间内的边时，能互相到达的点对数。

强制在线。

- $2 \leq n \leq 150$。
- $1 \leq m \leq 300000$。
- $1 \leq q \leq 50000$。
- Source： BZOJ 2017 省选十连测第二场

Solution

- 对于每个询问，求出强连通分量，那么一个点数为 t 的强连通分量对答案的贡献为 $\frac{t(t-1)}{2}$。

Solution

- 对于每个询问，求出强连通分量，那么一个点数为 t 的强连通分量对答案的贡献为 $\frac{t(t-1)}{2}$。
- 除了 Tarjan 算法，求强连通分量还可以用 Kosaraju 算法，只需要在正反图各做一次 DFS 即可。

Solution

- 对于每个询问，求出强连通分量，那么一个点数为 t 的强连通分量对答案的贡献为 $\frac{t(t-1)}{2}$。
- 除了 Tarjan 算法，求强连通分量还可以用 Kosaraju 算法，只需要在正反图各做一次 DFS 即可。
- 面对稠密图，Kosaraju 算法的瓶颈在于寻找与点 x 相连且未访问过的点。

Solution

- 对于每个询问，求出强连通分量，那么一个点数为 t 的强连通分量对答案的贡献为 $\frac{t(t-1)}{2}$。
- 除了 Tarjan 算法，求强连通分量还可以用 Kosaraju 算法，只需要在正反图各做一次 DFS 即可。
- 面对稠密图，Kosaraju 算法的瓶颈在于寻找与点 x 相连且未访问过的点。
- 考虑用 bitset 来保存边表 g_x，以及未访问过的点集 S，那么取出 $g_x\&S$ 内的所有 1 即可。

Solution

- 对于每个询问，求出强连通分量，那么一个点数为 t 的强连通分量对答案的贡献为 $\frac{t(t-1)}{2}$。
- 除了 Tarjan 算法，求强连通分量还可以用 Kosaraju 算法，只需要在正反图各做一次 DFS 即可。
- 面对稠密图，Kosaraju 算法的瓶颈在于寻找与点 x 相连且未访问过的点。
- 考虑用 bitset 来保存边表 g_x，以及未访问过的点集 S，那么取出 $g_x \& S$ 内的所有 1 即可。
- 这样一来，求强连通分量的复杂度降低为 $O(\frac{n^2}{w})$。

Solution

- 还有一个问题：如何取出所有在 $[l, r]$ 内的边？

Solution

- 还有一个问题：如何取出所有在 $[l, r]$ 内的边？
- 考虑将边分成 $\sqrt{m}$ 块，用 bitset 保存每块的边表，然后用 ST 表维护横跨任意两块的边表。

Solution

- 还有一个问题：如何取出所有在 $[l, r]$ 内的边？
- 考虑将边分成 $\sqrt{m}$ 块，用 bitset 保存每块的边表，然后用 ST 表维护横跨任意两块的边表。
- 那么对于每个询问，只需要在 ST 表中查询一次，然后往左往右暴力添加 $O(\sqrt{m})$ 条边即可。

Solution

- 还有一个问题：如何取出所有在 $[l, r]$ 内的边？
- 考虑将边分成 $\sqrt{m}$ 块，用 bitset 保存每块的边表，然后用 ST 表维护横跨任意两块的边表。
- 那么对于每个询问，只需要在 ST 表中查询一次，然后往左往右暴力添加 $O(\sqrt{m})$ 条边即可。
- 时间复杂度 $O\left(\frac{\left(\sqrt{m}\log m+q\right)n^2}{w} + q\sqrt{m}\right)$。

AABB

给定平面上 n 个平行坐标轴的矩形，编号为 1 到 n。

AABB

给定平面上 n 个平行坐标轴的矩形，编号为 1 到 n。

q 次询问，每次给定 $[l_1, r_1], [l_2, r_2]$，求在编号为 l_1 到 r_1 中选一个矩形 i，l_2 到 r_2 中选一个矩形 j，矩形 i 和矩形 j 有公共点的方案数。

AABB

给定平面上 n 个平行坐标轴的矩形，编号为 1 到 n。

q 次询问，每次给定 $[l_1, r_1], [l_2, r_2]$，求在编号为 l_1 到 r_1 中选一个矩形 i，l_2 到 r_2 中选一个矩形 j，矩形 i 和矩形 j 有公共点的方案数。

