競プロキャンプ 2026 関西 コンテスト 問題解説
(VRC 競プロ部提供)

A - kmnk

Assam, 原案: seekworser

$km - n = k$ を変形すると $k(m-1) = n$ なので、$1 \leq k(m-1) \leq X$ を満たす組 (k, m) を数えればよいです。辺々 k $(k \geq 1)$ で割ると $1/k \leq m - 1 \leq X/k$ となります。m は整数なので、条件は $2 \leq m \leq \lfloor X/k \rfloor + 1$ と同値です。k を固定したとき、条件を満たす m の個数は $\min(X, \lfloor X/k \rfloor + 1) - 2 + 1$ です。$k = 1, 2, \ldots, X$ についてこれを足し上げれば答えが求まります。

B - Evaluate F and G

Assam, 原案: KA37RI

与えられた文字列から `f`, `g` 以外を削除したものが答えです。正当性は、以下のように各演算の結果が正しいことを確認したのち、文字列の長さについて数学的帰納法を適用すれば証明できます。

S, T を文字列とします。文字列 S に対して、文字列 X を問題文中で示した形式の式として評価して得られる文字列を $[X]$ で表すとします。文字列の連結を表す演算子 $+$ を省略します。

- $[\mathtt{f(}S\mathtt{)}] = f([S]) = \mathtt{f}[S]$
- $[\mathtt{g(}S\mathtt{)}] = g([S]) = \mathtt{g}[S]$
- $[S\mathtt{+}T] = [S][T]$
- $[\mathtt{e}] = \varepsilon$ （ε は空文字列）

C - Comfortable Roads

Benihuki, 原案: ループ

以下、町を頂点、道を辺とみなした無向グラフとして考えます。

すべての i が $D_i \geq 2$ を満たす場合

この場合、答えは $N - 1$ です。

スターグラフ（ある 1 つの頂点に他のすべての頂点が隣接するような、 $N-1$ 本の辺をもつグラフ）を構築することで、任意の頂点間の距離が 2 以下となります。星の辺の本数は $N-1$ 本です。

逆に $N-2$ 本以下の辺で連結グラフを作ることができず、条件を満たすことができません。

ある i が $D_i = 1$ を満たす場合

この場合、 $D_i = 1$ を満たす i の個数を X 個としたとき、答えは $\sum_{k=1}^{X}(N-k)$ です。

まず、 $D_i = 1$ なる頂点 i はすべての頂点と辺で直接結ばれている必要があります。

逆に、上記を満たすようにグラフを構築することで必ず全域部分グラフに星が含まれます。よって前項の議論により、 $D_i \geq 2$ を満たす各 i についての条件も満たされます。

以上ですべての場合が尽くされました。計算量は $\Theta(N)$ です。

D - Flowers

Benihuki, 原案: seekworser

以下の貪欲解法により答えを求めることができます。

- 全てのマスを辞書順に走査し、まだ削除されていない # のマス (x, y) を見つけたら $(x+1, y)$ を中心とする花を答えに追加し、その花を構成する 5 マスを盤面から削除する。

この解法が正しいことを示します。

残っている # のうち、行番号が最小であり、行番号が同じものの中では列番号が最小であるマスを (x, y) とします。

(x, y) を含む花の中心の候補は高々 5 通りです。しかし、$(x+1, y)$ 以外を中心とする花は、必ず (x, y) より先に走査されるマスを含みます。そのようなマスはすでに削除されているか、初めから . であるため、その花を配置することはできません。

したがって、(x, y) を含む花の中心は必ず $(x+1, y)$ です。この花を削除した後の盤面も、残りの花を互いに重ならないように配置して得られた盤面です。よって、同じ操作を繰り返すことで、すべての花を正しく求められます。

各マスを一度ずつ調べ、花を見つけるたびに定数個のマスを削除するため、計算量は $O(HW)$ です。

E - Join Wars

Ceylon, 原案: ゆ〜

基本的にはインスタンスごとに待機列の queue を作成して下記の操作をしながらそれぞれの居場所をシミュレーションをしていけば良いです

1. 最後尾の待機列の queue に入る。ただし、インスタンスが空いている場合はそのまま入る。
2. インスタンスから離れた際の空きに queue の先頭の人を入れる。
3. 待機列から離れた際に queue から削除する。

