ICPC JAG夏合宿 2023 Day3 オンサイト解説

A: ジェットコースター

原案 : rika0384
問題文 : tsutaj
データセット : smiken
解答 : beet, ei1333, hos, kotatsugame, smiken, tatyam, tsutaj
解説 : tsutaj

問題概要

- ジェットコースターに N グループ並んでいる
  - i 番目のグループは a_i 人いる
  - グループ番号の昇順でジェットコースターに乗れる
- ジェットコースターは M 人乗り（M 人未満で運行してもよい）
- グループのメンバー全員が同じ回のジェットコースターに乗る必要がある
- 各グループについて、そのグループがジェットコースターに乗るために待たなければならない回数の最小値を求めよ
- 制約
  - N <= 10^5
  - M <= 10^9
  - 1 <= a_i <= M

解法

- 以下のアルゴリズムにより、1 番目のグループから順に答えを求められる
  - 答え ans の初期値を 0 に、乗客人数 p の初期値を 0 にする
  - 以下を 1 <= i <= N の範囲で繰り返す。実行後の ans がグループ i の答え:
    - p + a_i <= M を満たすとき
      - p += a
      - ans は変化なし
    - p + a_i > M を満たすとき
      - p = a
      - ans が 1 増える
- 計算量 O(N)

統計情報

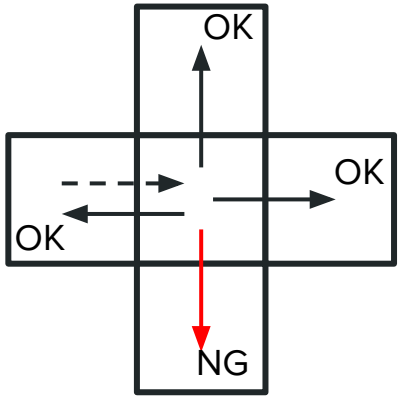
- Acceptances
  - 22 teams
- First Acceptance
  - Spead Star (1 min)

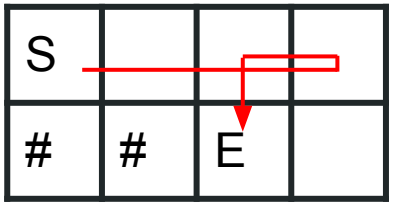
B: Break a Prison

原案 : amylase
問題文 : blue_jam
データセット : riantkb
解答 : blue_jam, ei1333, hos, kotatsugame, riantkb, tatyam

問題概要

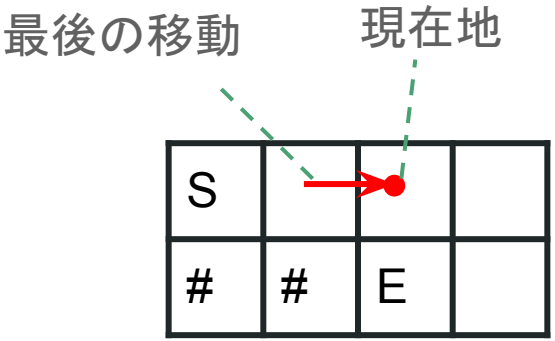
- ｎｘｍの二次元格子状の迷路（牢屋）のスタートから出口まで移動するときの最小の移動距離を見つける。
- ただし、右折をしてはいけない。（Uターンは可能）
- 原案者は、原付の免許を取って二段階右折をみた時に思いついたらしい。





解法

- 現在の部屋の位置＋最後に移動した方向（上下左右）の組を状態として、幅優先探索をする
- 計算量: O(nm)



なら、状態は (i = 1, j = 3, 右) のように持つ

統計情報

- Acceptances
  - 22 teams
- First Acceptance
  - Speed Star (6 min)

C: Camp room assignment

原案：beet
問題文：chatGPT (w/ riantkb)
データセット：beet
解答：beet, hos, kotatsugame, tatyam

問題概要

- 整数 M が与えられる
- n = 1, ..., M について、「長さ 2n の整数列であって、以下の条件を全て満たすもの」の個数を求めよ
  - 各要素は 1 以上 M 以下
  - 1 ~ M それぞれの出現回数が n 回以下
    - 例 : [1, 2, 1, 2] -> OK, [2, 2, 2, 1] -> NG

考察

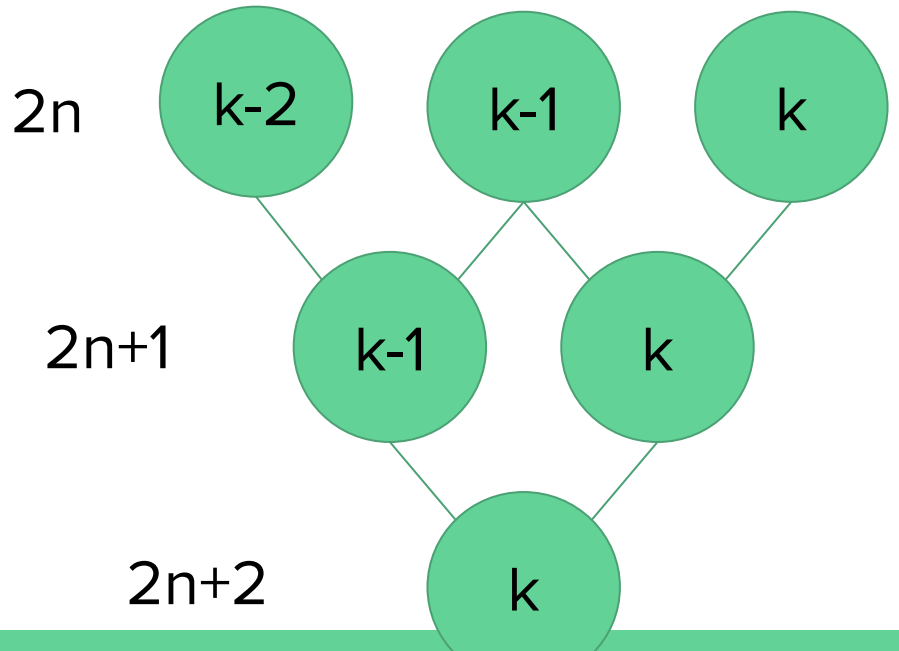
- 補集合:「n 回より多く登場する整数が存在するような列」の個数を求める
  - n 回より多く登場する整数は高々一つ
    - $(n + 1) + (n + 1) > 2n$
  - 対称性から「1 が過半数を占める列」の個数の M 倍になる
    - 「1 が過半数を占める列」と「2 が過半数を占める列」の個数は同じ

解法 (FPS)

- 1 に関する指数型母関数: f(x) = e^x
  - 次数 i の項の係数は 1 / i!
- 2 ~ M に関する指数型母関数: g(x) = (e^x)^(M-1) = e^((M-1)x)
  - 次数 i の項の係数は (M-1)^i / i!
- 「f 側の項の次数」が「g 側の項の次数」より大きいところに関して畳み込み
  - 分割統治
    - f の [l, m) 次の項と g の [m, r) 次の項を畳み込むのを再帰的にやる
    - O(M log^2 M)

解法 (算数)

- \sum[0<=k<n] binom(2n, k) (M-1)^k
  - binom(2(n+1), k) = binom(2n, k-2) + 2 binom(2n, k-1) + binom(2n, k) を使って順番に計算する
  - 全体で O(M)



統計情報

- Acceptances
  - 8 + 3 teams
- First Acceptance
  - tonosama (59 min)

D: Many-hued Tree

原案 : mtsd
問題文 : smiken
データセット : mtsd
解答 : hos, kotatsugame, mtsd, smiken
解説 : mtsd

