A: Stop & Go

原案 : darsein
問題文 : darsein
データセット : TumoiYorozu
解答 : darsein, TumoiYorozu, fuppy, hos, smiken

問題概要

- N 個の信号機が、長さ L の道路にある。
- 信号機 i は、x_i の位置にあり、最初はすべて青である。
- g_i 秒後に 赤に切り替わり、その r_i 秒後に青に戻る周期性がある。
- 車は 1 秒に付き 1 進み、信号が青のときのみ信号を通過でき、赤や赤になった瞬間は通過できない。
- 道路を通過し切るのにかかる時間を求めよ。

解法

- 愚直にシミュレーション。
- x_i で信号をソートし、mod (g_i + r_i) して、赤の時間なら青になるまで待つ
- 赤になった瞬間は通れないことに注意
- 答えが int に収まらない可能性に注意

統計情報

- Acceptances
  - 42 teams
- First Acceptance
  - NM17 (4 min, guest)
  - Time Manipulators (5 min)

B: 1-Player Concentration

原案 : darsein
問題文 : darsein
データセット : TumoiYorozu
解答 : darsein, TumoiYorozu, hos, smiken

問題概要

- 一人で神経衰弱をする。
  カードを右のようにめくっていく。

- 開いたときに、過去に開いたことのある数字であればそれを取っていく

- すべて取るのに何回かかるか

1	2	3
6	5	4
7	8	9
12	11	10

図: カードを開く順番

解法

- 愚直にシミュレーション。

- ハッシュやbool配列などを用い、それぞれの数が既に出たかどうかを管理すればよい。どこに出たかは関係ないので、その情報を覚える必要はない。同じカードは高々2回しかめくられないので O(HW)。

- 開く順番が面倒なので、受け取った入力をめくる順に直した1次元配列を作り、それに対して先頭から見ていくと楽かも。

統計情報

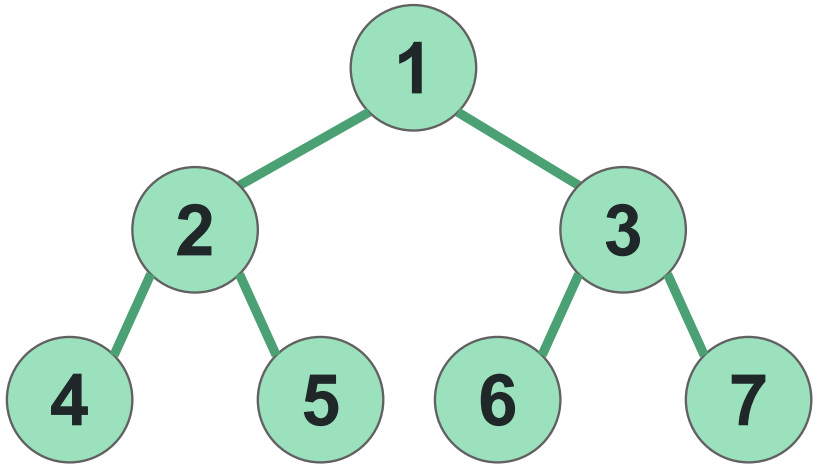
- Acceptances
  - 42 + ? teams
- First Acceptance
  - Maximum-goodlife9 (11 min)

L: Tree Automorphism

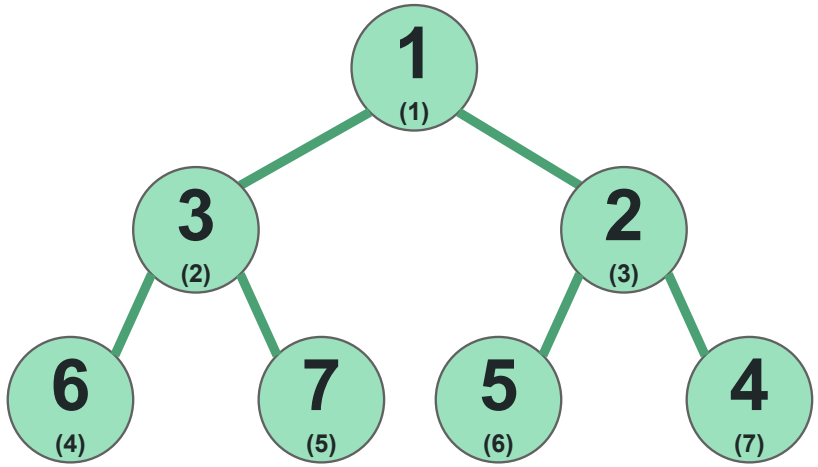
原案 : climpet
問題文 : tsutaj
データセット : riantkb
解答 : beet, hos, riantkb, smiken, tsutaj

問題概要

- N 頂点の木 T が与えられる
- 以下を満たす (1, 2, ..., N) の順列 P = (p_1, p_2, ..., p_N) は何通りあるか？
  - 辺 {u, v} が T に存在するならば、辺 {p_u, p_v} も T に存在する
- 例



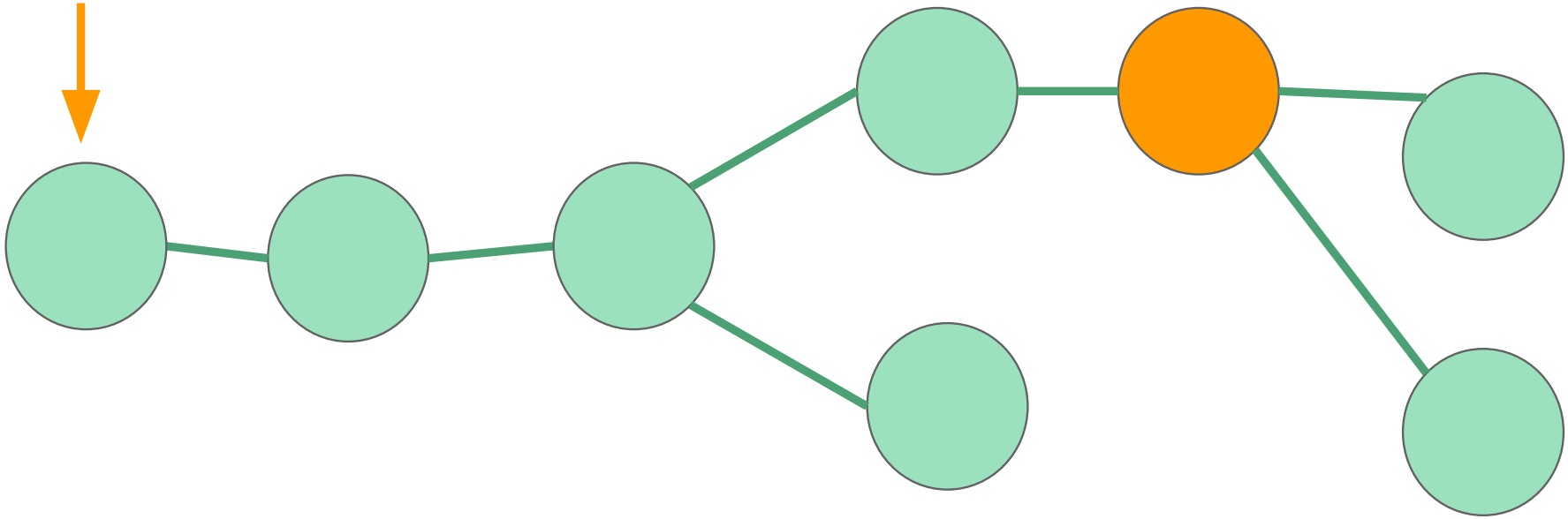
入力 T



条件を満たす順列の一例
(括弧内の整数は順列の添字)