强制在线。

AABB

给定平面上 n 个平行坐标轴的矩形，编号为 1 到 n。

q 次询问，每次给定 $[l_1, r_1], [l_2, r_2]$，求在编号为 l_1 到 r_1 中选一个矩形 i，l_2 到 r_2 中选一个矩形 j，矩形 i 和矩形 j 有公共点的方案数。

强制在线。

- $1 \leq n \leq 30000$。

AABB

给定平面上 n 个平行坐标轴的矩形，编号为 1 到 n。

q 次询问，每次给定 $[l_1, r_1], [l_2, r_2]$，求在编号为 l_1 到 r_1 中选一个矩形 i，l_2 到 r_2 中选一个矩形 j，矩形 i 和矩形 j 有公共点的方案数。

强制在线。

- $1 \leq n \leq 30000$。
- $1 \leq q \leq 30000$。

AABB

给定平面上 n 个平行坐标轴的矩形，编号为 1 到 n。

q 次询问，每次给定 $[l_1, r_1], [l_2, r_2]$，求在编号为 l_1 到 r_1 中选一个矩形 i，l_2 到 r_2 中选一个矩形 j，矩形 i 和矩形 j 有公共点的方案数。

强制在线。

- $1 \le n \le 30000$。
- $1 \le q \le 30000$。
- $1 \le x_i, y_i \le n$。

AABB

给定平面上 n 个平行坐标轴的矩形，编号为 1 到 n。

q 次询问，每次给定 $[l_1, r_1], [l_2, r_2]$，求在编号为 l_1 到 r_1 中选一个矩形 i，l_2 到 r_2 中选一个矩形 j，矩形 i 和矩形 j 有公共点的方案数。

强制在线。

- $1 \leq n \leq 30000$。
- $1 \leq q \leq 30000$。
- $1 \leq x_i, y_i \leq n$。
- Source：BZOJ 3567

Solution

- 考虑以块大小为 32 将序列分块，设 $s_{i,j}$ 表示前 i 块和前 j 块矩形相交的对数，$f_{i,j}$ 表示矩形 i 和前 j 块的相交个数。

Solution

- 考虑以块大小为 32 将序列分块，设 $s_{i,j}$ 表示前 i 块和前 j 块矩形相交的对数，$f_{i,j}$ 表示矩形 i 和前 j 块的相交个数。
- 如果矩形 i 和 j 相交，那么有：

$x_{1j} < x_{2i}$

$x_{2j} > x_{1i}$

$y_{1j} < y_{2i}$

$y_{2j} > y_{1i}$

Solution

- 考虑以块大小为 32 将序列分块，设 $s_{i,j}$ 表示前 i 块和前 j 块矩形相交的对数，$f_{i,j}$ 表示矩形 i 和前 j 块的相交个数。
- 如果矩形 i 和 j 相交，那么有：

  $x_{1j} < x_{2i}$

  $x_{2j} > x_{1i}$

  $y_{1j} < y_{2i}$

  $y_{2j} > y_{1i}$
- 将这 4 维分开处理，对于每一维按相应参数排序，维护一个集合，支持加入以及求交。

Solution

- 考虑以块大小为 32 将序列分块，设 $s_{i,j}$ 表示前 i 块和前 j 块矩形相交的对数，$f_{i,j}$ 表示矩形 i 和前 j 块的相交个数。
- 如果矩形 i 和 j 相交，那么有：

  $x_{1j} < x_{2i}$

  $x_{2j} > x_{1i}$

  $y_{1j} < y_{2i}$

  $y_{2j} > y_{1i}$
- 将这 4 维分开处理，对于每一维按相应参数排序，维护一个集合，支持加入以及求交。
- 这显然可以通过 bitset 在 $O(\frac{n^2}{32})$ 的时间内完成。

Solution

- 通过 bitset 求出某个矩形 i 和所有矩形的相交情况后，即可在 $O(\frac{n}{32})$ 的时间内更新 s 和 f。

Solution

- 通过 bitset 求出某个矩形 i 和所有矩形的相交情况后，即可在 $O(\frac{n}{32})$ 的时间内更新 s 和 f。
- 对 s 和 f 求前缀和即可完成 s 和 f 的预处理。

Solution

- 通过 bitset 求出某个矩形 i 和所有矩形的相交情况后，即可在 $O(\frac{n}{32})$ 的时间内更新 s 和 f。
- 对 s 和 f 求前缀和即可完成 s 和 f 的预处理。
- 对于查询，首先整块的部分可以通过 s 查询，然后暴力往右往上扩展，用 f 做到 $O(1)$ 询问。对于块内零散部分，暴力检验即可。