問題概要

- N 頂点の木が与えられる
- 長さ N の順列 P で、以下の条件を満たすものはいくつあるか
  mod 998,244,353 で答えよ
  - 頂点 i に色 P_i を塗ったとき、以下の操作を繰り返して1 頂点にできる
  - 操作：塗られている色番号の差の絶対値が1 の隣接する頂点 A, B をマージして、どちらか一方の色にする
- 制約
  - N <= 2,000

解法（必要十分条件パート）

- 「操作を繰り返して 1 頂点にできる」という条件は以下の条件と同値
- 色 1 で塗られている頂点から昇順にマージできる最大の色番号を X、
  色 n で塗られている頂点から降順にマージできる最小の色番号を Y
  としたときに、X+1 >= Y となる
  - つまり、「1 -> 2 -> ... とマージ」と「n -> n-1 -> ... とマージ」の2種類を組み合わせて
    上手くいくことが必要十分条件

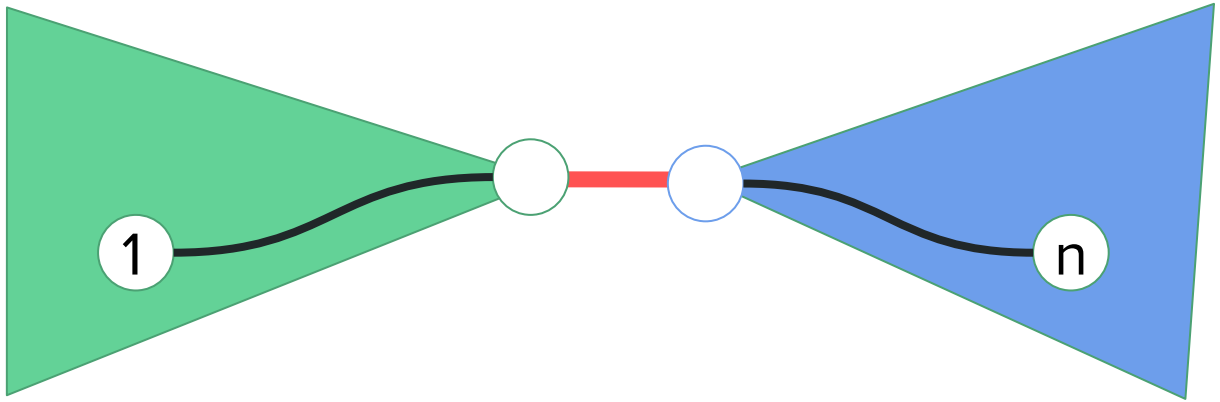
- （証明の概略）
  - 残っている色が区間 [L, R] であるとき、L < x, x+1 < R であるようなペア (x,x+1) をマージしてしまうと、色の区間が分かれるので全てをマージすることは不可能になる
  - よって、必ず (L, L+1) か (R-1, R) のどちらかがマージされる

解法（数え上げパート 1）

- 色 1 で塗られている頂点から昇順に**全ての頂点を**マージできる順列の総数は O(n) で計算可能
  - 色 1 で塗られている頂点が固定されている場合：
    - 条件を満たす順列は
      「色 1 で塗られている頂点を根とした根つき木のトポロジカル順序」
      と対応する
    - 木 DP により O(n) で計算可能
    - 実は n! / (各頂点の部分木サイズの総積) に等しい（※この事実を使わなくても解ける）
  - 色 1 で塗られている頂点が固定されていない場合：
    - 各頂点に対して、色 1 で塗られている場合の順列の総数を計算したい
    - 全方位木 DP により O(n) で計算可能

解法（数え上げパート 2）

- 色 1 側のマージと色 n 側のマージを分ける辺を固定する
  - 辺を固定すると、それぞれの側の色の塗り方はO(n) で計算可能
  - 全ての辺に対して行うと全体でO(n^2)
  - そのまま総和を取ると重複して数え上げてしまう（後述）

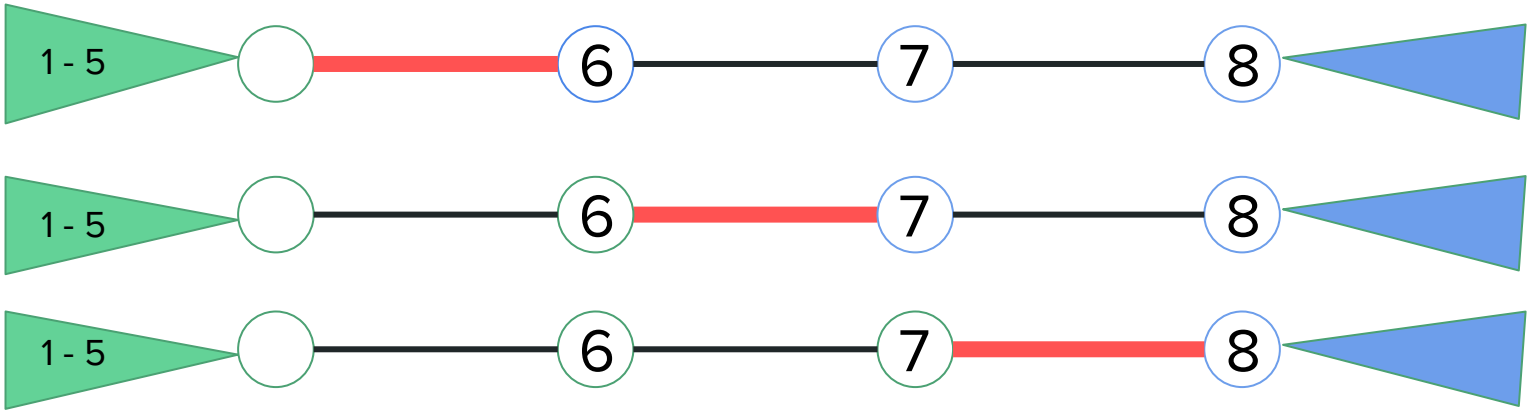


緑色のゾーンが色 1 側, 青色のゾーンが色 n 側

解法（数え上げパート３）

- 同じ書き込み方でも「色 1 側のマージと色 n 側のマージを分ける辺」が複数通り考えられることがあり、重複を引かなければならない
  - そのような辺集合はパスを成し、かつ、そのようなパスの途中には分岐がない
    - 色 1 側のサイズが k で、マージしてできる頂点が、色 k+1, k+2, … , k+m というパスに接続していて k+1, …, k+m が塗られている頂点の次数が 2

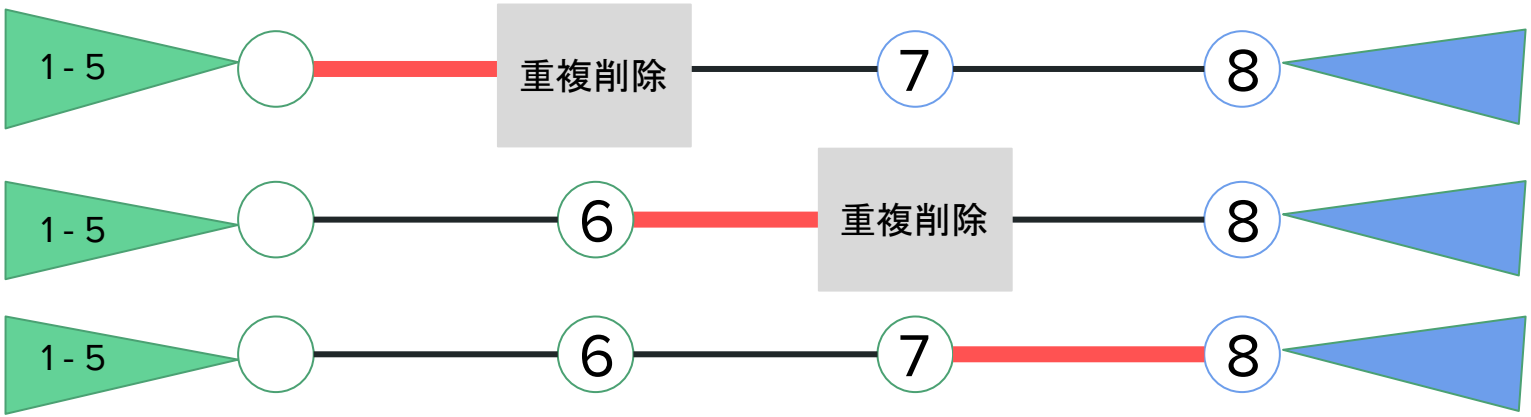
例



解法（数え上げパート 4）

- 次数 2 の各頂点に対して、「その頂点を取り除いて、片側を色 1 側、もう片側を色 n 側と場合の数」を引けば重複が取り除ける
  - 「色 1 から限界までマージしたパターンのみ」を数え上げると解釈できる