前提知識

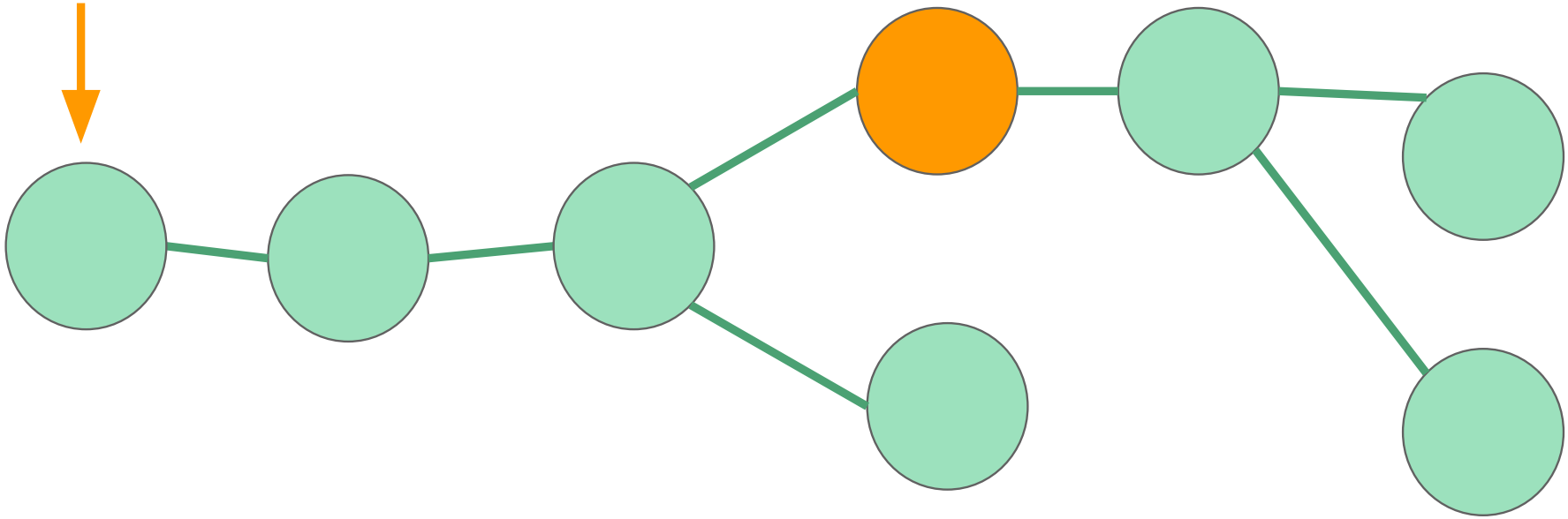
- 木の**中心**: 最も遠い頂点までの距離が最小となる頂点のこと
- 中心は 1 個または 2 個存在する



橙色で塗られた頂点を選んだとき、最も遠い頂点は矢印で示した頂点
距離は 4

前提知識

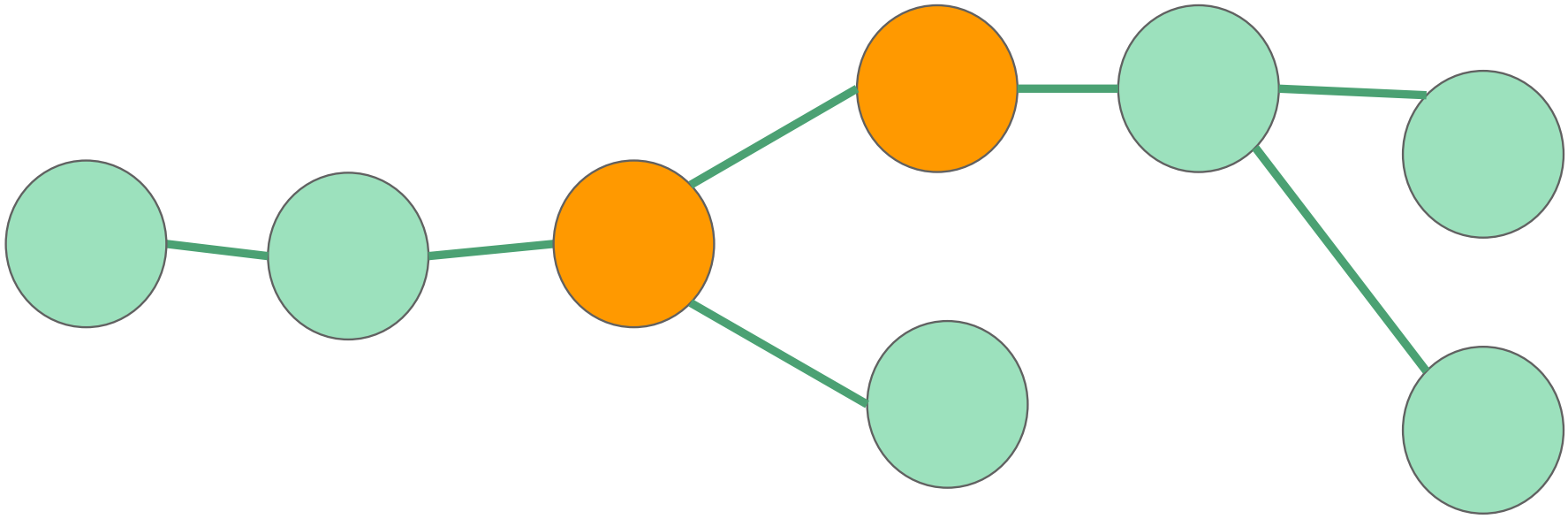
- 木の**中心**: 最も遠い頂点までの距離が最小となる頂点のこと
- 中心は 1 個または 2 個存在する



橙色で塗られた頂点を選んだとき、最も遠い頂点は矢印で示した頂点
距離は 3
最も遠い頂点との距離がこれより小さくなるものは存在しないので、 **この頂点は木の中心**

前提知識

- 木の**中心**: 最も遠い頂点までの距離が最小となる頂点のこと
- 中心は 1 個または 2 個存在する



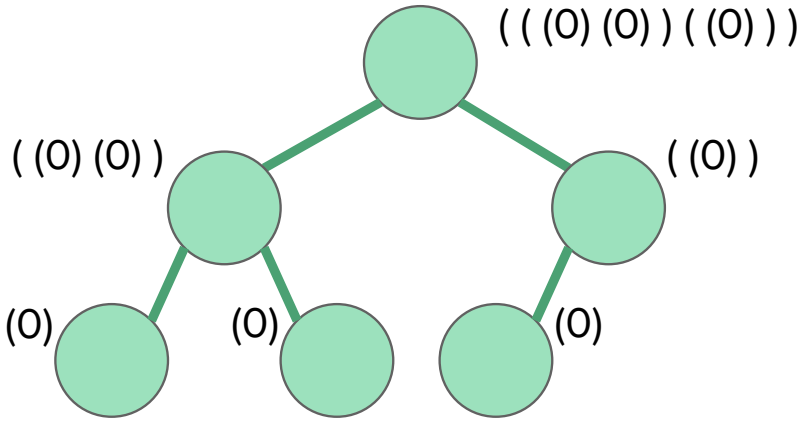
この木は中心を 2 個持つ

解法

- (必須ではないが) 各辺の間に頂点を 1 つ追加しておく
  - これをすることで木の直径が偶数になり、木の中心が一意に定まる
- 木の中心を根とした根付き木を考える
- 木 T に対する答え f(T) は、根の直下に t_1 に同型な部分木が c_1 個、t_2 に同型な部分木が c_2 個, ... あったとすると、以下のように求められる

$$f(T) = f(t_1)^{c_1} \times c_1! \times f(t_2)^{c_2} \times c_2! \times \ldots$$

- 根付き木の同型性判定は AHU algorithm などでできる



統計情報

- Acceptances
  - 15 teams
- First Acceptance
  - onsite: TLE_WARLD (74 min)
  - online: Heno World (99 min)

K: Sort Compressed Strings

原案 : climpet
問題文 : tsutaj
データセット : riantkb
解答 : hos, riantkb, smiken, tsutaj
解説 : riantkb