操作 1, 2 は一般的な queue 処理になります。操作 3 の処理は list や queue で愚直な実装すると一回のクエリごとに $O(Q)$ がかかってしまいますが、部分点は達成可能となります。

満点解法の場合は、例えば操作 1 の際にプレイヤー番号とその時のクエリ番号を tuple として追加します。同時に各プレイヤーが最後に移動したクエリ番号を記憶しておくことで、操作 2 の際にクエリ番号が食い違う場合は削除されたと認識でき、インスタンスに入れずに pop のみをして操作を中断することで $O(1)$ の削除ができます。全体的な queue 処理の計算量は追加と削除で高々 $2Q$ 回しか行わないため $O(N+M+Q)$ で解くことができます。

F - Keep Far

Ceylon, 原案: seekworser

まず、隣り合う # がない場合 $K!$ が答えになります。

次に、隣り合う # がある上で両隣が空いている席がある場合 0 が答えになります。

そうではない場合、ある瞬間にどの席も両隣が空いていることがない状態に初めてなる瞬間があります。この瞬間に座っている人数が X であった場合、答えは $X!(K-X)!$ になります。

このような場合の数え上げは # ある場所を左から DP をして、そこまでの場所で隣を避けつつ三つ連続空かないように席に座った時に、席に座ってる人数を同時に保存した二次元 DP を作成すれば良いです。具体的には、2 個前と 3 個前の状態からの合計で遷移します。

この数え上げを行いそれぞれで $X!(K-X)!$ を乗算した合計を求めることで解くことができます。

計算量は $O(NK)$ です。

G - Point Query on Disjoint Rectangles

Ceylon, 原案: mizarjp

この問題は平面走査とクエリの先読みという典型を使用した問題になります。

クエリの先読みは、クエリをすべて読み込んでおける場合に、回答をする手法です。平面操作は、二次元のクエリに対して一つの軸に対してソートなどをして順で差分更新をしていく手法です。

今回の場合、クエリの情報を先に読んでおき二次元の長方形の下辺、上辺、クエリの三つの要素をまとめて X 軸方向にソートして処理をすることができます。下辺が来た場合領域を追加し、上辺が来た場合領域を削除し、クエリが来た場合にはそのクエリに回答をすればよいです。

今回は長方形の領域の重複がないため、例えば下辺、上辺の処理は set で Y 座標の区間を記憶しておき、クエリが来た際にはそのクエリの Y 座標の区間が存在するかを二分探索することができ、どちらも計算量は $O(\log(N+Q))$ となります。セグメント木や遅延セグメント木を用いても計算量 $O(\log \max(X))$ で操作、探索ができます。

計算量は最初のソートも含め $O((N+Q)\log(N+Q))$ になります。

H - Max Bust Darts

Darjeeling, 原案: あめんばー

どのような場合でも最後に選択する A_i は選んだ K 個の中の最大です。A をソートして昇順に探索を行うことで、現在探索している数字がこれまでの最大として扱うことができます。

ある M 未満の X を作るために必要な数字の最小個数を保存しておく DP を作成します。この DP の更新は愚直に行うと $O(M^2)$ かかりますが、昇順に $\bmod A_i$ が同一のものの中の累積 min を記録することにより $O(M)$ で更新することができます。

また最終的な累積 min が K 未満となっている余りについては、そのまま A_i を最後に選択するものとして $O(\min(A_i, M))$ で答えを更新することができます。

これにより $O(N\log N + NM)$ で解くことができます。

別解

N, K が小さいので工夫をすれば $O(NKM)$ なども通すことは可能です。

例えば A_i を使用した個数と、到達可能である X の boolDP 考えます。この配列を K 個分の bitset で持ち、更新を $A_i, 2A_i, 4A_i, 8A_i \ldots$ を bitshift + or を行ったダブリング

の更新を同時に重ねることによって $O(KM \log M)$ で更新することができます。計算量は $O(NKM \log M)$ ですが、bitset による 64 倍の高速化が入っているため $O(NM)$ に十分近い計算量が可能です。

その他メモリ管理の最適化、下記の K の範囲などの枝刈り、高速な言語を使用するなど、非常に多くの定数倍高速化をすれば $O(NKM)$ を間に合わせることも可能です。