例



解法（数え上げパート まとめ）

- 色 1 側のマージと色 n 側のマージを分ける辺を固定
- 分けた各側について独立に、全方位木 DP で順列の総数を計算
- 重複分を引くために、次数 2 の頂点に対して、
  「その頂点を取り除いて、片側を色 1 側、もう片側を色 n 側とした場合の数」を全方位木 DP で計算する

- 計算量: O(n^2)

統計情報

- Acceptances
  - 0 + 1 teams
- First Acceptance
  - SPJ (292 min)

E: Gacha 101

原案 : hos
問題文 : chatGPT (w/ riantkb)
データセット : climpet
解答 : beet, climpet, hos, kotatsugame

問題概要

- 長さ N の文字列があり、始めは全部 ‘0’ で初期化されている。
- N 種類のボールがある。i 種類目のボールは A_i 個ある。
- ボールをランダムな順序で一個ずつ引いていく。i 種類目のボールを引いたとき、文字列の i 番目が ‘0’ であれば ‘1’ に変更する。
- 全部のボールを引き終えるまでに、(連続) 部分文字列として “101” が出現する確率を求めよ。

制約

- $1 \leq N \leq 2 \times 10^5$
- $1 \leq A_i$

問題の言い換え

- 同じ種類のボールを 2 回目以降に引いても実質何も起きないとみなしてよい。したがって、この問題は以下のように言い換えられる。
  - N 種類のボールが一個ずつある。これらをランダムな順序で引いていく。次に引くボールが i 種類目のものである確率は、A_i に比例する。(以下略)

考察

- 余事象である、「部分文字列として “101” が一度も出現しない」という事象の確率を求めることにする。
- この事象は以下の事象と同値であることがわかる。
  - 途中経過において、すべての‘1’ は連続する。つまり、任意の時点で 00…0011…1100…00 という形の文字列にしかならない。
- 逆に、10…01 のように、二つの ‘1’ の間に ‘0’ が挟まる場合、必ずどこかの時点で “101” という部分文字列が出現してしまう。

解法

- 最初に引くボールの番号を f とする。部分文字列として “101” が出現しないためには、次の 2 つの条件をともに満たさなければならない。
  - f より番号の小さいボールは、番号の降順に引かなければならない。
  - f より番号の大きいボールは、番号の昇順に引かなければならない。
- これら 2 つの条件は独立である。したがって、それぞれの確率を求めた後、掛け合わせてよい。それぞれの確率は簡単な漸化式で求められる。
- すべての f についての上記の積の重み付き和が (余事象の) 確率となる。
- 計算量は全体で $O(N)$ で実装できる。

統計情報

- Acceptances
  - 13 teams
- First Acceptance
  - tonosama (31 min)