概要

- 「圧縮された文字列」が N (≦ 50) 個あるので、展開したときの辞書順でソートしてください
  - 同じ文字列に展開される場合は先に入力された方が前に来なければならない（安定ソート）

  - |S_i| ≦ 2000
  - （各文字列の展開後の長さ）≦10^{10}

  - 圧縮された文字列の例：
    - 3(AB)C → ABABABC
    - 2(2(2A)) → AAAAAAAA

解法

- ローリングハッシュを使って「展開された後の文字列の先頭 K 文字が一致しているかどうか」の判定を行うことを考える

- もしそれができれば、二分探索をすると 2 つの文字列が最初にどこで異なるかが分かり、その異なった最初の文字を取ってくれば
  2 つの文字列の比較ができる

解法

- 先頭 K 文字のローリングハッシュはローリングハッシュの値を持ちながら構文解析をすればできる

- 基本的には「いま展開後何文字となるところまで読んだか」を持ちながら構文解析をし、それが K 以下であるならばローリングハッシュに追加する、ということを行っていく
  - M(seq) のような形で途中までのものが必要となるが、
    seq の展開後の長さが分かればx 回繰り返したあと先頭y 文字が必要、
    とわかるので再帰していけばよい

解法

- M 回繰り返しのハッシュの求め方については、
  hsh * (1 + x + x^2 + ... + x^{M-1})
  みたいなのが求まればよく、繰り返し自乗法などで求まる
  - (x^M - 1) / (x - 1) という形にする場合は、逆元を求めるのに余計に O(log MOD) かけると TLE する可能性が高いので注意
    - ハッシュと一緒に base^{-length} を持っておくとその log が落ちる
  - また、方針によっては定数倍が重いことがあり、メモ化などでの高速化が必要な場合がある
    - 構文木の各ノードについてハッシュの結果を持っておいて、
      その部分木全て使う場合はその持っておいた結果を使用する、など
    - また、各文字列について先頭 K 文字の結果を一度計算したらメモする、というのもかなり有効
    - 繰り返しのときの再帰で実装をミスるとその部分が指数になったりもするので注意

- 計算量は O(N max(|S|) log N log L)（L: 展開後の文字列の長さの max）
  - TL は 4 sec ですが、ジャッジ解は 0.1 sec ほどで通っています

ジャッジ解

- hos (C++): 259 lines, 8.0 kB
- riantkb (C++): 243 lines, 7.3 kB
- smiken (C++): 268 lines, 4.5 kB
- tsutaj (C++): 292 lines, 8.8 kB

統計情報

- Acceptances
  - 0 teams
- First Acceptance
  - N/A

E: Sharp 2-SAT

原案 : hos
問題文 : amylase
データセット : climpet
解答 : beet, climpet, fuppy, hos

問題概要

- bool 変数 x[1], ..., x[N] についての M 節の 2-SAT のインスタンスが与えられる

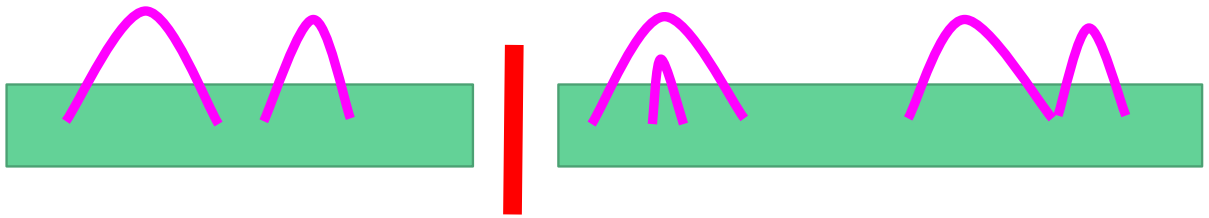
$$(x[13] \lor \neg x[12]) \land (x[6] \lor x[4]) \land (\neg x[6] \lor x[4]) \land (\neg x[10] \lor \neg x[10])$$

- **ただし同じ節に登場する変数の index の差は ≤ 2**
- 各 k = 0, ..., N に対し、充足する割り当てのうちちょうど k 変数を true にするものの個数 mod 998244353 を答えよ

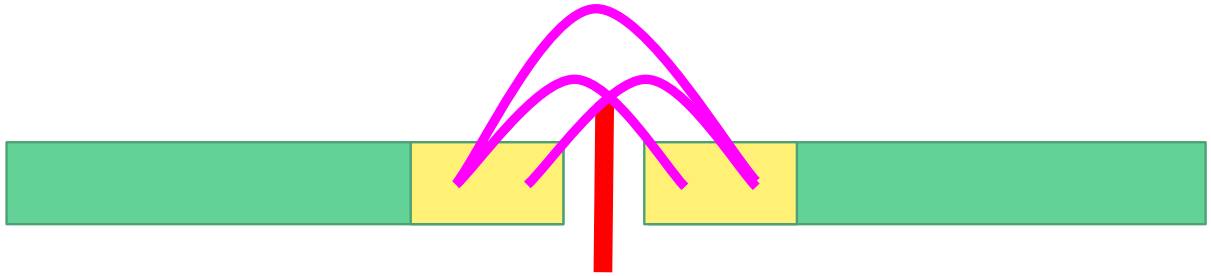
- 制約 : $1 \le N \le 100{,}000$, $1 \le M \le 100{,}000$

考察

- 列を切って条件が左右独立になってたら → **畳み込み**



- なってなくても自分で切る → **分割統治** → 区間の端の情報が要る



解法

- f(l, r, A, k) := 区間 [l, r) に対する割り当てであって、「端の情報」が A で、k 変数が true で、区間内に含まれる条件はすべて満たすものの個数
  - 「端の情報」は左端 2 マスと右端 2 マスに何を割り当てたか → 最大 2^4 通り

- [l, r) を [l, (l+r)/2), [(l+r)/2, r) に分けて再帰して、畳み込みでマージ
  - 左右の「端の情報」の組 $2^4 \times 2^4$ 通り (のうち条件を満たすもの) について畳み込み

- base case: r-l <= 3 あたりで全探索するとよい
- 2-SAT の条件は前処理して変数の index から取得できるようにしておく
- 時間計算量 $\Theta(N (\log N)^2 + M)$

計算量

- $2^4 \times 2^4$ 回 $\Theta((r-l) \log (r-l))$ 時間で畳み込むとおそらく TLE

- 左右の区間について各「端の情報」ごとに FFT しておいて、各点積を足し上げてから戻すと 3×2^4 回の FFT で済む
  - 参考：成分が d 次多項式の $n \times n$ 行列の積も同様にすると $\Theta(n^3 d \log d)$ 時間が $\Theta(n^3 d + n^2 d \log d)$ 時間になる

- 一般に index の差が $\leq K$ だったら $\Theta(2^{3K} N \log N + 2^{2K} N (\log N)^2 + M)$ 時間

統計情報

- Acceptances
  - 1 team
- First Acceptance
  - QWE_titled (228 min)

D: 2-LIS

原案 : tsutaj
問題文 : beet
データセット : tsutaj, riantkb
解答 : beet, hos, riantkb, smiken, tsutaj

問題概要

数列 a から以下の条件を満たすように部分列 b をつくるとき、b の最大の長さを答えよ

- b は a の部分列である（連続とは限らない)
- b_{i}<b_{i+2}

解法

```
for(int j=0;j<n;j++) {
 int res=1;
 for(int v: ord) { // a 昇順
   if(v==j) continue;
   if(v<j) chmax(res,dp[v][j]);
   if(j<v) dp[j][v]=res+1;
 }
}
```

dp[i][j] := 最後に使った要素が a[j], 最後から二番目に使った要素が a[i]

愚直に遷移すると O(N^3)

dp[*][j] から dp[j][*] にまとめて遷移すると O(N^2) で解ける

統計情報

- Acceptances
  - 29 + 2 teams
- First Acceptance
  - Anrirled (32 min)

