みんなのプロコン 2019 Final 解説

はじめに

- この解説スライドでは、解法そのものに加え、解くのに便利な考え方や、上手な実装の仕方も登場します。

A - Affiches

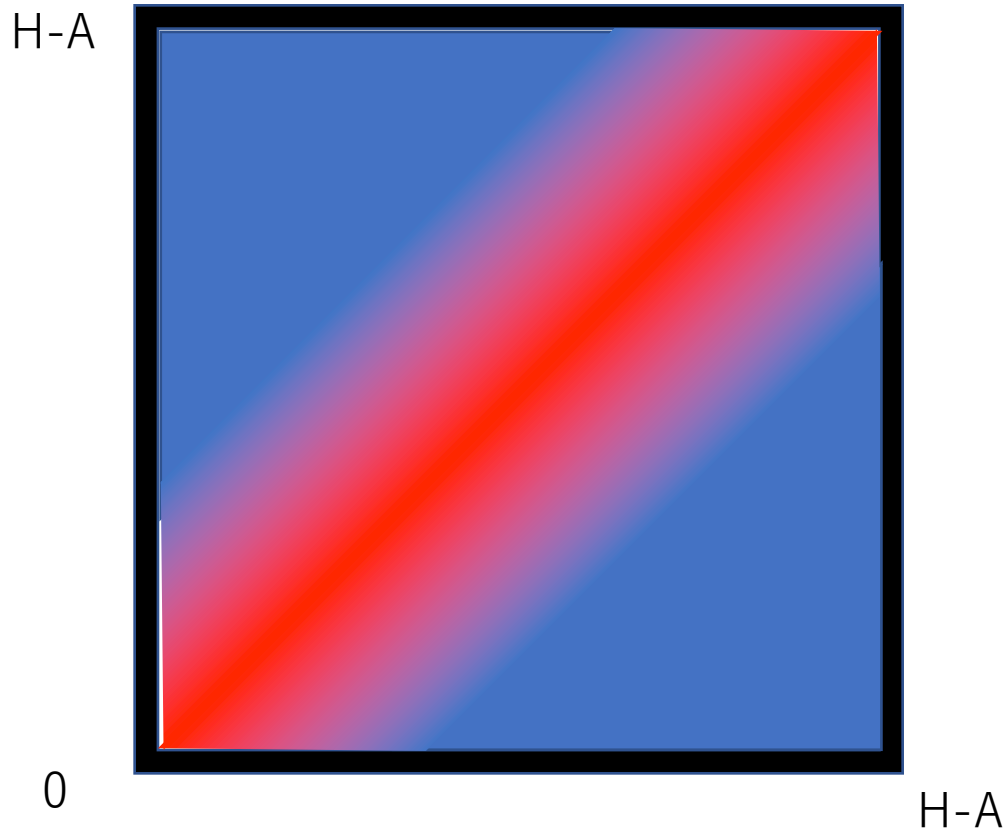
- 大長方形と小長方形 2 個が与えられます
- 小長方形を大長方形の内部に 2 個ランダムに置きます (辺が平行になるように)
- 小長方形が重なる部分の面積の期待値は？

A - Affiches

- まずはじめに、この問題は xy 独立です。
- 以下、x 軸に関して考えましょう。(1次元の問題だと思ってください)