F: Digit-only subrectangles

原案 : smiken
問題文 : climpet
データセット : smiken
解答 : hos,ei1333,smiken

問題概要

3	**X**	**2**	**7**
1	**9**	**8**	**X**

マス目があり、数字か記号が書き込まれている。
記号マスを含まないすべての長方形領域について、
数値の和の2乗を計算し、その合計を求めよ

例

3	4
7	X

記号マスを含まないすべての長方形領域について、
数値の和の2乗を計算し、その合計を求めよ

3	4
7	X

3	4
7	X

3	4
7	X

3	4
7	X

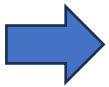
3	4
7	X

3^2 + 4^2 + 7^2 + (3+4)^2 + (3+7)^2 = 223

問題の言い換え

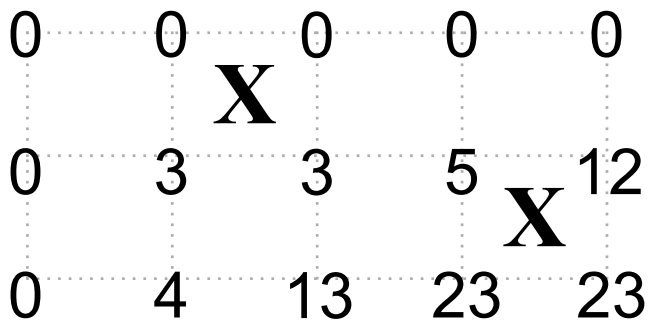
2次元累積和を取る

3	**X**	**2**	**7**
1	**9**	**8**	**X**



0	0	0	0	0
	3	**X**	**2**	**7**
0	3	3	5	12
	1	**9**	**8**	**X**
0	4	13	23	23

問題の言い換え



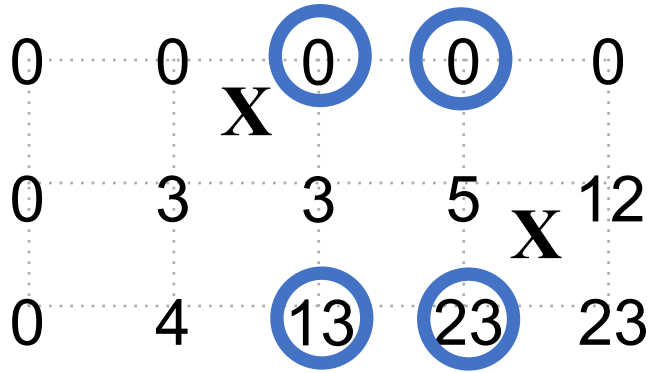
H＊Wのグリッドの格子点に数が書かれ、
いくつかのマスには記号が書かれている。

長方形になるように頂点を4つ選ぶ
スコアは頂点の数の足し引きの2乗

4頂点で囲われた領域の内部に記号がある場合は無視

スコアの言い換え

3	X	2	7
1	9	8	X

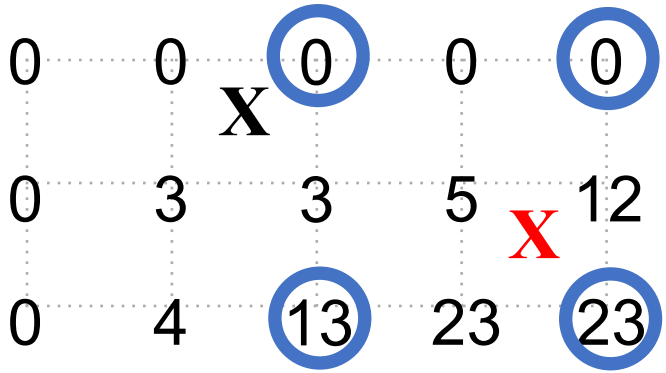


スコアは(2+8)^2

スコアは(23-13-0+0)^2

スコアの言い換え

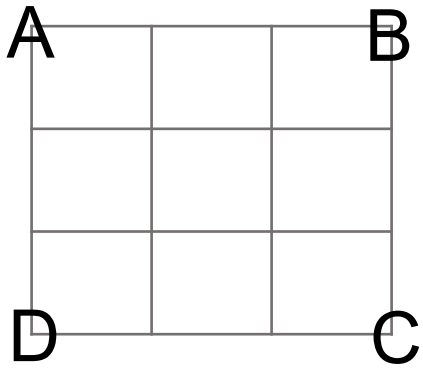
3	X	2	7
1	9	8	X



選んだ領域内に記号があるのでスコアは加算しない

四隅で囲った内側に記号があるので加算しない

スコアの言い換え



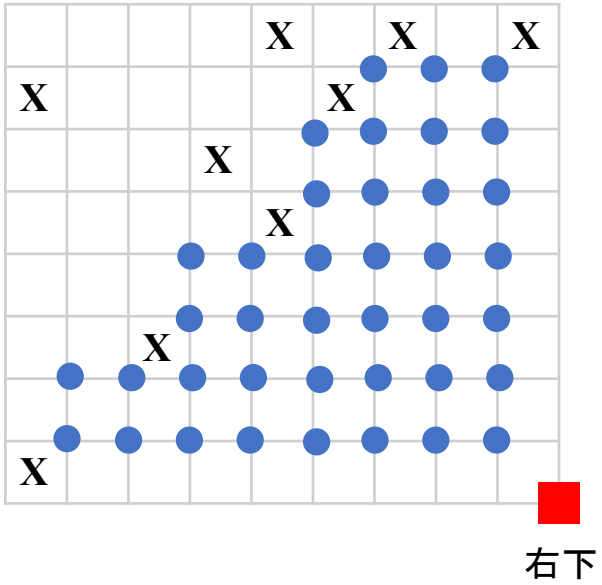
こんな4頂点を選ぶと、
左上:A
右上:B
右下:C
左下:Dとして

スコアは**(A-B+C-D)^2**

これを記号マスを含まない全長方形に対して
和を取ればよい。

Score　= Σ(A-B-C+D)^2
　　= Σ (A^2+B^2+C^2+D^2+2AC+2BD-2AB-2BC-2CD+2AD)

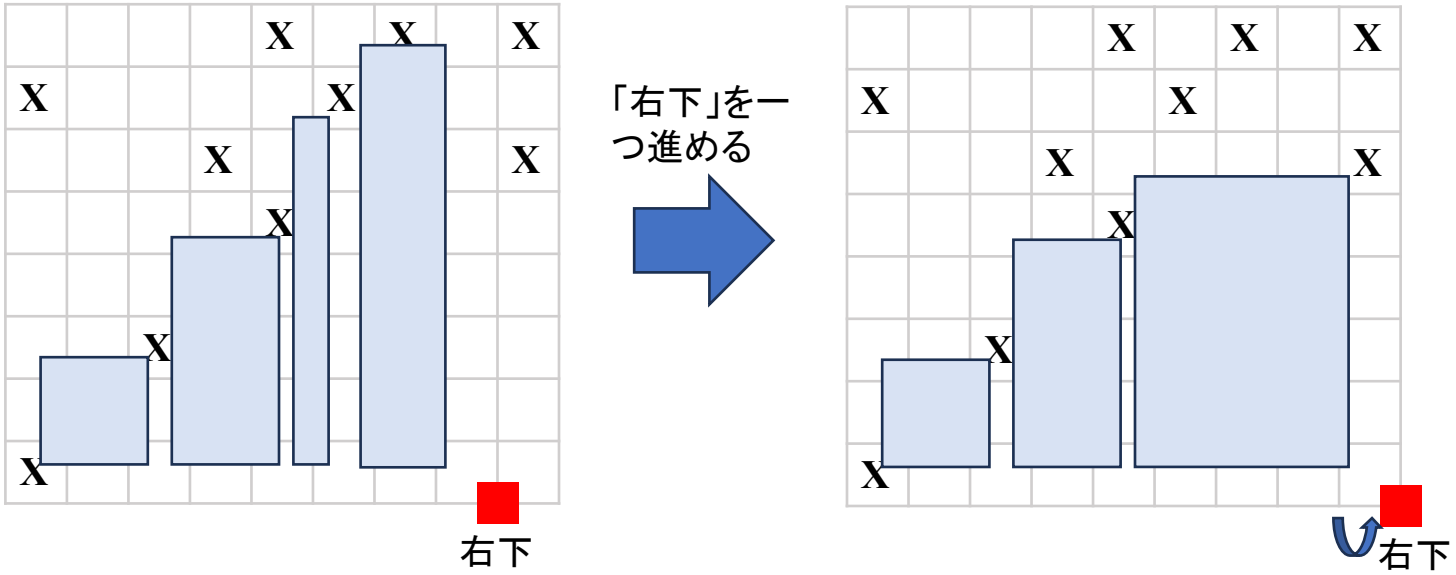
記号マスを含まない領域



長方形の右下(C)■を固定したときに、
左上(A)●として選べる範囲はこんな感じ

すべての格子点について、そこが右下になった場合の長方形の情報を求めたい

記号マスを含まない領域の管理



右下を決めたとき、左上として選べる領域の情報はstackで管理できる
さらに、長方形の右下(C)を固定したときに、長方形の個数、左上の数の和、左下の数の和を逐次的に計算できる（2次元累積和）

一部の項を求める

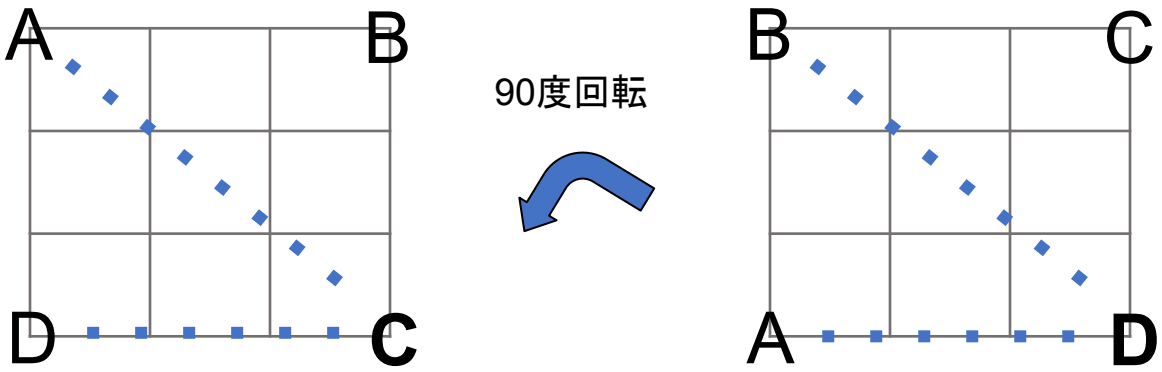
すべての頂点に対して、
その頂点が右下(C)になるような
・長方形の個数
・左上の数(A)の和
・左下の数(D)の和
を逐次的にO(HW)で計算できる

O(HW)で　Σ C^2, Σ AC, Σ DC　が計算可能

Score　= Σ(A-B-C+D)^2
　　= Σ (A^2+B^2+C^2+D^2+2AC+2BD-2AB-2BC-2CD-2AD)

10項のうち3項の計算ができたので、
残りの7項もおんなじ感じで頑張る

実装の工夫



盤面を90度回転すると

Σ C^2
Σ AC　と全く同じ処理で
Σ DC

Σ D^2
Σ BD　が計算できる
Σ AD

実装の工夫

Score= Σ(A-B+C-D)^2

= Σ C^2-2DC+AC ┐ 90度回転

+Σ D^2-2AD+BD ┐ 90度回転

+Σ A^2-2BA+CA ┐ 90度回転

+Σ B^2-2CB+DB

Σ C^2-2DC+AC を求めるのを
盤面を回転しながら4回繰り返して合計すればよい

統計情報

- Acceptances
  - 0 team
- First Acceptance
  - なし

G: Convex Polygon MST

原案：tatyam
問題文：tatyam
データセット：tatyam
解答：beet(誤解法), hos(愚直解), tatyam

問題概要

- N 頂点の凸多角形が与えられる. この N 頂点のユークリッド最大全域木を求めよ.
- $N \leq 120{,}000$

愚直解法

- クラスカル法 : O(N^2 log N)
- プリム法 : O(N^2)

➔ 調べる辺の数を減らさないといけない！

解法

- ブルーフカ法に着目
  - 頂点集合 X について,「X 内と X 外を結ぶ辺のうち最も長いもの」は必ず使われるので,
  - 辺がない状態から始め,「各連結成分に対してこれを求めて採用する」操作を O(log N) 回行って全域木にする.
- つまり, 各頂点 v について,「v と連結でない頂点のうち, v から最も遠い頂点」が求まれば良い.