F: Seimei Handan 999.0

原案 : climpet
問題文 : amylase
データセット : climpet
解答 : climpet, fuppy, hos

問題概要

- 1, ..., L からなる整数列 A, B が与えられる。
- 全単射 $f : [1, L] \to [1, L]$ のうち、次の条件を満たすものは何通りか？
  - 列 $(f(A[0]), f(A[1]), \ldots)$ は B を連続部分列として含む。

- 制約
  - $|A|, |B|, L \leq 3 \times 10^5$

定義 : Parameterized match (p-match)

- 同じ長さの列 X, Y が次の条件を満たすとき、X と Y は parameterized match (p-match) すると定義する。
  - ある全単射 f が存在し、任意の位置 i に対し $f(X[i]) = Y[i]$

考察

- A の連続部分列と B が p-match するとき、それに対応する全単射 f は、「ほぼ」一意に定まる。
  - 正確には、B に一度も出現しない要素についてはどう割り当てても良い。具体的には B に出現する要素が K 種類のとき、ちょうど $(L - K)!$ 通り存在する。
- つまり、次の問題に帰着できる。
  - A の連続部分列のうち、B と p-match するものは何種類あるか？最後に $(L - K)!$ 倍して答えよ。
- 連続部分列の重複除去は、接尾辞配列やローリングハッシュなどを使えばよい。よって、次の問題が解ければよい。
  - A の連続部分列のうち、B と p-match するものの出現位置を全列挙せよ。

考察

列 X に対し、整数列 prev_X を次のように定義する。

- $X[i]$ が初出ならば、$\mathrm{prev}_X[i] = 0$
- $X[i]$ が以前に出現したことがあるならば、その最右位置を j とすると、
  $\mathrm{prev}_X[i] = i - j$

	0	1	2	3	4	5	6	7	8	9
X	10	20	30	20	10	40	20	10	30	50
prev_X	0	0	0	2	4	0	3	3	6	0

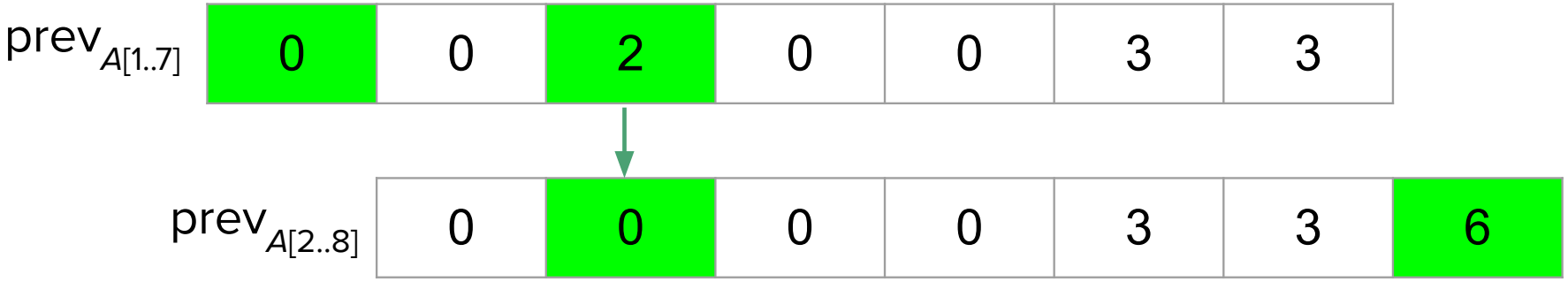
考察

- 列 X, Y が p-match することと、$\mathrm{prev}_X = \mathrm{prev}_Y$ であることは同値である。
- したがって、A の連続部分列 A' のうち、$\mathrm{prev}_{A'} = \mathrm{prev}_B$ であるものを見つけられればよい。
- ここからは、少なくとも二通りの方針がある。
  - ローリングハッシュを使う方法
  - Knuth-Morris-Pratt や Z-algorithm を基にした方法

方針1: ローリングハッシュ

長さ $|B|$ のウィンドウを一要素ずつずらしながら、A の連続部分列 A' に対し $\mathrm{hash}(\mathrm{prev}_{A'})$ を計算していく。中間の要素が 0 に変わる場合があることに注意。

	0	1	2	3	4	5	6	7	8	9
A	10	20	30	20	10	40	20	10	30	50



方針2: KMP や Z-algorithm など

(Knuth-)Morris-Pratt や Z-algorithm において、文字の比較を行っている部分を、prev を用いたものに置き換える。ただし、prev が大きすぎる場合には 0 とみなす必要がある。

例えば、この実装 ($y = A, x = B$ に相当) であれば、
x[i] != y[j] となっている部分を prevX[i] != (prevY[j] <= i ? prevY[j] : 0) のように置き換えるとよい。x[i] != x[j] となっている部分についても同様。

Z-algorithm についても同様に改修できる。

解法まとめ

1. ローリングハッシュ、Knuth-Morris-Pratt, Z-algorithm などを用いて、A の連続部分列 A' のうち $\mathrm{prev}_{A'} = \mathrm{prev}_B$ となるものの出現位置を全列挙する。
2. ローリングハッシュや接尾辞配列などを用いて、それらの連続部分列の重複を除去する。
3. 最後に $(L - K)!$ をかける。ただし K は B に出現する要素の種類数。

計算量は全体で $O(N + M + L)$ 時間。

統計情報

- Acceptances
  - 8 teams
- First Acceptance
  - KOMOREBI (163 min)

H: Sloppy city planning

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

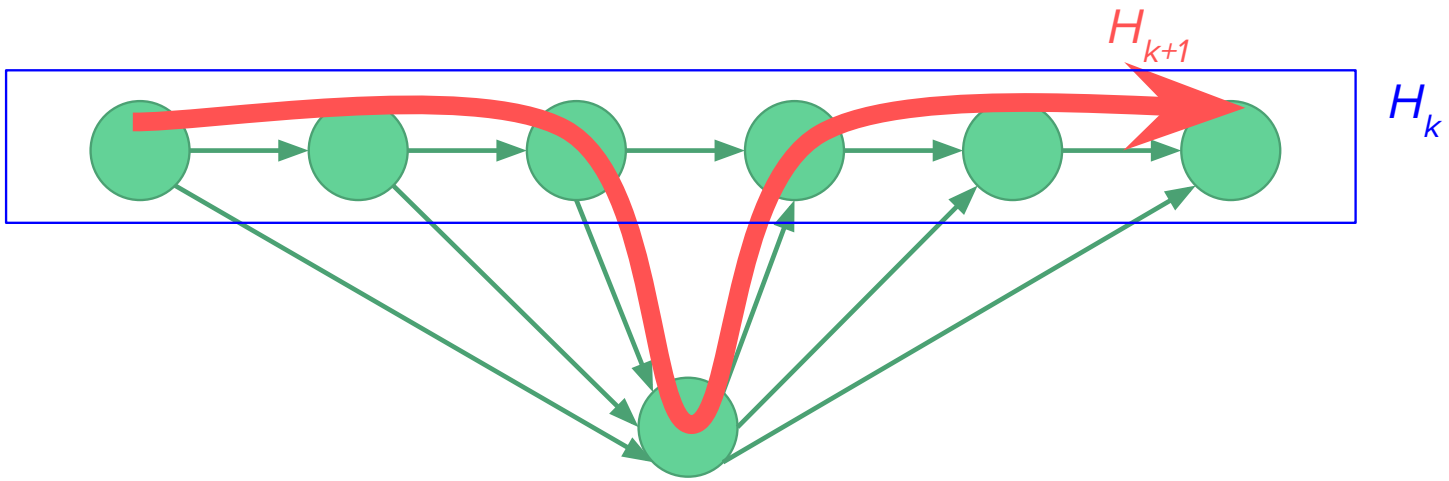
問題概要

- N 頂点の tournament が与えられる。
- 辺ごとに所定のコストを支払うことで、その辺の向きを反転させることができる。
- グラフを強連結にするのに必要なコストの最小値は？