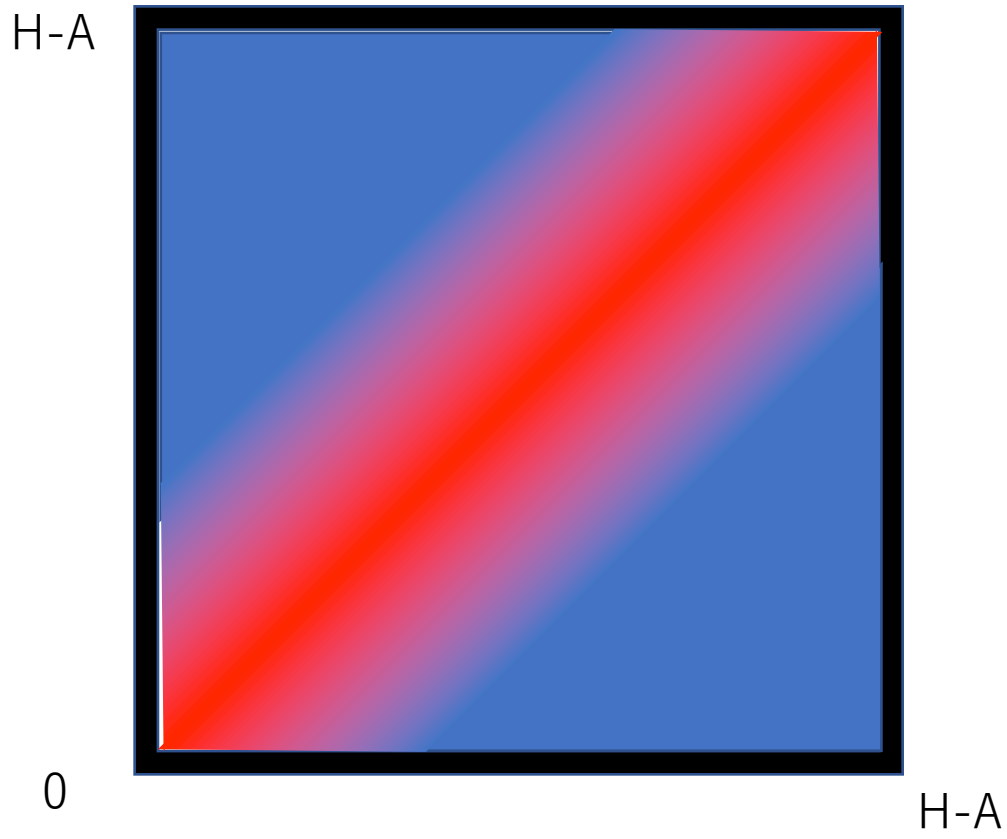
A - Affiches

- 1つ目の左端の座標をx軸に、2つ目の左端の座標をy軸にとったグラフはこんな感じです (赤いところが一番重なってる)