K の上限

$K \geq 17$ であれば、より具体的には $K2^{K-1} \geq M$ であれば $K = N$ と同義になります。

最後の操作の直前の $X(< M)$ について考えます。この X を表現するために使用する最小の集合 $B_1, \ldots, B_t$ について考えます。

この時任意の i について多重集合 $\{\sum_{j \in S} B_j \mod B_i \mid S \subset \{1, \ldots, i-1, i+1, \ldots t \quad \}\}$ は全て異なります。同値の組み合わせがあった場合、その組み合わせの差を B_i で調整することにより、ある B の使用回数を 0 にすることができ t の最小性を満たしません。この時の S の組み合わせは 2^{t-1} 通りのため、鳩の巣原理から $B_i \geq 2^{t-1}$ を満たします。さらに $X \geq t2^{t-1}$ を満たすため、 $t2^{t-1} \leq X < M$ となりこれを示すことができます。

例えば下記の集合は $t = 16$ かつ $\sum A < M$ を満たします

$\{2^{16} - 1, 2^{16} - 2, \ldots, 2^{16} - 2^i, \ldots, 2^{16} - 2^{15}\}$

I - Bubble Sort Problems

Darjeeling, 原案: ループ

$\mathrm{pos}[x]$ を、 時刻 0 時点で $P_i = x$ を満たす i の値と定義します。

$\mathrm{pos}[X_k] < \mathrm{pos}[Y_k]$ を満たすとき、はじめから条件を満たしているため k 番目の質問の答えは 0 です。以下、 $\mathrm{pos}[X_k] > \mathrm{pos}[Y_k]$ の場合を考えます。

部分点解法

バブルソートのシミュレーションは $O(N^2)$ 回の操作で $P = (1, 2, \cdots, N)$ にできます。

よって、問題文の操作を実際にシミュレーションをし、 P の i 番目が $i + 1$ 番目の場所を入れ替えたとき、その二数と時刻を map や dict で記録することで解くことができます。

計算量は全体で $O(N^2 + Q)$ や $O((N^2 + Q) \log Q)$ です。

満点解法

cnt_k を「時刻 0 時点で X_k よりも左に位置する、 Y_k よりも大きい値の個数」と定義します。この cnt_k は、クエリ先読みをして fenwick tree で求めたり、 Wavelet Matrix で求めたりすることができます。

そうしたとき、答えは $\mathrm{cnt}_k(N-1) + \mathrm{pos}[X_k] - \mathrm{cnt}_k$ となります。

計算量は全体で $O((N+Q)\log N)$ です。

簡単な説明

X_k と Y_k を入れ替えるまでに、以下の二つステップで操作が行われます。

1. cnt_k 回だけバブルソートの一周を行われる。
2. その後、 X_k と Y_k を入れ替えるまでに $\mathrm{pos}[X_k] - \mathrm{cnt}_k$ 単位時間かかる。

1 ステップ目については、 X_k より左に位置する Y_k よりも大きい値を動かす必要があり、そのためにバブルソートの一周が cnt_k 回行われます。

2 ステップ目については、cnt_k 回のバブルソートの一周が完了した時点で、

- X_k より左にある値はすべて Y_k 以下である。
- 時刻 0 時点で X_k より左にある Y_k 以下の値は、 X_k より右に位置することはない。

の両者を満たすため、 X_k の位置が $\mathrm{pos}[X_k] - \mathrm{cnt}_k$ 番目となります。

J - Substring Query

Darjeeling, 原案: cleantted

複数の文字列アルゴリズムで、それぞれ解くことができます。

ローリングハッシュ

文字列 S のすべての部分文字列についてローリングハッシュを計算し、長さで分類して hashset で管理しておきます。

クエリで文字列 t が与えられたとします。t の l 文字目から始まる部分文字列のうち S の部分文字列でもあるものの最大の長さを x_l とおきます。$x_l - 1 \leq x_{l+1}$ より、尺取り法で x_l $(l = 1, 2, \ldots, |t|)$ はすべて尺取り法で求まります。x_l $(l = 1, 2, \ldots, |t|)$ の最大値が答えです。

計算量は $O(|S|^2 + \sum_{i=1}^{Q} |T_i|)$ です。

Aho–Corasick 法

S のすべての接尾辞について Aho–Corasick 法の trie を構築します。trie の子を平衡二分探索木で管理し、failure link のみ計算（つまり、「各文字 c について、文字 c を読み込んだときに failure link をたどった後に行くつく場所」は計算しない）すれば、必要な空間は $O(|S|^2)$ です。