解法

- 各頂点 v について,「**v と連結でない頂点のうち**, v から最も遠い頂点」が求まれば良い.

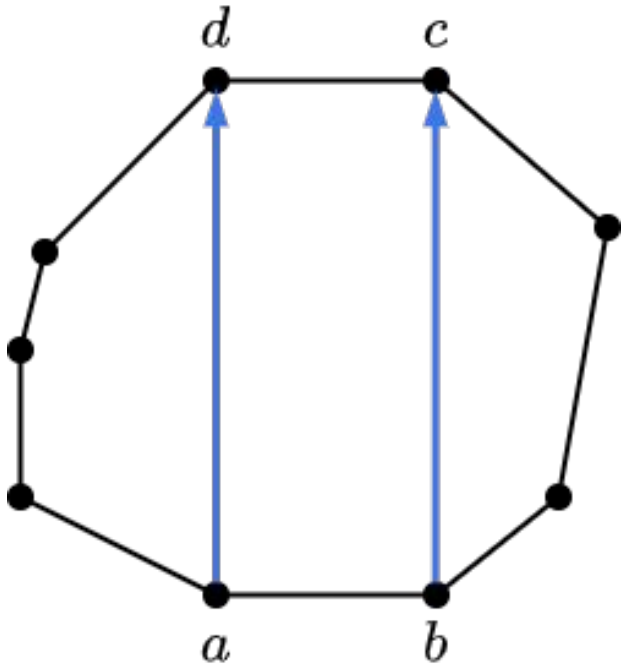
→ **この部分** が面倒なので一旦忘れてみよう

解法

- **各頂点 v について,「v から最も遠い頂点」を求める.**
- どうやって？
- 凸多角形の構造を使ってみよう

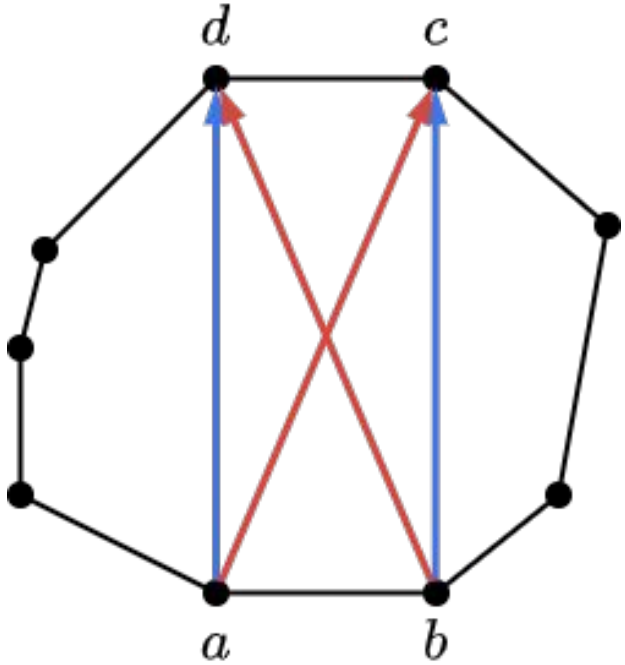
凸多角形の構造

- a の最遠点が d, b の最遠点が c のように, 最遠点への辺が並行することがない



凸多角形の構造

- a の最遠点が d, b の最遠点が c のように, 最遠点への辺が並行することがない
- クロスさせたほうが, 2 本の合計が長くなるため



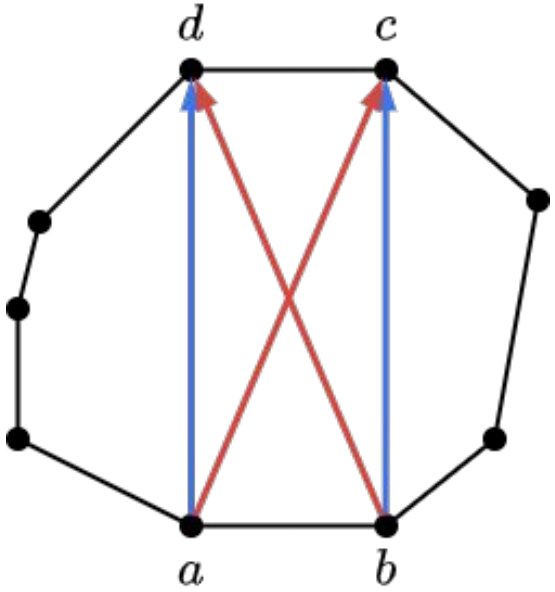
凸多角形の構造

- 反時計回りに 4 頂点 a, b, c, d を取ると,

  dist(a, c) + dist(b, d) ≥ dist(a, d) + dist(b, c)

- これは... Monge ！

最大化の文脈なので, 不等号が逆であることに注意



凸多角形の構造・Monge

- 距離行列の下三角を右にずらし, 残りを -∞ で埋めた行列が Monge

0	3	5	4
3	0	4	5
5	4	0	3
4	5	3	0

0	3	5	4	0			-∞
	0	4	5	3	0		
		0	3	5	4	0	
-∞			0	4	5	4	0

凸多角形の構造・Monge

- 距離行列の下三角を右にずらし, 残りを -∞ で埋めた行列が Monge

➔ monotone minima や SMAWK algorithm で各頂点からの最遠点を求められる.

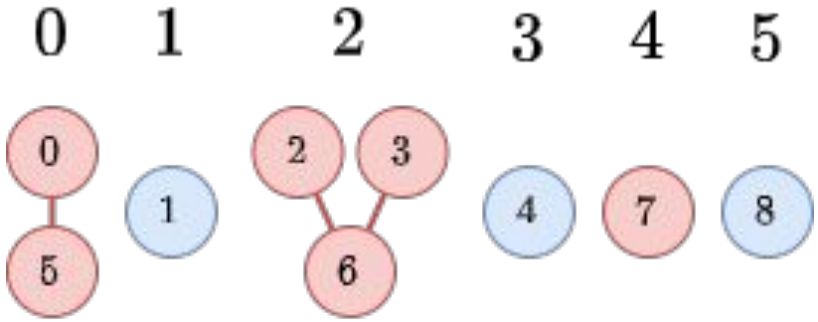
思い出してみよう

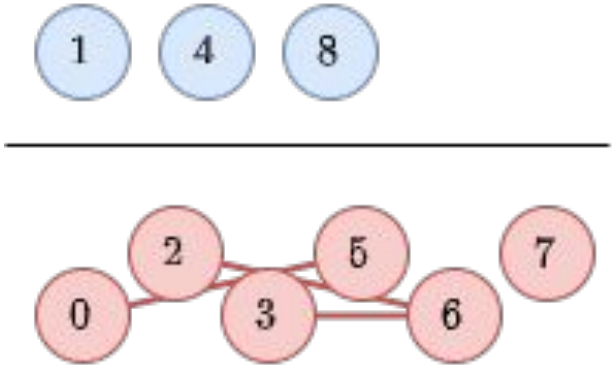
- 各頂点 v について,「**v と連結でない頂点のうち**, v から最も遠い頂点」が求まれば良い.
- 距離行列から, その頂点と連結な列を取り除ければ良いんだけど...