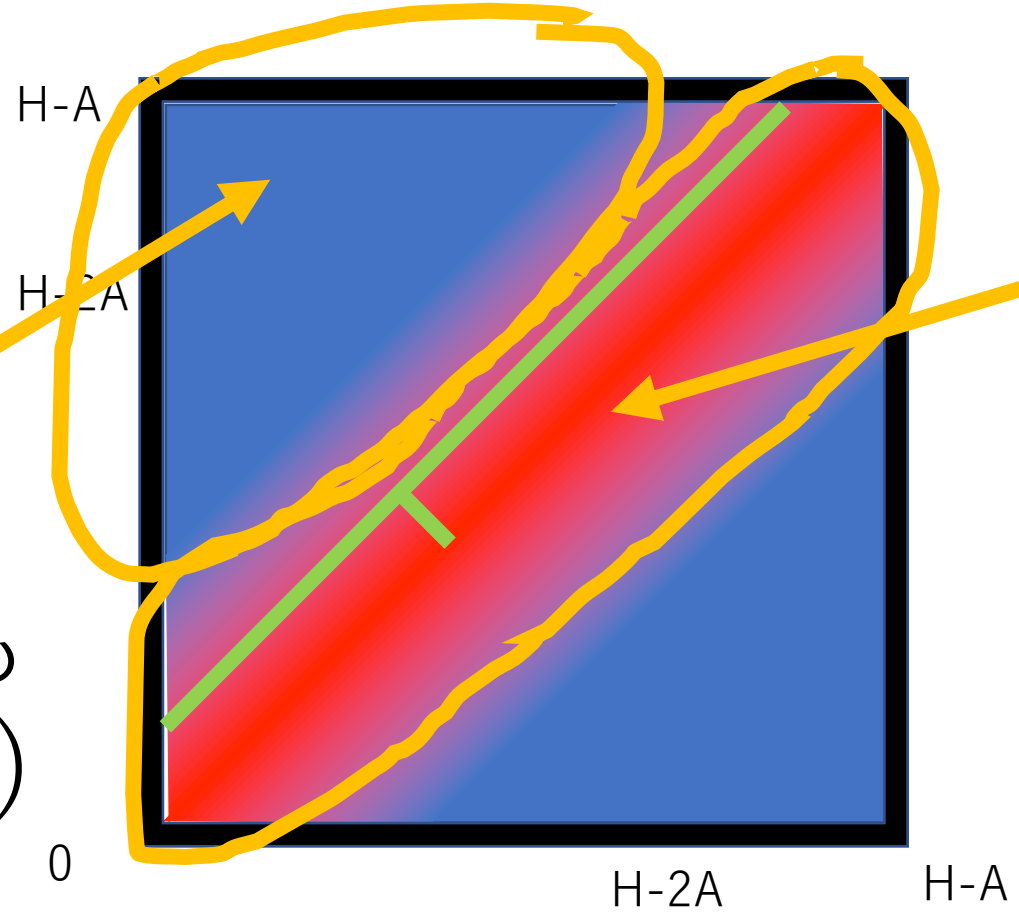
A - Affiches

- これを [0,H-A]×[0,H-A] で積分すればよい



A - Affiches

- これを [0,H-A]×[0,H-A] で積分すればよい



0
(ないかもしれない)

／対角線からの距離と幅と重なりの長さを掛けて積分

A - Affiches

- ほぼほぼ手計算
- 実装は、式をソースコードに書き写すだけ

- ネタバレ:
- T = min(A, H-A) とすると、x軸側の期待値は、
- $\frac{2(\frac{1}{3}T^3-\frac{1}{2}HT^2+HAT-A^2T)}{(H-A)^2}$

B – Bonsai Grafting

- 木が二つ与えられます
- 二つの木を一つの辺でつなぐ方法はNM通り
- それぞれについて直径を計算し、合計を求めてください

- $N, M \leq 10^5$

B – Bonsai Grafting

- 木Aの頂点u, 木Bの頂点vをつないだとき、直径には以下の 2 通りのパターンがありえます。
- パターン1: (uから最も遠い木Aの頂点) → u → v → (vから最も遠い木Bの頂点)
- パターン2: 木Aと木Bの直径のうち長い方

B – Bonsai Grafting

- 要は、それぞれの木について、各頂点から最も遠い頂点までの距離と、木の直径がわかればよいことになります。
- 直径: 2回BFSなりDFSなりすれば計算できます
- 各頂点から最も遠い頂点までの距離: 根付き木だと思って1回DFSをし、下方向に最も遠い頂点までの距離を求めます。もう一回 DFS をし、どの頂点から上に上がって下の最も遠い点に降りるかを、上から答えを確定していくことで、全ての頂点について計算できます。