クエリで文字列 t が与えられたとき、Aho–Corasick 法で t を検索し、k 文字目を読み込んだ直後に見ているノードは、t の k 文字目が右端であるような部分文字列であって S の部分文字列に一致するもののうち、長さが最大のものです。よって、t を検索したときに訪れた各ノードに対応する文字列の長さが分かれば、クエリの答えが求まります。

計算量は $O((|S|^2 + \sum_{i=1}^{Q} |T_i|) \log \sigma)$ (σ は文字種数) です。

Suffix Array

$|S|$ が大きい場合でも Suffix Array, LCP Array を用いることでで解くことができます。

具体的には文字列を $T_1 + \texttt{\#} + T_2 + \texttt{\#} + \cdots + T_Q + \texttt{\#} + S$ のように区切りを入れて結合した文字列による Suffix Array および LCP Array を作成します。

この時 S の末尾は `#` を持たないため、その接尾辞からを先頭とした T_i の部分文字列は、Suffix Array の順で直近の前か後ろの S の接尾辞と最も先頭が一致します。

そこから具体的な一致文字数を求めるには、累積した共通部分文字列の長さを $X = 0$ で初期化した後に、作成した Suffix Array の順に探索し、先頭が S の位置である接尾辞では X を接尾辞の長さに更新します。 先頭が T_i の位置に該当する接尾辞では LCP Array の値で X を min で更新し、 T_i の答えを X で max に更新してゆくことで、先に現れた S の接尾辞との一致文字数を計算することができます。

後に現れる S の接尾辞に関しては、逆から探索して同様にすればよいです。

計算量は $O((|S| + \sum_{i=1}^{Q} |T_i|) \log (|S| + \sum_{i=1}^{Q} |T_i|))$ です。

K - 5D Polyomino

Earlgray, 原案: TwoSquirrels

回転行列としてありうるものは、非ゼロ要素が各行各列 1 つずつの対角行列であり、行列式が 1 となるもののみです。非ゼロ要素の位置の選び方は $K!$ 通りあります。非ゼロ要素の値は ± 1 なので組み合わせは 2^K 通りありますが、行列式が 1 となるのはそのうち 2^{K-1} 通りです。よって、回転の方法は $K!2^{K-1}$ 通りであり、特に $K = 5$ のときは 1920 通り

です。

与えられた 3 つの立体を、順に A, B, C とします。各 K 次元ポリオミノについて、回転後に占めるマスの組み合わせ（それぞれ高々 1920 通り）を列挙します。K 次元ポリオミノが占めるマスの組み合わせは、占めるマスを 1、それ以外を 0 としたビット列で表せます。本問題の制約下では $2^K \leq 32$ なので、以降 2^K ビットの演算の計算量を $O(1)$ とします。

A は回転せず、B, C を回転して合わせる場合を考えれば十分です（対称性より）。

B の回転を全探索します。すると、B の回転を決めるごとに、C が占めるべき領域が定まります。この領域に一致するような C の回転パターンが存在するかどうかの判定は、hashset などで高速化できます。

以上により、本問題は $O(f(K) \log f(K))$ $(f(K) = K! 2^{K-1})$ 程度の計算量で解けます。

出題時の設定においては、$O((f(K))^2)$ 程度に高速であれば正解とするようにしています。

L - Inconvenient Dice

Earlgray, 原案: seekworser + PCTProbability

立式

N^k を M で割った余りを r_k とすると、求める値は $\sum_{k=0}^{\infty} r_k N^{-k}$ と表せます。以下でこれを示します。

操作が終了するまでにサイコロを振る回数を T として、 任意の公平な操作方法について $\mathbb{E}[T]$ の下界を求めます。

正整数 k を固定します。サイコロを k 回振った結果としてあり得る目の列は N^k 個あり、それぞれ確率 $\frac{1}{N^k}$ で現れます。（実際には $d < k$ 回目で操作が終了する場合もサイコロは振り続けたものとして数えることとします。）

各 $i = 1, 2, \ldots, M$ について、k 回以内に i を出力して終了するような長さ k の目の列の個数を A_i とします。ここで、実際には $d < k$ 回目で操作が終了する場合、その後に任意の $k - d$ 個の目を付け加えた N^{k-d} 個の列をすべて数えるものとします。