Solution

- 通过 bitset 求出某个矩形 i 和所有矩形的相交情况后，即可在 $O(\frac{n}{32})$ 的时间内更新 s 和 f。
- 对 s 和 f 求前缀和即可完成 s 和 f 的预处理。
- 对于查询，首先整块的部分可以通过 s 查询，然后暴力往右往上扩展，用 f 做到 $O(1)$ 询问。对于块内零散部分，暴力检验即可。
- 每次查询的复杂度为 $O(32^2)$。

Solution

- 通过 bitset 求出某个矩形 i 和所有矩形的相交情况后，即可在 $O(\frac{n}{32})$ 的时间内更新 s 和 f。
- 对 s 和 f 求前缀和即可完成 s 和 f 的预处理。
- 对于查询，首先整块的部分可以通过 s 查询，然后暴力往右往上扩展，用 f 做到 $O(1)$ 询问。对于块内零散部分，暴力检验即可。
- 每次查询的复杂度为 $O(32^2)$。
- 时间复杂度 $O(\frac{n^2}{32} + q \times 32^2)$。

Solution

- 内存限制只有 64MB，n 个 bitset 开不下？

Solution

- 内存限制只有 64MB，n 个 bitset 开不下？
- 注意到 bitset 加入元素是 $O(1)$ 的，所以考虑再次分块。

Solution

- 内存限制只有 64MB，n 个 bitset 开不下？
- 注意到 bitset 加入元素是 $O(1)$ 的，所以考虑再次分块。
- 每加入 128 个元素后备份一次，即可满足内存限制。

带可选字符的多字符串匹配

给定一个长度为 m 的字符串 T，请找出其中所有的符合特定模式的子串位置。

带可选字符的多字符串匹配

给定一个长度为 m 的字符串 T，请找出其中所有的符合特定模式的子串位置。

符合特定模式是指，该子串的长度为 n，并且第 i 个字符需要在给定的字符集合 S_i 中。

带可选字符的多字符串匹配

给定一个长度为 m 的字符串 T，请找出其中所有的符合特定模式的子串位置。

符合特定模式是指，该子串的长度为 n，并且第 i 个字符需要在给定的字符集合 S_i 中。

- $1 \le n \le 500, n \le m$。

带可选字符的多字符串匹配

给定一个长度为 m 的字符串 T，请找出其中所有的符合特定模式的子串位置。

符合特定模式是指，该子串的长度为 n，并且第 i 个字符需要在给定的字符集合 S_i 中。

- $1 \leq n \leq 500, n \leq m$。
- $1 \leq m \leq 5000000$。

带可选字符的多字符串匹配

给定一个长度为 m 的字符串 T，请找出其中所有的符合特定模式的子串位置。

符合特定模式是指，该子串的长度为 n，并且第 i 个字符需要在给定的字符集合 S_i 中。

- $1 \leq n \leq 500, n \leq m$。
- $1 \leq m \leq 5000000$。
- 字符集大小：62。

带可选字符的多字符串匹配

给定一个长度为 m 的字符串 T，请找出其中所有的符合特定模式的子串位置。

符合特定模式是指，该子串的长度为 n，并且第 i 个字符需要在给定的字符集合 S_i 中。

- $1 \leq n \leq 500, n \leq m$。
- $1 \leq m \leq 5000000$。
- 字符集大小：62。
- Source： HDU 5716

Solution

- shift-and 算法。

Solution

- shift-and 算法。
- 设 $v_{i,j}$ 表示文本串长度为 i 的前缀能否匹配模式串长度为 j 的前缀，$f_{i,j}$ 表示字符 i 能否匹配模式串的第 j 个位置。

Solution

- shift-and 算法。
- 设 $v_{i,j}$ 表示文本串长度为 i 的前缀能否匹配模式串长度为 j 的前缀，$f_{i,j}$ 表示字符 i 能否匹配模式串的第 j 个位置。
- $v_{i+1,j+1} = v_{i,j} \& f_{T_{i+1},j+1}$

Solution

- shift-and 算法。
- 设 $v_{i,j}$ 表示文本串长度为 i 的前缀能否匹配模式串长度为 j 的前缀，$f_{i,j}$ 表示字符 i 能否匹配模式串的第 j 个位置。
- $v_{i+1,j+1} = v_{i,j} \& f_{T_{i+1},j+1}$
- 显然 j 这一维可以用 bitset 加速。