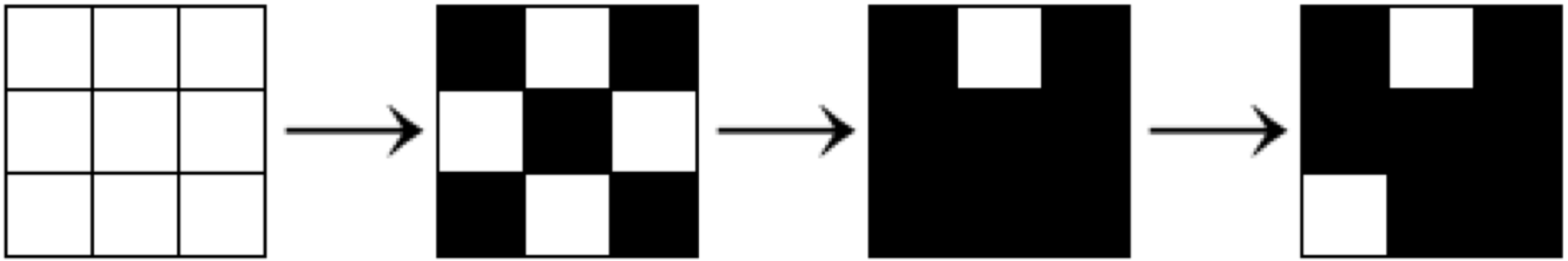
B – Bonsai Grafting

- あとは数列の問題です

- 数列 A と数列 B と整数 C が与えられる。sum(i,N)sum(i,M)max(C,A[i]+B[j]) を計算
- それぞれのA[i] について、A[i]+B[j]がCを超えるタイミングを二分探索で求めて累積和等で O(1) でまとめて合計を計算すればよいです

C – Checkered Stamps

- 紙に市松模様を反転させるスタンプを N 回押した後、黒いマスの個数を求めてください。



C – Checkered Stamps

- 仮に押すスタンプが長方形領域を全て反転させるなら…？

- JOI 2012 春合宿の Fortune Telling と同じ
- 以下のような Segment Tree を使って平面走査すると計算できます。
- Segtree1[i]: 区間 i の黒いマスの個数
- Segtree2[i]: 区間 i 全体が Segtree1 の状況と反対の状態担っているかどうか
- 区間 i を訪れるたびに、Segtree2[i]が1ならSegtree1[i]の値を再計算して適宜下に伝播します

C – Checkered Stamps

- この問題においても、以下のような方法で比較的楽に Fortune Telling バージョンに帰着できます。
- それぞれの長方形に対し、座標が偶数・奇数それぞれについて、上下左右の端の座標がどこになるか求める (ついでに /2 する)
- R[i]+C[i] が奇数のとき:
  - 上下が偶数、左右が奇数の端となる長方形を追加
  - 上下が奇数、左右が偶数の端となる長方形を追加
- R[i]+C[i] が偶数のとき:
  - 上下左右が偶数の端となる長方形を追加
  - 上下左右が奇数の端となる長方形を追加
- これら 4 通りを別々に Fortune Telling すればよいです

D – Dangerous Hopscotch

- N 個の石が 1 列に並んでいます
- Q クエリ処理:
- 　ある石の上に障害物を置いたり、取り除いたりします。石が置かれるとその石の上には乗れません
- 　L, R が与えられるので、 L 番目の石から R 番目の石に移動する方法の個数を 10^9+7 で割ったあまりを求めてください。人は、1つ右か2つ右の石に1回で移動できます。

- $Q \leq 10^5$, $N \leq 10^9$.

D – Dangerous Hopscotch

- そもそも障害物がない場合は？
- dp[i] = dp[i-1] + dp[i-2], dp[L-1] = 0, dp[L] = 1
- フィボナッチ数になる。

- 障害物があると面倒？

D – Dangerous Hopscotch

- 実はそうでもなくて、単純に障害物がある場所は
- dp[i] = 0
- であることを考えると、次のような行列の掛け算になる
- 障害物がない i に関して: [[1 1] [1 0]]
- 障害物がある i に関して: [[0 0] [1 0]]

- (2番目の式 ([1 0]) は、dp[i-1] を行列の 1 行目から 2 行目に移動しているだけ)

D – Dangerous Hopscotch

- 扱うものは行列であるが、「一点を更新する」「区間の積を求める」という2つの処理がしたいのは一緒

- Segment Tree に行列を乗せればできる
- (整数だと思って区間の和を求めるのと同じ形)