- 制約
  - $3 \leq N \leq 3000$

ハミルトン路

- tournament には常にハミルトン路が必ず存在する。
  - ハミルトン路：すべての頂点をちょうど一度ずつ通る経路
- 証明および構成方針：数学的帰納法。k 頂点の tournament にハミルトン路 H_k が存在すると仮定する。新たに 1 頂点 (とそれに接続する辺) を追加したとき、H_k の先頭・末尾・中間のいずれかにその頂点を追加して、H_{k+1} を構成できる。



考察

- 元のグラフ G のハミルトン路を一つ求め、これを H とする。
- H の先頭および末尾の頂点をそれぞれ X, Y とする。
- X からは任意の頂点に到達可能である。同様に、任意の頂点から Y に到達可能である。
- あとは、Y から X に到達できるようにすれば、任意の頂点から任意の頂点に到達可能になりそう。

考察

- 実際、次のことが証明できる。(証明は後述)
  - Y から X への経路を (元々の辺の向きを一旦無視して) 一つ取り、これを P とする。P 上の各辺について、G と向きが異なる (つまり G 上に辺 $u \to v$ が、P 上に $v \to u$ が存在する) ならば、その辺の向きを反転させる。こうしてできたグラフ G' は強連結である。
- よって、Y から X への最短経路長が答えになる。
  - ただし、順向きに辺をたどるときは辺の長さを0 とし、そうでない場合は反転コストを辺の長さとする。

解法まとめ

1. 元のグラフのハミルトン路を一つ求める。その先頭の頂点を X、末尾の頂点を Y とする。
   a. または、元のグラフを強連結成分分解し、入次数0 の強連結成分に属する任意の頂点を X、出次数 0 の強連結成分に属する任意の頂点を Y としてもよい。
2. Y から X への最短経路長をダイクストラ法で求める。

- どちらも $O(N^2)$ 時間で実装できる。
  - 注意：入力が大きいので、ダイクストラの実装にlog を付けると、もしかしたら通らないかもしれません。

ちなみに

- この問題の答えは、全点対間最短経路長の最大値にも一致する。
- なので、Warshall-Floyd を走らせて最大値を出力すると、正しい答えが得られる。
- ただし、入力が大きいので、$\Theta(N^3)$ 時間の解法は多分通らないと思います。

補遺: パス上の辺を反転させてよいことの証明

補題: Y から X への経路を (元々の辺の向きを一旦無視して) 一つ取り、これを P とする。P 上の各辺について、元のグラフ G と向きが異なる
(つまり G 上に辺 $u \to v$ が、P 上に $v \to u$ が存在する) ならば、その辺の向きを反転させる。こうしてできたグラフ G' は強連結である。

補遺: パス上の辺を反転させてよいことの証明

G' が強連結であるための十分条件として、*G'* 上で次のことが示されればよい。

A) *Y* から *X* に到達可能 (そのように反転させたので、自明)
B) 元のハミルトン路 *H* 上に辺 $u \to v$ が存在するならば、*G'* 上においても *u* から *v* へ到達可能

B について、辺 $u \to v$ が反転されない場合は自明。辺 $u \to v$ が反転される場合を考える。

補遺: パス上の辺を反転させてよいことの証明

元のハミルトン路 H 上の辺 $u \to v$ が反転される場合を考える。

明らかに、元のグラフ G において Y から X へは到達不能であるから、G 上に辺 $X \to Y$ が存在する。さらに、$N \geq 3$ より、辺 $X \to Y$ は H に含まれない。また、この状況下で辺 $X \to Y$ は反転されていない (さもなくば $u \to v$ を反転させる意味がない)。

すなわち、G' 上には辺 $X \to Y$ が存在する。この辺とパス P 上の辺をたどることで、$u \to \ldots$ (P 上の辺) $\ldots \to X \to Y \to \ldots$ (P 上の辺) $\ldots \to v$ という歩道が存在する。

よって、G' 上で u から v への到達可能性は失われない。∎

統計情報

- Acceptances
  - 29 teams
- First Acceptance
  - TLE_WARLD (34 min)

J: Most distant point from the stations

原案 : beet
問題文 : beet
データセット : Darsein
解答 : beet, ei1333, hos, rian

問題概要

[0, W] x [0, H] の長方形領域に N 個の駅がある。i 番目の駅の座標は (x_i, y_i).

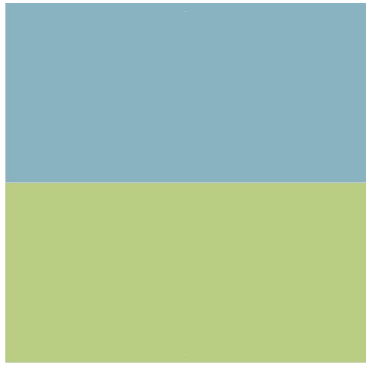
$$\max_{(x,y)\in[0,W]\times[0,H]} \min_{v} (x_v-x)^2+(y_v-y)^2$$ の値を求めよ。(x, y は実数を動く)

元ネタ: Most Distant Point from the Sea (アジア地区予選 2007)

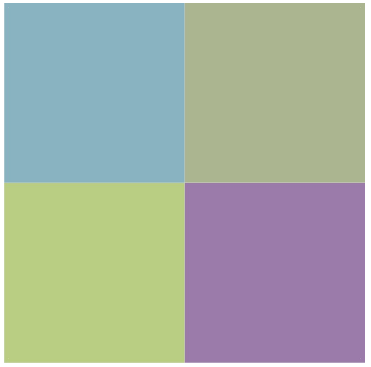
解法

ボロノイ図

2 6 6
3 1
3 5



4 10 10
1 1
1 9
9 1
9 9



9 10 10
1 1
1 5
1 9
5 1
5 5
5 9
9 1
9 5
9 9

ボロノイ図

ボロノイ図（ボロノイず、英: Voronoi diagram）は、ある距離空間上の任意の位置に配置された複数個の点（母点）に対して、同一距離空間上の他の点がどの母点に近いかによって領域分けされた図のことである

- https://ja.wikipedia.org/wiki/%E3%83%9C%E3%83%AD%E3%83%8E%E3%82%A4%E5%9B%B3

実装

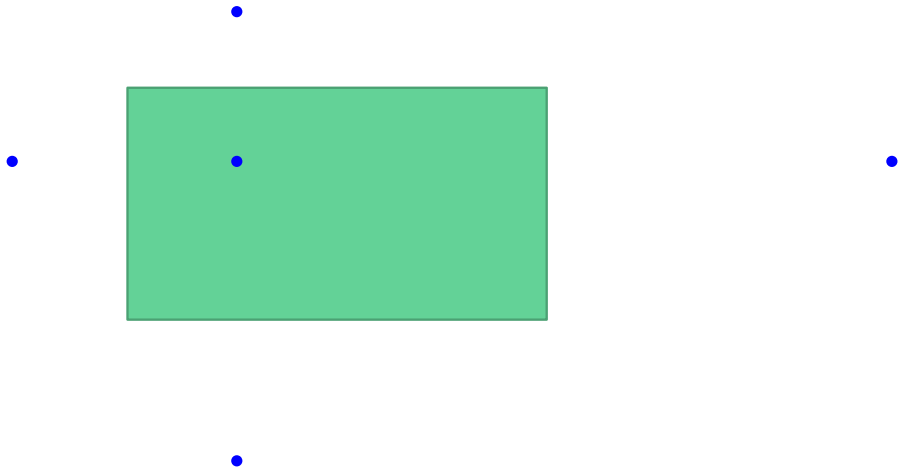
- 三点固定して外心を全て試す - O(N^4)
- N 回 Convex Cut - O(N^3)