Solution

- shift-and 算法。
- 设 $v_{i,j}$ 表示文本串长度为 i 的前缀能否匹配模式串长度为 j 的前缀，$f_{i,j}$ 表示字符 i 能否匹配模式串的第 j 个位置。
- $v_{i+1,j+1} = v_{i,j} \& f_{T_{i+1},j+1}$
- 显然 j 这一维可以用 bitset 加速。
- 时间复杂度 $O(\frac{nm}{w})$。

迷失的字符串

给定一棵 n 个点的无根树，每条边上有个小写字符。

迷失的字符串

给定一棵 n 个点的无根树，每条边上有个小写字符。

m 次询问，每次给定一个字符串 S，问是否存在一条路径满足沿路边上的字符拼起来恰好是 S。

迷失的字符串

给定一棵 n 个点的无根树，每条边上有个小写字符。

m 次询问，每次给定一个字符串 S，问是否存在一条路径满足沿路边上的字符拼起来恰好是 S。

- $2 \leq n \leq 30000$。

迷失的字符串

给定一棵 n 个点的无根树，每条边上有个小写字符。

m 次询问，每次给定一个字符串 S，问是否存在一条路径满足沿路边上的字符拼起来恰好是 S。

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

迷失的字符串

给定一棵 n 个点的无根树，每条边上有个小写字符。

m 次询问，每次给定一个字符串 S，问是否存在一条路径满足沿路边上的字符拼起来恰好是 S。

- $2 \leq n \leq 30000$。
- $1 \leq m \leq 30000$。
- $\sum |S| \leq 30000$。

迷失的字符串

给定一棵 n 个点的无根树，每条边上有个小写字符。

m 次询问，每次给定一个字符串 S，问是否存在一条路径满足沿路边上的字符拼起来恰好是 S。

- $2 \le n \le 30000$。
- $1 \le m \le 30000$。
- $\sum |S| \le 30000$。
- Source： BZOJ 4713

Solution

- 离线询问，将所有询问串拼接起来得到大串 T。

Solution

- 离线询问，将所有询问串拼接起来得到大串 T。
- 设 $f_{i,j}$ 表示 i 点往下，是否可以从某个询问串开头一直向后匹配到 T_j，且 j 已经匹配。

Solution

- 离线询问，将所有询问串拼接起来得到大串 T。
- 设 $f_{i,j}$ 表示 i 点往下，是否可以从某个询问串开头一直向后匹配到 T_j，且 j 已经匹配。
- 设 $h_{i,j}$ 表示 i 点往下，是否可以从某个询问串结尾一直向前匹配到 T_j，且 j 等待匹配。

Solution

- 离线询问，将所有询问串拼接起来得到大串 T。
- 设 $f_{i,j}$ 表示 i 点往下，是否可以从某个询问串开头一直向后匹配到 T_j，且 j 已经匹配。
- 设 $h_{i,j}$ 表示 i 点往下，是否可以从某个询问串结尾一直向前匹配到 T_j，且 j 等待匹配。
- 通过树形 DP 求出 f 和 h，在转移的过程中即可得到每个串是否出现过。

Solution

- 离线询问，将所有询问串拼接起来得到大串 T。
- 设 $f_{i,j}$ 表示 i 点往下，是否可以从某个询问串开头一直向后匹配到 T_j，且 j 已经匹配。
- 设 $h_{i,j}$ 表示 i 点往下，是否可以从某个询问串结尾一直向前匹配到 T_j，且 j 等待匹配。
- 通过树形 DP 求出 f 和 h，在转移的过程中即可得到每个串是否出现过。
- 显然 j 这一维可以用 bitset 加速。

Solution

- 离线询问，将所有询问串拼接起来得到大串 T。
- 设 $f_{i,j}$ 表示 i 点往下，是否可以从某个询问串开头一直向后匹配到 T_j，且 j 已经匹配。
- 设 $h_{i,j}$ 表示 i 点往下，是否可以从某个询问串结尾一直向前匹配到 T_j，且 j 等待匹配。
- 通过树形 DP 求出 f 和 h，在转移的过程中即可得到每个串是否出现过。
- 显然 j 这一维可以用 bitset 加速。
- 时间复杂度 $O(\frac{n\sum|S|}{w})$。