D – Dangerous Hopscotch

- 座標がでかい……

- 以下のように座標圧縮すると、一点更新もかなり楽
- クエリ 1 x に関して: x, x+1 を追加
- クエリ 2 l r に関して: l, r を追加
- 全部で 2Q 個の座標が残ることになる
- 初期状態で、 [x, x+k) の要素には $[[1\ 1][1\ 0]]^k$ を持たせる
- クエリ 1 では、[x, x+1) に対応する行列を [[1 1][1 0]] と [[0 0][1 0]] の間で変化させれば良いことになる

D – Dangerous Hopscotch

Segtree で毎回行列を計算？
めんどくさすぎる！

行列ライブラリは作るのがよいと思う
　加減乗、累乗、行列式、ランク、逆行列……

Segtree は template (かtypedef) を使って真面目にライブラリにするか、毎回手で書くか
　writerは毎回手で書いてます
　~~結果、C の Segtree がおかしくてかなり時間を溶かした~~

E – Espionage

- 高橋君は忍者
- N 個の M 階建てビルの全ての階をめぐって 1番目のビルの 1 階に戻ってくる
- 同じビルの隣の階もしくは他のビルの同じ階に移動可能
- 最初と最後以外、同じところに2度行かない

- 何通りの移動の仕方がある？ mod 10^9+7 で求めてください
- $N \leq 100$, $M \leq 10^9$.

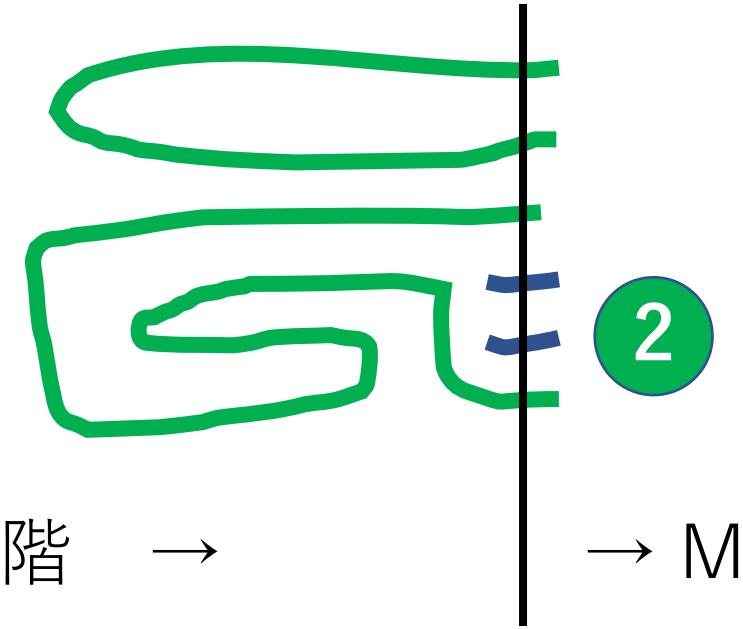
E – Espionage

- 仮に高橋君が隣の階か隣のビルにしか動けなかったら？
- お姉さん死んじゃう！

- TDPC の「列間の隣接関係を覚えておく面倒なことで有名な DP」（フロンティア法みたいな感じ）
- dp[i][S]: i 行目で、それぞれの列間での接続状況が S（ビット列）