———————— TLE の壁 ————————————————

- N 回 half plane intersection - O(N^2 log N)
- fortune algorithm（平面走査）- O(N log N)

Half Plane Intersection

- ある頂点を固定して、他の点との垂直二等分線で囲む
- 偏角ソートして CHT
  - 類題：Magic Triangles (JAG summer 2018 day3 E)
  - 今回の問題設定では共通部分が空にならないため、場合分けが多少減らせる
- 長方形領域パートは、領域の外側に 4 点置くと楽



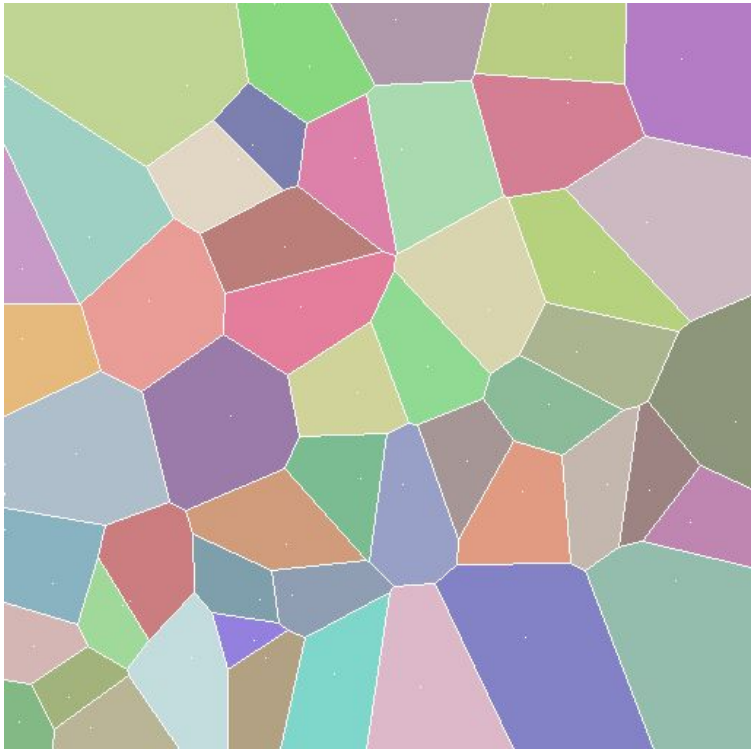
おまけ：fortune algorithm

"voronoi diagram implementation" とかで検索すると
https://github.com/JCash/voronoi が出てきます

実装をサボって平衡二分木の代わりに連結リストを使っているらしく、最悪計算量は不明

https://github.com/JCash/voronoi/issues/48

誰か実装して Library Checker に追加してください

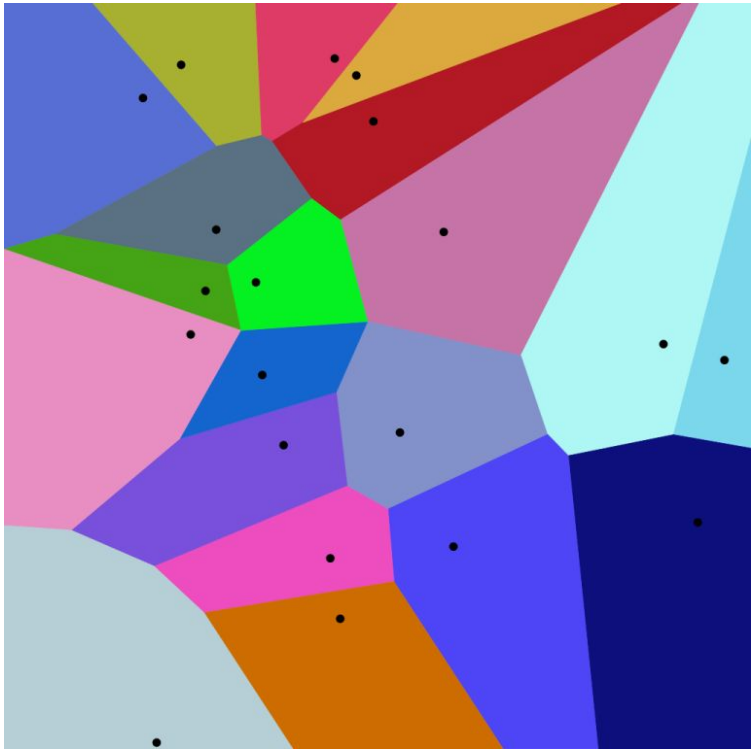


おまけ2: fortune algorithm

"voronoi diagram codeforces" とかで検索すると
https://codeforces.com/topic/86176 が出てきます

*My code doesn't work when two points have the same x coordinate. This is handled simply by rotating all input points by 1 radian and **praying to the geometry gods**.*

誰か実装して Library Checker に追加してください



統計情報

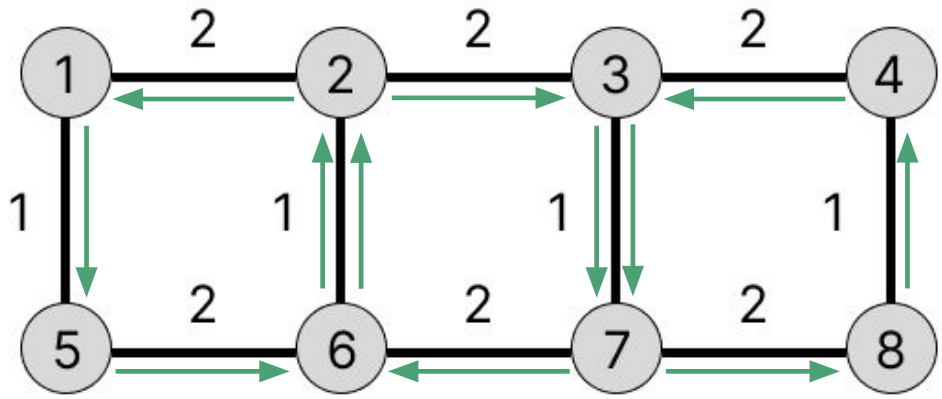
- Acceptances
  - 0 + 0 teams
- First Acceptance
  - なし

C: Track Train Trail

原案 : prime
問題文 : darsein
データセット : riantkb
解答 : darsein, ei1333, hos

問題概要

- 無向グリッドグラフ上で全ての辺を必ず1度以上通るような、コスト最小の閉ウォークを求めよ
- 制約: $2 \leq H, W \leq 100$

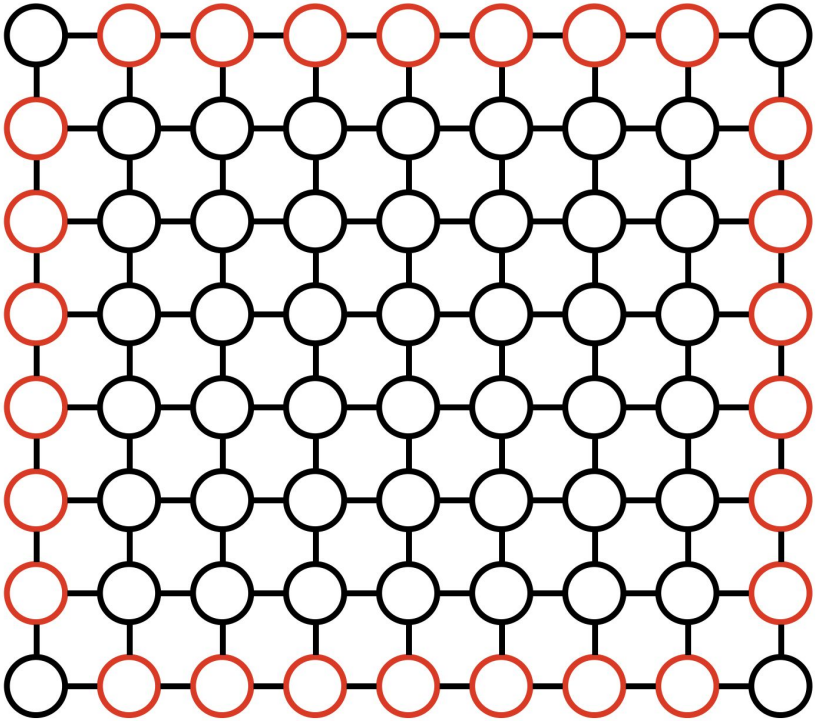


考察

- 「中国人郵便配達問題」のグリッドグラフ版である
- 一般のグラフの場合、$O(V^3)$ で解けることが知られている (V := 頂点数)
  - 元ある辺と同じコストの多重辺を追加する操作のみで、元のグラフをオイラーグラフにすることを考える
  - オイラーグラフである必要十分条件は「全ての頂点の次数が偶数」
  - 奇数の頂点はちょうど偶数個ある(握手補題) ので、どの奇数頂点同士をマッチングするかの問題に帰着できる
  - 全頂点対間最短路 + 最小一般マッチングで $O(V^3)$
- $O(V^3) = O(H^3W^3)$ なので、今回の制約では遅すぎる