0	3	-∞	4	0			-∞
	0	4	-∞	3	0		
		0	3	-∞	4	0	
-∞			0	4	-∞	4	0

解法

- 各連結成分に 0 から番号をつけ, i = 0, 1, ... について, i bit 目が 0 である連結成分に属する頂点 X とそれ以外の頂点 Y に分ける





解法

- 各連結成分に 0 から番号をつけ, i = 0, 1, ... について, i bit 目が 0 である連結成分に属する頂点 X とそれ以外の頂点 Y に分ける
- X を行に, Y を列にした距離行列も Monge なので, 同様にして X の各頂点から Y のいずれかへの最長距離が求められる
- Y から X への最長距離も求め, これを各 bit についてやれば,「連結でない頂点のうち最も遠い頂点」が求まった

まとめ

- ブルーフ力法: 各連結成分の最遠点を求めることを O(log N) 回
- 各連結成分の最遠点を求める: 連結成分を 2 分することを O(log N) 回やれば, 各頂点の最遠点を求めることに帰着
- 各頂点の最遠点を求める: SMAWK algorithm で O(N)

→ O(N (log N)^2) で解けた！

tatyam「もっと速くなるんだろな〜」

論文、ありました

- Computing euclidean maximum spanning trees

  Clyde Monma, Michael Paterson, Subhash Suri, Frances Yao
  Algorithmica volume 5, pages 407–419 (1990)

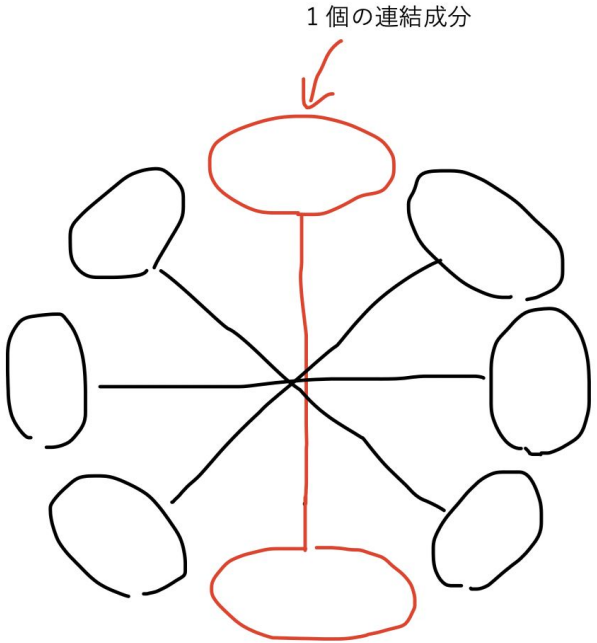
- O(N) になります

最遠点に辺を張ったときの形を観察

- ある範囲と, その対面にあるある範囲が 1 個の連結成分というような形で 1 周している

証明: 有向辺をたどっていくと辺の長さが単調増加であることに注目

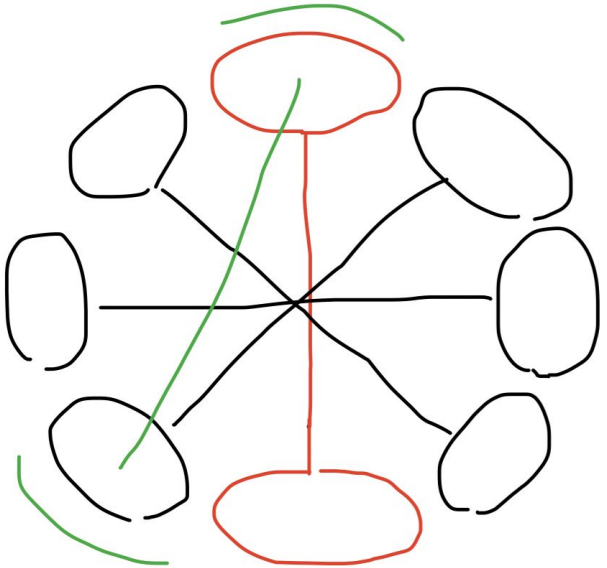
Monge の不等式とかを賢く使う



最遠点に辺を張ったときの形を観察

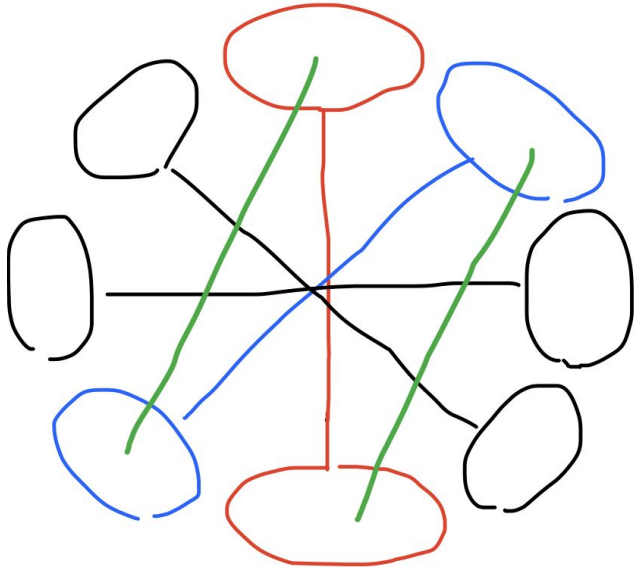
- したがって, 対面から 1 個ずれた部分への辺しか考慮する必要がない

証明: 気合い場合分け + 幾何



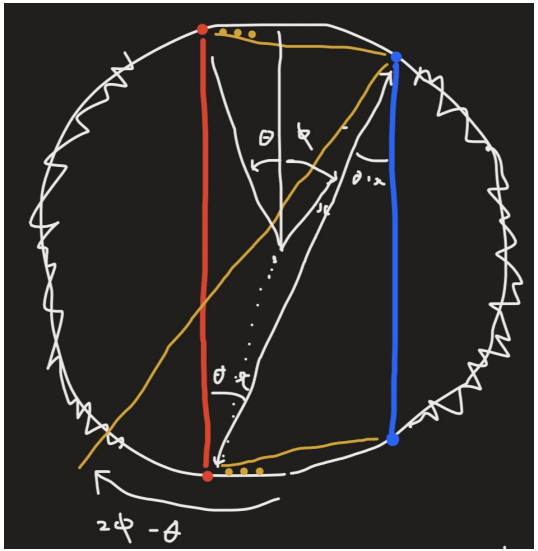
最遠点に辺を張ったときの形を観察

- すべての隣り合う連結成分について, それらの間の最遠点対を求める: SMAWK algorithm で O(N)
- 求めたものを全て採用すると N 辺になるので, 最も短い 1 本を捨てて完成