Solution

- 空间复杂度 $O(\frac{n\sum|S|}{w})$，存不下？

Solution

- 空间复杂度 $O(\frac{n\sum|S|}{w})$，存不下？
- 对树进行轻重链剖分，那么可以与重儿子共用同一个 DP 数组，轻儿子再单独开辟新空间。

Solution

- 空间复杂度 $O(\frac{n\sum|S|}{w})$，存不下？
- 对树进行轻重链剖分，那么可以与重儿子共用同一个 DP 数组，轻儿子再单独开辟新空间。
- 对于每个点，从根到它最多只会经过 $O(\log n)$ 条轻边，故只会同时存在 $O(\log n)$ 个 DP 数组。

Solution

- 空间复杂度 $O(\frac{n\sum|S|}{w})$，存不下？
- 对树进行轻重链剖分，那么可以与重儿子共用同一个 DP 数组，轻儿子再单独开辟新空间。
- 对于每个点，从根到它最多只会经过 $O(\log n)$ 条轻边，故只会同时存在 $O(\log n)$ 个 DP 数组。
- 那么可以通过树形 DP 求出 f 和 h，在转移的过程中即可得到每个串是否出现过。

Solution

- 空间复杂度 $O(\frac{n\sum|S|}{w})$，存不下？
- 对树进行轻重链剖分，那么可以与重儿子共用同一个 DP 数组，轻儿子再单独开辟新空间。
- 对于每个点，从根到它最多只会经过 $O(\log n)$ 条轻边，故只会同时存在 $O(\log n)$ 个 DP 数组。
- 那么可以通过树形 DP 求出 f 和 h，在转移的过程中即可得到每个串是否出现过。
- 空间复杂度 $O(\frac{\sum|S|\log n}{w})$。

JZPSTR

维护一个数字串 S，支持 q 次操作：

JZPSTR

维护一个数字串 S，支持 q 次操作：

(1) 给定数字串 T，往 S 某个位置插入 T。

JZPSTR

维护一个数字串 S，支持 q 次操作：

(1) 给定数字串 T，往 S 某个位置插入 T。

(2) 删除 S 某段子串。

JZPSTR

维护一个数字串 S，支持 q 次操作：

(1) 给定数字串 T，往 S 某个位置插入 T。

(2) 删除 S 某段子串。

(3) 给定数字串 T，查询 S 的某个区间中出现了多少次 T。

JZPSTR

维护一个数字串 S，支持 q 次操作：

(1) 给定数字串 T，往 S 某个位置插入 T。

(2) 删除 S 某段子串。

(3) 给定数字串 T，查询 S 的某个区间中出现了多少次 T。

- 插入次数 ≤ 1000，插入串总长度 $\leq 2 \cdot 10^6$。

JZPSTR

维护一个数字串 S，支持 q 次操作：

(1) 给定数字串 T，往 S 某个位置插入 T。

(2) 删除 S 某段子串。

(3) 给定数字串 T，查询 S 的某个区间中出现了多少次 T。

- 插入次数 ≤ 1000，插入串总长度 $\leq 2 \cdot 10^6$。
- 删除次数 ≤ 1000。

JZPSTR

维护一个数字串 S，支持 q 次操作：

(1) 给定数字串 T，往 S 某个位置插入 T。

(2) 删除 S 某段子串。

(3) 给定数字串 T，查询 S 的某个区间中出现了多少次 T。

- 插入次数 ≤ 1000，插入串总长度 $\leq 2 \cdot 10^6$。
- 删除次数 ≤ 1000。
- 询问串总长度 ≤ 10000。

JZPSTR

维护一个数字串 S，支持 q 次操作：

(1) 给定数字串 T，往 S 某个位置插入 T。

(2) 删除 S 某段子串。

(3) 给定数字串 T，查询 S 的某个区间中出现了多少次 T。

- 插入次数 ≤ 1000，插入串总长度 $\leq 2 \cdot 10^6$。
- 删除次数 ≤ 1000。
- 询问串总长度 ≤ 10000。
- 任意时刻 $|S| \leq 10^6$。

JZPSTR

维护一个数字串 S，支持 q 次操作：

(1) 给定数字串 T，往 S 某个位置插入 T。

(2) 删除 S 某段子串。

(3) 给定数字串 T，查询 S 的某个区间中出现了多少次 T。

- 插入次数 ≤ 1000，插入串总长度 $\leq 2 \cdot 10^6$。
- 删除次数 ≤ 1000。
- 询问串总长度 ≤ 10000。
- 任意时刻 $|S| \leq 10^6$。
- Source： BZOJ 2628