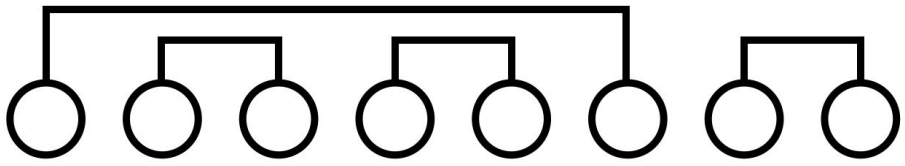
解法

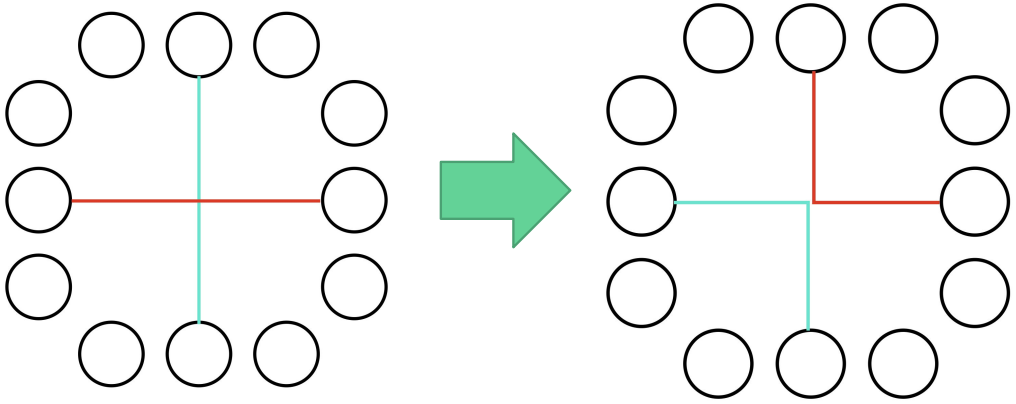
- グリッドグラフの奇数頂点は端にしかなく、2(H-2) + 2(W-2) 個だけ
  - O(H+W) 個の頂点だけでマッチングを考えればよい
- 全頂点対間距離:
  - 対象になる頂点のみを始点にダイクストラ
    O((H+W) HW log HW)
- 最小一般マッチング:
  - $O((H+W)^3)$
- となり、間に合う



解法の工夫

- 一般マッチングを書くのは大変なので工夫して避けたい
- マッチングを図示したとき、合計コストを変えずにパスの交差を減らすことができる
- 交差がない最小マッチング
  - ➔ 頂点を時計周りに列、マッチングの辺を区間と見たとき、交差区間がない
  - ➔ 交差区間のない重み最小ペアリングを求める問題に帰着できる
- 区間DPにより $O(N^3)$ で求められる





解法の工夫2

- 交差がないという条件から、偶数番目と奇数番目の頂点間でのマッチングとしてよい
- 二部グラフの最小重み完全マッチングになる
- ハンガリアン法で$O(N^3)$、Primal-Dualで$O(N^3 \log N)$

統計情報

- Acceptances
  - 20 teams
- First Acceptance
  - Guest: plasma (73 min)
  - Contestants/Onsite: Time Manipulators (82 min)

G: Avoid bombings

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

概要

[1]