E – Espionage

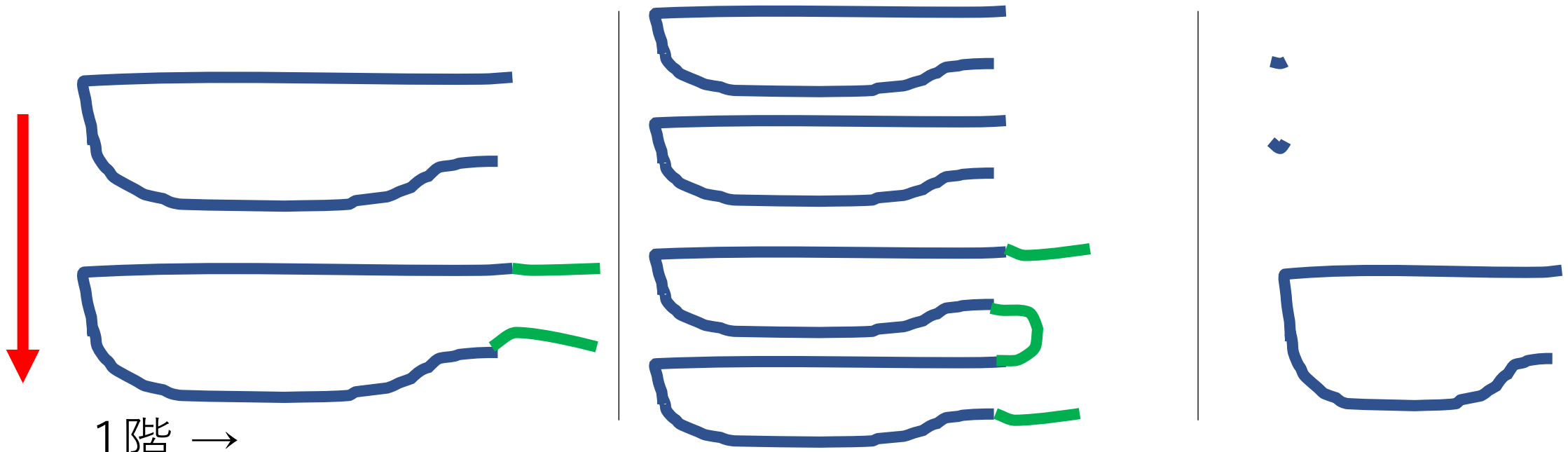
- 幸い、今回は隣以外のビルにも移動ができるので楽
- こんな状態が全部で O(N) 個なので、遷移行列が作れれば行列累乗で答えが求められる



ある階で、左につながったループが
i 個

E – Espionage

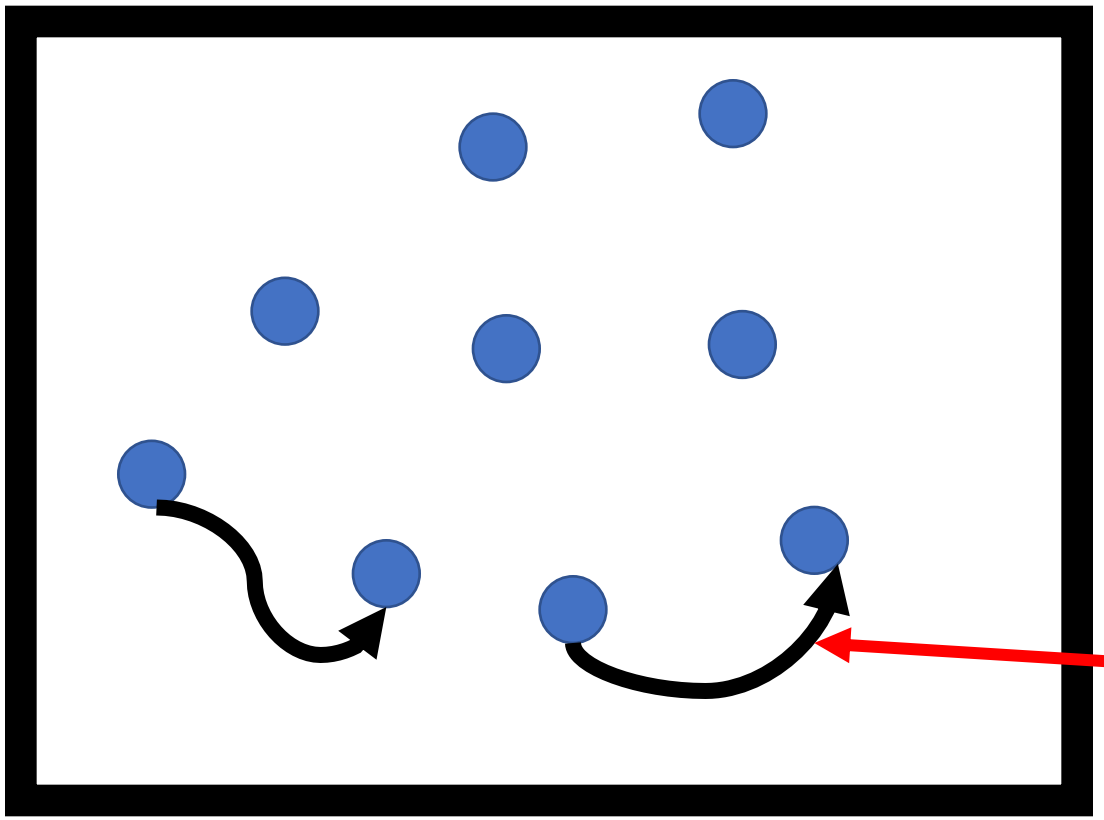
- 下の階からループの断片を構成していく
- できることは、「既存のループを伸ばす」 「既存のループの断片をつなげる」 「新しいループを作る」の3通り