すると、k 回以内に i を出力して終了する確率は $\frac{A_i}{N^k}$ です。

操作方法は公平なので、有限回のうちに i を出力して終了する確率は $1/M$ です。したがって $\frac{A_i}{N^k} \leq \frac{1}{M}$ であり、A_i は整数なので $A_i \leq \left\lfloor \frac{N^k}{M} \right\rfloor$ が成り立ちます。

よって、k 回以内に操作が終了する長さ k の目の列は、高々 $M \left\lfloor \frac{N^k}{M} \right\rfloor$ 個です。したがって、k 回振った時点でも操作が終了していない確率について

$$\Pr(T > k) \geq \frac{N^k - M \left\lfloor \frac{N^k}{M} \right\rfloor}{N^k} = \frac{N^k \bmod M}{N^k}$$

が成り立ちます。

非負整数値をとる確率変数について

$$\mathbb{E}[T] = \sum_{k=0}^{\infty} \Pr(T > k)$$

であるため、任意の公平な操作方法について

$$\mathbb{E}[T] \geq \sum_{k=0}^{\infty} \frac{N^k \bmod M}{N^k}$$

を得ます。

次に、この下界を達成する操作方法を構成します。

$r_k = N^k \bmod M$ とおきます。特に $r_0 = 1$ です。

$k-1$ 回振った時点で、まだ操作を終了していない長さ $k-1$ の目の列がちょうど r_{k-1} 個あるとします。それらについてもう一度サイコロを振ると、長さ k の目の列は Nr_{k-1} 個得られます。

ここで $Nr_{k-1} = Mq_k + r_k,\ 0 \leq r_k < M$ となる整数 q_k, r_k をとり、この Nr_{k-1} 個の目の列のうち、各 $i = 1, 2, \ldots, M$ に q_k 個ずつを割り当て、その列が現れたときには i を出力して終了します。残った r_k 個については、さらにサイコロを振ります。

$r_0 = 1$ および $r_k = Nr_{k-1} \bmod M$ より、帰納的に $r_k = N^k \bmod M$ が成り立ちます。よって、$\Pr(T > k) = \frac{r_k}{N^k}$ です。したがって、$\mathbb{E}[T] = \sum_{k=0}^{\infty} r_k N^{-k}$ です。

計算

r_k は $r_0 = 1,\ r_{k+1} = Nr_k \bmod M$ という漸化式で求められます。

r_k がとり得る値は $0, 1, \ldots, M-1$ の M 通りしかないため、数列 $r_0, r_1, \ldots$ は高々 M 回の遷移の後に、以前現れた値に戻ります。r_k の第 s 項 (r_s) 以降が周期 L をもつとすると、求める値は

$$\sum_{k=0}^{s-1} \frac{r_k}{N^k} + \sum_{t=0}^{\infty} \sum_{j=0}^{L-1} \frac{r_{s+j}}{N^{s+j+tL}}$$

と表せます。ここで現れた無限和は等比級数なので、計算すると

$$\sum_{k=0}^{s-1} \frac{r_k}{N^k} + \frac{\displaystyle\sum_{j=0}^{L-1} \frac{r_{s+j}}{N^{s+j}}}{1 - \frac{1}{N^L}}$$

となります。$N^{-k}\ (k = 0, 1, 2, \ldots)$ を前計算したのちに、この式を式の通りに計算すれば正解となります。計算量は $O(M + \log P)$ です。

計算上必要なこととして、式中で分母として現れる値 N^{s+j} および $1-\frac{1}{N^L}$ が、制約下において $\bmod 10^9+7$ で 0 と等しくならないことを示します。

$P=10^9+7$ とおきます。制約より $2 \leq N \leq 10^6 < P$ なので、N は P と互いに素です。よって、N^{s+j} は $\bmod P$ で 0 と等しくなりません。

P が安全素数であること、つまり P が素数であってかつ $P-1$ の正の約数は $1, 2, (P-1)/2, P-1$ に限られることに注意します。$\bmod P$ における乗法位数としてあり得る値は $1, 2, (P-1)/2, P-1$ に限られます。特に、$N \not\equiv \pm 1 \pmod P$ より、N の乗法位数は $(P-1)/2$ 以上です。これと $1 \leq L \leq M \leq 10^6$ より、$N^L \neq 1$ が導かれ、したがって $1-\frac{1}{N^L}$ は $\bmod P$ で 1 と等しくないことが示されます。