Solution

- 块状链表 + 后缀自动机？

Solution

- 块状链表 + 后缀自动机？
- 我可不会写。

Solution

- 块状链表 + 后缀自动机？
- 我可不会写。
- 考虑 shift-and 算法，那么只需要维护 10 个 bitset，即 $f_{i,j}$ 表示 S_j 是否是字符 i。

Solution

- 块状链表 + 后缀自动机？
- 我可不会写。
- 考虑 shift-and 算法，那么只需要维护 10 个 bitset，即 $f_{i,j}$ 表示 S_j 是否是字符 i。
- 对于修改操作，直接暴力修改 10 个 bitset 即可，时间复杂度 $O(\frac{10|S|}{w})$。

Solution

- 块状链表 + 后缀自动机？
- 我可不会写。
- 考虑 shift-and 算法，那么只需要维护 10 个 bitset，即 $f_{i,j}$ 表示 S_j 是否是字符 i。
- 对于修改操作，直接暴力修改 10 个 bitset 即可，时间复杂度 $O(\frac{10|S|}{w})$。
- 对于查询 T 在 S 中所有出现的位置，有 $ans = ans << 1 \& f_{T_i}$，时间复杂度 $O(\frac{|S||T|}{w})$。

Solution

- 块状链表 + 后缀自动机？
- 我可不会写。
- 考虑 shift-and 算法，那么只需要维护 10 个 bitset，即 $f_{i,j}$ 表示 S_j 是否是字符 i。
- 对于修改操作，直接暴力修改 10 个 bitset 即可，时间复杂度 $O(\frac{10|S|}{w})$。
- 对于查询 T 在 S 中所有出现的位置，有 $ans = ans << 1 \& f_{T_i}$，时间复杂度 $O(\frac{|S||T|}{w})$。
- 需要手写 bitset 来支持需要的功能。

课后习题

- Bipartite Graph (HDU 5313)

课后习题

- Bipartite Graph (HDU 5313)
- PolandBall and Gifts (Codeforces 8VC Venture Cup 2017 - Elimination Round F)

课后习题

- Bipartite Graph (HDU 5313)
- PolandBall and Gifts (Codeforces 8VC Venture Cup 2017 - Elimination Round F)
- Slim Cut (Hong Kong Regional Contest 2016)

课后习题

- Bipartite Graph (HDU 5313)
- PolandBall and Gifts (Codeforces 8VC Venture Cup 2017 - Elimination Round F)
- Slim Cut (Hong Kong Regional Contest 2016)
- [Jsoi2010] 连通数 (BZOJ 2208)

课后习题

- Bipartite Graph (HDU 5313)
- PolandBall and Gifts (Codeforces 8VC Venture Cup 2017 - Elimination Round F)
- Slim Cut (Hong Kong Regional Contest 2016)
- [Jsoi2010] 连通数 (BZOJ 2208)
- 由乃打扑克 (BZOJ 4812)

课后习题

- Bipartite Graph (HDU 5313)
- PolandBall and Gifts (Codeforces 8VC Venture Cup 2017 - Elimination Round F)
- Slim Cut (Hong Kong Regional Contest 2016)
- [Jsoi2010] 连通数 (BZOJ 2208)
- 由乃打扑克 (BZOJ 4812)
- Metal Processing Plant (WF 2014)

课后习题

- Bipartite Graph (HDU 5313)
- PolandBall and Gifts (Codeforces 8VC Venture Cup 2017 - Elimination Round F)
- Slim Cut (Hong Kong Regional Contest 2016)
- [Jsoi2010] 连通数 (BZOJ 2208)
- 由乃打扑克 (BZOJ 4812)
- Metal Processing Plant (WF 2014)
- Scores (HihoCoder 1236)

课后习题

- Bipartite Graph (HDU 5313)
- PolandBall and Gifts (Codeforces 8VC Venture Cup 2017 - Elimination Round F)
- Slim Cut (Hong Kong Regional Contest 2016)
- [Jsoi2010] 连通数 (BZOJ 2208)
- 由乃打扑克 (BZOJ 4812)
- Metal Processing Plant (WF 2014)
- Scores (HihoCoder 1236)
- Forgiveness (HDU 6028)

Thank you!