E – Espionage

！！！有向にすると楽です！！！

- では、この遷移はどうすれば……



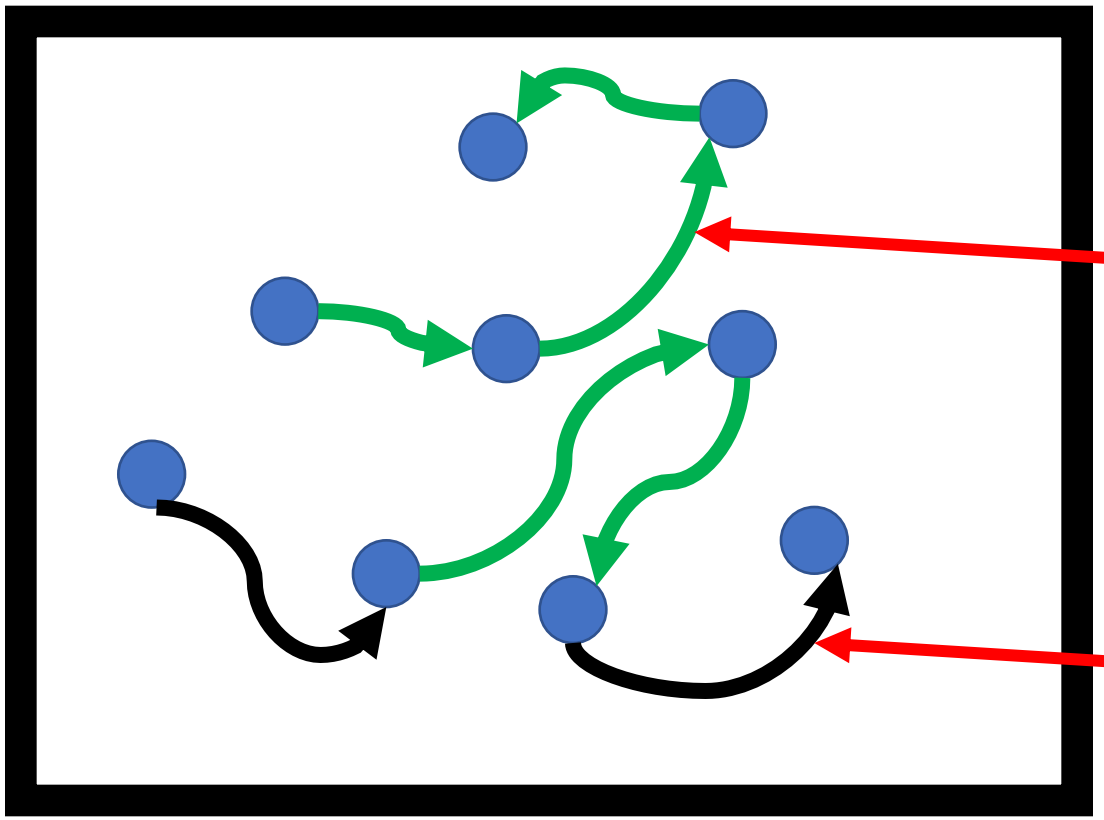
←ある階の様子

すでに下の階を経由して繋がっている 2つのビル

E – Espionage

！！！有向にすると楽です！！！