M - Rotate 2x2

Earlgray, 原案: ゆ〜 + あめんばー

操作可能な 2×2 の部分を列挙し、マスを共有するような操作どうしが隣接するとしたときの連結成分を求めておきます。異なる連結成分どうしは独立するので、各連結成分についての答えの積が全体の答えになります。

以降、1 つの連結成分について、実現可能なチップの並べ替え方の個数を求めることを考えます。

チップの並べ替え方全体を探索する BFS 等により、以下の場合についての解が得られます。

```
(1)     (2)     (3)     (4)
..      ....    ...     ..
...     ....    ...     ..
 ..
```

- 移動できないチップ、つまり 1 つのチップからなる同値類について、並べ替え方は 1 通りです。
- (1), (2) の場合、すべての並べ替え方が実現可能である。
- (3) の場合、実現可能な並べ替え方は 120 通りである。
- (4) の場合、実現可能な並べ替え方は 4 通りである。

これ以外の形の同値類は、(1) または (2) を部分として含むので、数学的帰納法によってすべての並べ替え方が実現可能であることが示せます。具体的には、(1) または (2) を含み、すべての並べ替え方が実現可能である形があるとき、マスを 1 つ以上共有するような 2×2 の領域を選んで合併したような形でも、すべての並べ替え方が実現可能であることが示せます。

計算量は $O(HW)$ です。

おまけコンテンツ

- https://brief.mathenachia.blog/media/kyoprocamp2026west/am3mbjqbpsqqeggq.html

N - Equipment Collector

Flavor, 原案: Nachia

本問題は、同原案者の過去の出題 [1] を発展させて作成した問題です。

- [1] yukicoder No.2463 ストレートフラッシュ (https://yukicoder.me/problems/no/2463)

$f = 0$ とし、最終的に集めるアイテムが $a_1, a_2, \ldots, a_m$ 番目であると仮定すると、以下のような手順が操作回数を最小化します。

1. a_1 番目のアイテムを獲得していない間、1 回の操作で不要なアイテムを M 個ずつ破棄する。
2. $\cdots$
3. a_i 番目のアイテムを獲得していない間、1 回の操作で不要なアイテムを $M - i + 1$ 個ずつ破棄する。
4. $\cdots$
5. a_M 番目のアイテムを獲得したら、操作をやめる。

操作回数は、計算量 $O(M)$ のシミュレーションで求まります。

次に、$f = 0$、a_M のみを固定します。$a_1, \ldots, a_{m-1}$ のどれかの値を増加させたとき、各回の操作後について獲得済みのアイテムの個数が減少しないので、全体の操作回数は増加しません。よって、a_M 番目のアイテム以外に必要なアイテム $M - 1$ 種類は、それぞれなるべく遅く出現するものを選ぶのが最適です。このような選び方は、すべての組 (種類, 色) ごとにそのアイテムの出現位置のリストを作成し、そのリストを二分探索することで、すべての a_M に対して計算量 $O(NM \log N)$ で求まります。

次に、f を変化させます。簡単のため $f = 0$ のとき $a_M = N$ だとすると、同様の獲得パターンが $0 \leq f \leq a_0 - 1$ で実行可能です。この区間内において、次の 2 つのことが示せます。

- f が増加したとき、操作回数の最小値が増加することはない。
- f が 5 だけ増加すると、操作回数の最小値が 1 だけ減少する。

よって、操作回数の最小値は、ある b を用いて $\lfloor -(1/5)f + b \rfloor$ $(0 \leq f \leq a_0 - 1)$ と表せ

ます。

以上、$f = 0$ で $a_M = N$ となる場合を示しましたが、それ以外のアイテムが最後になる場合も、区間が巡回しうるということを除いて同様です。実行可能な f の区間が巡回する場合は、2 つの区間に分割すれば巡回しないものとして処理できます。

最後に、各 f について、その f で実行可能なケースに対する b の最小値が求まれば、本問題に答えられます。区間 chmin クエリをバッチで処理できればよいので、例えば遅延セグメント木を用いれば十分高速です。計算量は $O(NM \log N)$ です。