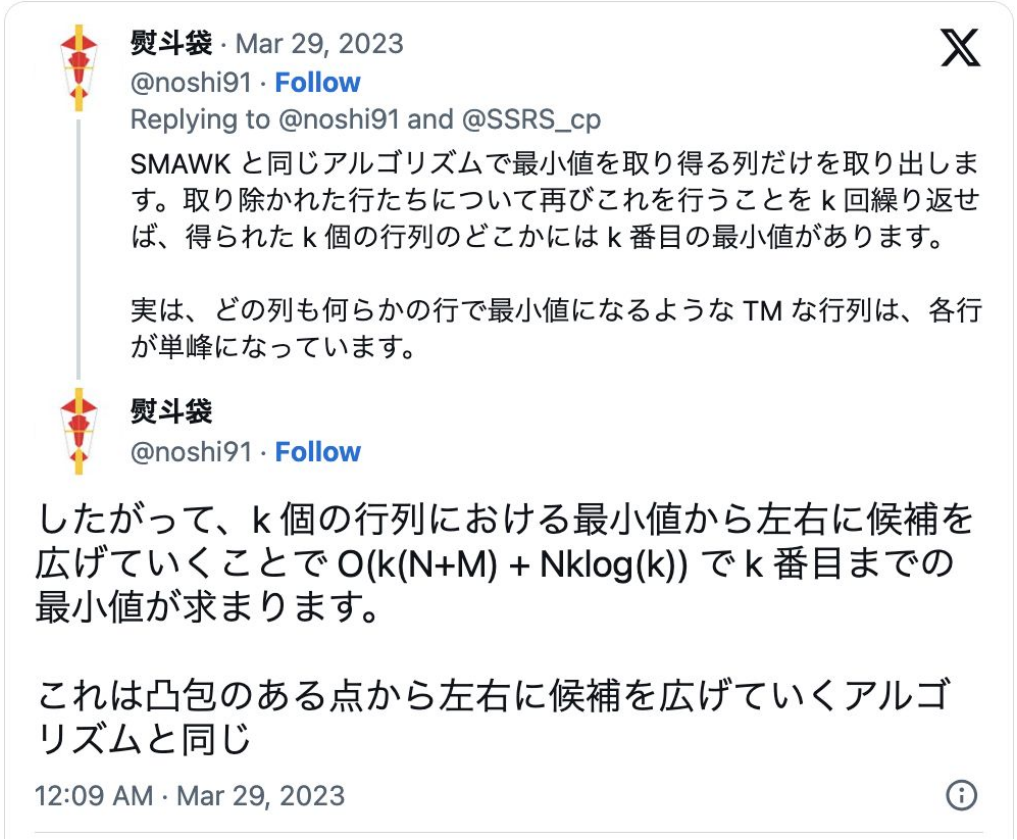
強かった嘘解法

- 各頂点から最遠点を求める. 最遠点の周囲 K 点への辺を候補としてクラスカル法を行う.
- ARC を横目にがんばると落とせる 😭



強かった嘘解法

- これは落とせなかった



熨斗袋 · Mar 29, 2023
@noshi91 · **Follow**
Replying to @noshi91 and @SSRS_cp

SMAWK と同じアルゴリズムで最小値を取り得る列だけを取り出します。取り除かれた行たちについて再びこれを行うことを k 回繰り返せば、得られた k 個の行列のどこかには k 番目の最小値があります。

実は、どの列も何らかの行で最小値になるような TM な行列は、各行が単峰になっています。

熨斗袋
@noshi91 · **Follow**

したがって、k 個の行列における最小値から左右に候補を広げていくことで O(k(N+M) + Nklog(k)) で k 番目までの最小値が求まります。

これは凸包のある点から左右に候補を広げていくアルゴリズムと同じ

12:09 AM · Mar 29, 2023

統計情報

- Acceptances
  - 0 team
- First Acceptance
  - なし

H: LCP Queries

原案 : mtsd
問題文 : climpet
データセット : mtsd
解答 : beet, climpet, hos, kotatsugame, mtsd
解説 : mtsd

問題概要

- 英子文字列集合 S = {S_1, ..., S_n}, T = {T_1, ..., T_m} が与えられる
- 以下のクエリに Q 回答えよ
  - 長さ K の数列 a_1, ... , a_k (1 <= a_i <= m) が与えられる
  - 文字列 X を T の a_1 番目の文字列 + ... + T の a_k 番目の文字列と定義する
  - 「S_iと X の最長共通接頭辞の長さ」の総和を返答する
- 制約
  - S に含まれる文字列の長さの総和<= 200,000
  - T に含まれる文字列の長さの総和<= 200,000
  - クエリの K の総和 <= 200,000

想定解1: トライ木 + 文字列のハッシュ

- 前処理:
  - S に対応するトライ木を作り、各頂点に対して以下の情報を計算する
    - その頂点に対応する文字列のハッシュ
    - その頂点に対応する文字列がクエリに来た時の答え
  - 文字列のハッシュ -> トライ木の頂点 となる map を作成する
- クエリの計算
  - 二分探索でクエリの文字列の prefix でトライ木に含まれる最長の文字列を求める
  - トライ木の頂点に記録している答えを参照する
- 計算量

  O(sum |S_i| + sum |T_i| + |S| log |S| + (sum K) log(max |T_i|) log |S|)

  (※2項目の log |S| は接尾辞配列などを使えば消せる)

想定解2: パトリシア木 + 文字列一致判定

- S に対応するパトリシア木の深さが sqrt(sum |S_i|) であることを利用する
  - パトリシア木(Patricia Trie)
    - ざっくり言うと、トライ木を経路圧縮したもの
    - トライ木では辺が 1 文字に対応していたが、パトリシア木では辺は文字列に対応する
- 前処理
  - S に対応するパトリシア木を作り、各頂点に対して以下の情報を計算する
    - その頂点に対応する文字列がクエリに来た時の答え
    - その頂点に対応する文字列を含む S_i の個数
  - S と T の部分文字列の一致比較が可能なデータ構造を作成する
    - ハッシュや接尾辞配列
- クエリ
  - クエリ文字列について、パトリシア木を順に下り、クエリ文字列がパトリシア木のどこに対応するかを計算する（辺上の可能性もあることに注意）
- 計算量

  O(sum |S_i| + sum |T_i| + sum K + Q sqrt(sum |S_i|))

想定解3: 接尾辞配列＋クエリ先読み

- 各クエリ文字列について以下の値を求めておく
  - ソートされたS に対して辞書順でどこの間に入るか
  - その際の前後の文字列とのLCP の値
    - A = (S_1)$...$(S_n)(T_1)...$(T_n) （$ は区切りに対応する文字）
      という文字列に対する接尾辞配列と LCP 配列を利用する
- S とクエリ文字列はソートされた状態で管理し、昇順・降順それぞれの方向に計算を行う
  - ソートされたクエリ文字列に対して、今見ているS_i との LCP の値の列を stack で管理する
  - 各クエリ文字列の LCP の総和を segment tree などで管理し、LCP の値が減少する際に（複数のクエリ文字列に対して同時に）LCP の総和の加算処理を行う
- 計算量
  O(sum |S_i| + sum |T_i| + sum K + |Q| log |Q|)

統計情報

- Acceptances
  - 3 + 3 teams
- First Acceptance
  - SPJ (145 min)

I: Best parentheses

原案 : ei1333
問題文 : climpet
データセット : ei1333
解答 : beet, climpet, ei1333, hos, kotatsugame

問題概要

- 括弧列 S = (s_1, s_2, ..., s_n) と数列 C = (c_1, c_2, ..., c_n) が与えられる
- 以下を満たすように長さ 0 以上の数列 T = (t_1, t_2, ..., t_k) をつくる
  - 1 ≦ t_1 < t_2 < ... < t_k ≦ n
  - s_{t_1}, s_{t_2}, ..., s_{t_k} を連結した文字列はバランスがとれた括弧列
- sum_{i=1}^k c_{t_i} の最大値を求めよ

- 制約
  - 1 ≦ n ≦ 300 000
  - |c_i| ≦ 10^9

解法

- 文字列を左から見る
- Q を 1 つの ‘(‘ の閉じるためにかかる損益の集合 とする
- s_i が ‘(‘ のとき
  - c_i を Q に追加する
- s_i が ‘)’ のとき
  - (Q の最大値 + c_i) > 0 のとき、答えにこれを加算し
    -c_i を Q に追加する
    - i 番目の ‘)’ で閉じるのをやめて、後から出現する ‘)’ で閉じることができる
- Q を priority_queue で持つことにより O(n log n) で解ける