[1] https://twitter.com/supermario35th/status/1321588807490105344

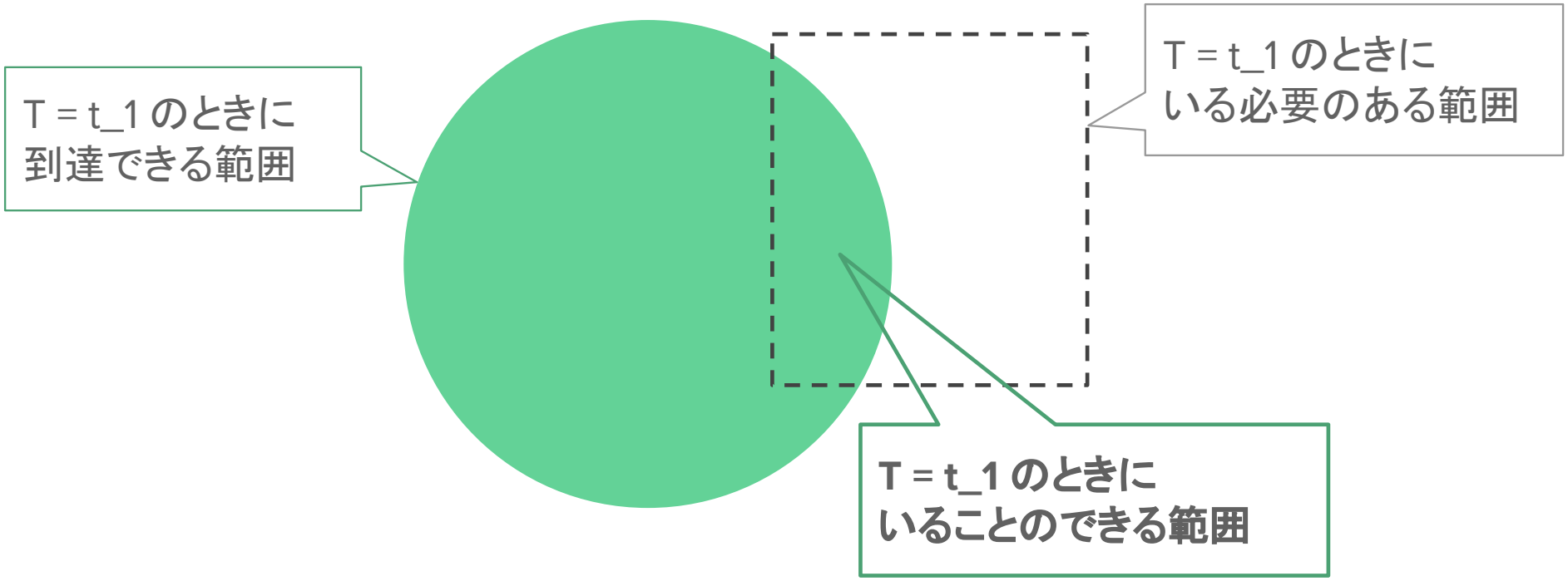
概要

- 時刻 t にこの凸多角形の中にいなければならない、
  という条件が N (≦ 20) 個与えられる
- 全て満たすために必要な速度の最小値は？

  - 各多角形の頂点数 ≦ 20
  - t_i ≦ 100
  - |座標| ≦ 100
  - 一つの入力で最大 20 ケース与えられる

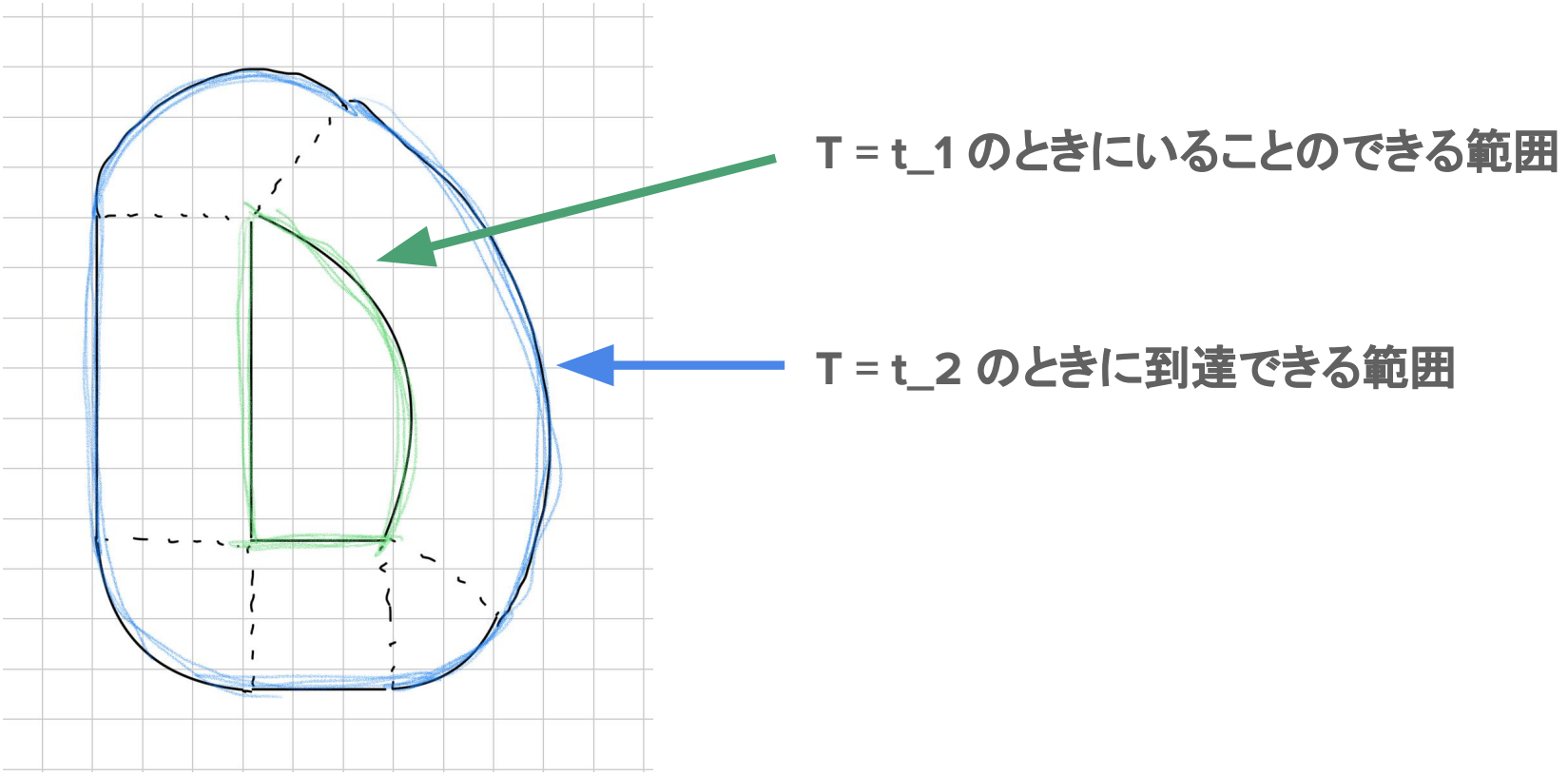
考察

- 速度が v のとき全ての条件を達成できるか？



考察

- 速度が v のとき全ての条件を達成できるか？



解法

- 速度が v のとき全ての条件を達成できるか？
- これは、以下ができればよい
  - 線分および円弧のみからなる凸図形を**r だけ膨らませる**
  - 線分および円弧のみからなる凸図形と凸多角形の**共通部分の計算**
    - 共通部分は線分および円弧のみからなる凸図形になる

解法

- 線分および円弧のみからなる凸図形と多角形の共通部分の求め方

- 以下の点を列挙して反時計回りに並べてつなぐとできる
  - 各図形の頂点のうち、もう片方に完全に含まれるようなもの
  - 図形の辺同士の交点

  - 実際には円弧でつなぐ場合があるので、その情報を持つか
    復元を行う必要がある

- 計算量は 1 ケースあたり O(N^2 M^2 log (max(|X|, |Y|)/eps))

ジャッジ解

- climpet (C++): 255 lines, 5.0 kB
- riantkb (C++): 458 lines, 13.5 kB

統計情報

- Acceptances
  - 0 teams
- First Acceptance
  - N/A

I: N-1 Truths and a Lie

原案 : not
問題文 : darsein
データセット : prime
解答 : beet, hos, prime

問題概要

N人がそれぞれ「山a_iは山b_iよりc_iメートル高い」という主張をする

1人だけ嘘をついているので、だれが嘘をついているか求めよ

2≤N≤2e5, 山の数2≤K≤2e5, a_i ≠ b_i, {a_i, b_i} ≠ {a_j, b_j}, 1≤c_i≤1e9

嘘つきが一意に定まることが保証される

考察

K頂点N辺のグラフとして見る（頂点: 山, 辺: 人）

嘘つきがいない場合、辺を辿ることで、連結成分内の山同士の高さの差がわかる

嘘つきは1人なので、

サイクルが嘘つきの辺を含まない⇔サイクルを一周して高さの差が0

がいえる

方針1

各連結成分の適当な頂点からDFSしながら、根からの高さの差を計算する

後退辺を見たとき、高さが矛盾していないか確かめる

DFS木と後退辺でできるサイクルについて、

矛盾している場合: このサイクル内に嘘つきがいる

矛盾していない場合: このサイクル外に嘘つきがいる

がわかる

この情報を集計することで嘘つきが求められる

方針1

矛盾が報告された後退辺が1本だけ→その後退辺が嘘つき

複数ある場合→DFS木上に嘘つきがいる　次のようにして嘘つきが求まる

矛盾がある場合、サイクルを構成するパスに+1、矛盾しない場合は全体に+1してそのパスに対して-1
していったときに、和が後退辺の数と同じになった辺が答え

パスに対して加算するのは、木に対するimos法を使うことで定数時間でできる

計算量O(N+K)

注意: 矛盾がある/ないの一方だけ集計しても一意に定まらないことがある

方針2

各辺を除去したときに矛盾が生じないか調べる

愚直にやると$O(N^2)$

重み・rollback付きUnionFindを使って分割統治すると、$O(N(\log N)^2)$になる

rollback付きUnionFind:

- 過去の状態を復元できるUnionFind
- snapshot()で今の状態を記録し、後でその状態まで戻す(rollback)ことができる
- 経路圧縮をしないことで、1回のunion操作で高々2要素しか触らないので、操作履歴を保存することで実現できる

方針2

人1〜Nを順に一列に並べる

mid = floor(N/2)とする

snapshotを保存...①

[mid, N]内で矛盾がないか重み付きUnionFindで確認

矛盾がないとき➔[1,mid)について再帰的に調べる

①の状態にrollbackし、[1,mid)と[mid,N]を逆にした場合についても同様に調べる

区間が1人になったらその人が答え

その他

連結性は保証していないので、DFSするときは注意

実はN≥5（嘘つきの一意性と、多重辺や自己ループがないことから言える）

N=1はイジワルなので制約から外しました

> 嘘つきxは「xを除くと矛盾しない」「x以外の任意のyを除くと矛盾する」
> と問題文に書いてあり、N=1のとき後者はyとなりうる人はいないので真

統計情報

- Acceptances
  - 17 teams
- First Acceptance
  - Time Manipulators (93 min)