統計情報

- Acceptances
  - 13 + 1 teams
- First Acceptance
  - Speed Star (19 min)

J: Edit Distance on Table

原案 : rika0384
問題文 : blue_jam
データセット : tsutaj
解答 : beet, hos, kotatsugame, smiken, tsutaj
解説 : tsutaj

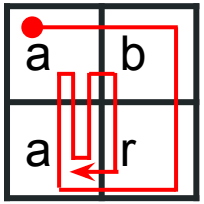
問題概要

- H 行 W 列のテーブルがあり、各セルには文字が 1 つ書かれている
- あるセルから始め、上下左右に隣接する場所に移動を繰り返したときに通過する文字を順番に結合してできる文字列を S とする
- T が入力で与えられる
- S と T の編集距離の最小値を求めてください
- 制約
  - 2 <= H, W <= 100
  - |T| <= 2,000

サンプル

- H = 2, W = 2, T = "abracadabra"
  - S = **"abraaaabra"** で編集距離 2 を達成、それ未満はできないため答えは 2

a	b
a	r



S = abracaaabra

d

T = abracadabra

編集距離の DP

2 つの文字列 S, T の編集距離を求めるのは、以下のような DP でできる

削除　　追加　　置換

- dp[i][j] ← min(dp[i-1][j] + 1, dp[i][j-1] + 1, dp[i-1][j-1] + c)
  - c はコスト。s_i と t_j が一致していたら 0、そうでなければ 1
- これと同じアイデアをグリッドに適用できればよい

解法

- dp[i][r][c] := マス (r, c) にいて、T の i 文字目まで見たときの編集距離の最小値
- i の昇順にこれを更新していけばよい
  - 編集距離の「追加」「置換」の処理は割と素直に書ける
- 注意：編集距離の「削除」の処理では、i は変わらず (r, c) の部分が変わる
  - dp[i][r][c] の値を、**上下左右に隣接しているdp[i][r'][c'] へ伝搬させる**点に注意
  - これをやるには BFS が必要（単純なループでは書けないはず）

- 計算量 O(|T| H W)
  - 定数倍が重く、log をどこかでつけてしまうと落ちるかも

統計情報

- Acceptances
  - 16 + 1 teams
- First Acceptance
  - Speed Star (25 min)

K: Odd trip plans

原案 : riantkb
問題文 : climpet
データセット : riantkb
解答 : hos, kotatsugame, riantkb

問題概要

- N 頂点 M 辺の無向グラフが与えられる
- 以下の 2 種類のクエリが合計 Q 個与えられるので処理しろ
  - 1 u v
    - (u, v) が存在するかどうかを反転する
  - 2 u v
    - u から v への walk で、全ての頂点に奇数回訪れるものが存在するか判定

- N, M, Q <= 10^5

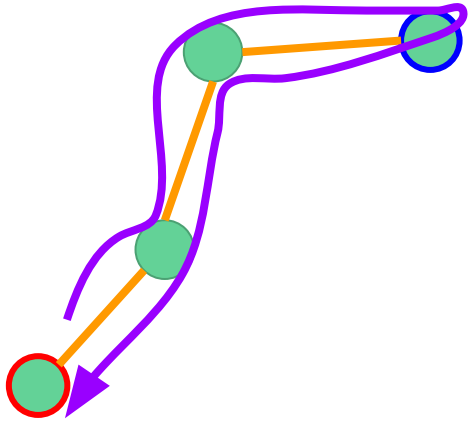
考察

- ひとまず判定クエリ 1 個を解くことを考える
- 自明に、全体が連結でないと不可能
  - 以下は全体が連結であるとする
- 適当な walk を 1 つとった後、偶数回の頂点を奇数回に変えていけないか？

考察

- **今いる頂点**（適当にとった walk の終点）から**ある頂点**まで
  行って帰ってくると、**今いる頂点**と**ある頂点**の偶奇だけ反転する

- 今いる頂点を S としたとき、
  S → A → S → B → S と移動することで
  A と B の 2 頂点のみの偶奇が反転する
  - A = S のときも成り立つ



- よって、少なくとも偶数回の頂点が偶数個であるときは条件が満たせる

考察

- 偶数回の頂点が偶数個である u から v の walk が作れますか？
- 以下では、「walk の長さ」を「頂点列の長さ」で定義します
  - 通った辺の本数とは偶奇が反転するので注意

- 補題：u から v の長さ k の walk について、
  「奇数回訪れた頂点の個数の偶奇」と「k の偶奇」は一致する
  - 各頂点の訪問回数の総和がk と一致するので、場合分けすれば証明できる

- よって、N の偶奇によって条件が変わる
  - N が偶数のとき：偶数長の長さのwalk が作れれば可能
  - N が奇数のとき：奇数長の長さのwalk が作れれば可能

考察

- 偶数または奇数の walk が作れるか？
- u から v の最短路の長さの偶奇がそもそも欲しいものであったら問題ない
- そうでなければ、偶奇が反転できる必要がある
  - これは、奇閉路があるかどうか（二部グラフかどうか）と同値
- 以上をまとめると
  - N が偶数のとき：奇閉路があるまたはu, v が二部グラフで異なる側のときに可能
  - N が奇数のとき：奇閉路があるまたはu, v の二部グラフで同じ側のときに可能

考察

- 残りは全て不可能か？

- 二部グラフなので、walk の偶奇を変えられない
- さっき使った「補題：u から v の長さ k の walk について、
  「奇数回訪れた頂点の個数の偶奇」と「k の偶奇」は一致する」により、
  奇数回訪れた頂点の個数の偶奇も変えられない

- つまり不可能であるとわかる

考察

- これでクエリ 1 つに答えることができた
  - 具体的には N の偶奇、二部グラフかどうか、二部グラフのときu, v が同じ側に属しているかどうかがわかればYesNo のどちらであるかがわかる

- 辺の削除を無視すると、例えば以下をするとできる
  - 頂点数 2N の union-find を用意する
  - 辺 (u, v) の追加については、(2u, 2v+1) と (2u+1, 2v) をそれぞれ union する
  - (u, v) の判定クエリについては以下を調べればよい
    - 連結成分数が 1, 2, 3以上 のどれか
    - (0, 1) が連結かどうか
      - 連結成分数が 2 だけど全体連結でない場合を判定するため
    - (2u, 2v) が連結かどうか

考察

- どうすれば辺が削除できる？

- 以下の 2 つの方針がある
  - 1. offline dynamic connectivity
  - 2. クエリ平方分割

  - 類題：Graph Construction（JAG 夏合宿 2010 : day3D）

700	Graph Construction (☆☆☆☆)	夏合宿 2010:day3D	92

考察

- offline dynamic connectivity
  - クエリ先読みができるときの辺の追加・削除・連結判定クエリは O(log^2 N) とかで実現できる
  - offline dynamic connectivity でググると多分たくさんヒットします

- クエリ平方分割
  - B 個のクエリに対し、その中で変更のある辺は高々O(B) 本
    - 他の頂点を圧縮してしまえば、B 頂点 B 辺の追加削除になって、毎回愚直にやっても O(B α(B))
  - B = sqrt(Q) くらいにすると計算量は O(Q sqrt(Q) α(Q)) とか
    - 実際には B=1000 くらいにするのが速いです（バケットサイズガチャはしよう）

解法

- offline dynamic connectivity またはクエリ平方分割などで、
  辺の追加・削除・連結判定を高速に行う

- その上で、N の偶奇に注意しながら YesNo 判定をする
  - 連結成分数 2 だからといって全体連結かつ二部グラフとは限らないことに注意

統計情報

- Acceptances
  - 4 + 1 teams
- First Acceptance
  - Speed Star (72 min)