- では、この遷移はどうすれば……



←ある階の様子

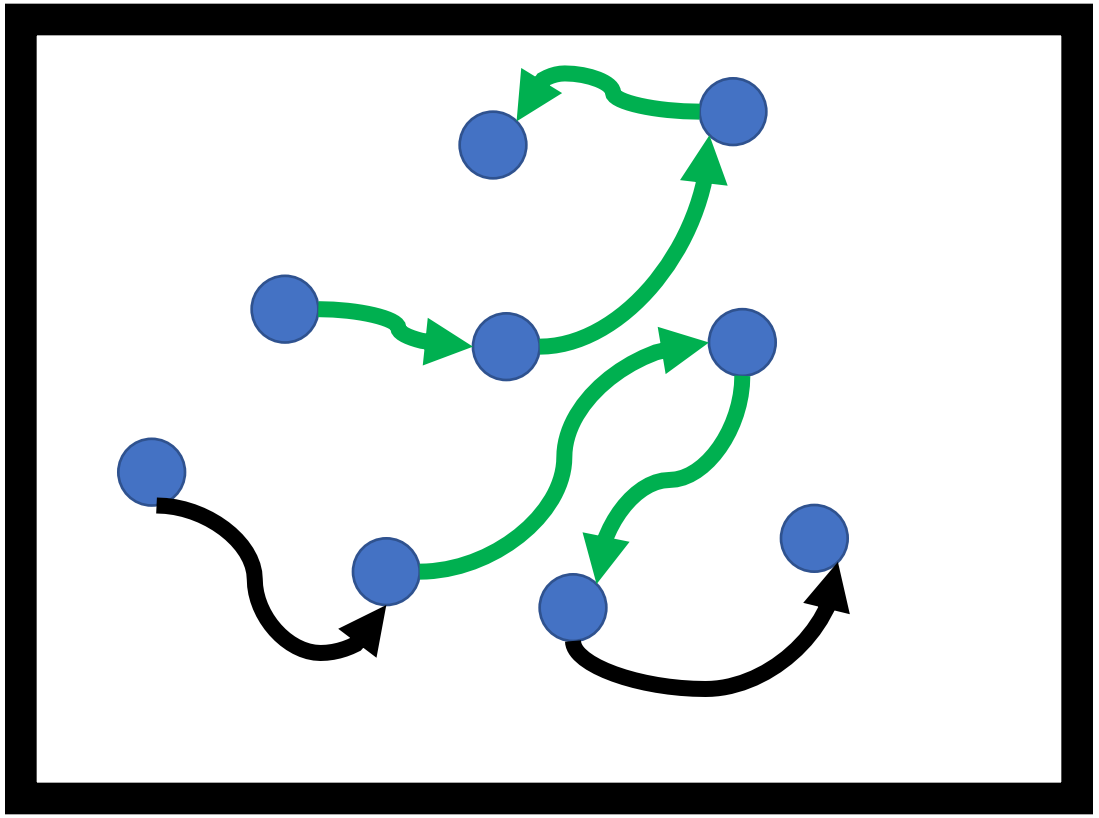
この階で新しくつなぐ

すでに繋がっている 2つのビル

E – Espionage

！！！有向にすると楽です！！！

- では、この遷移はどうすれば……



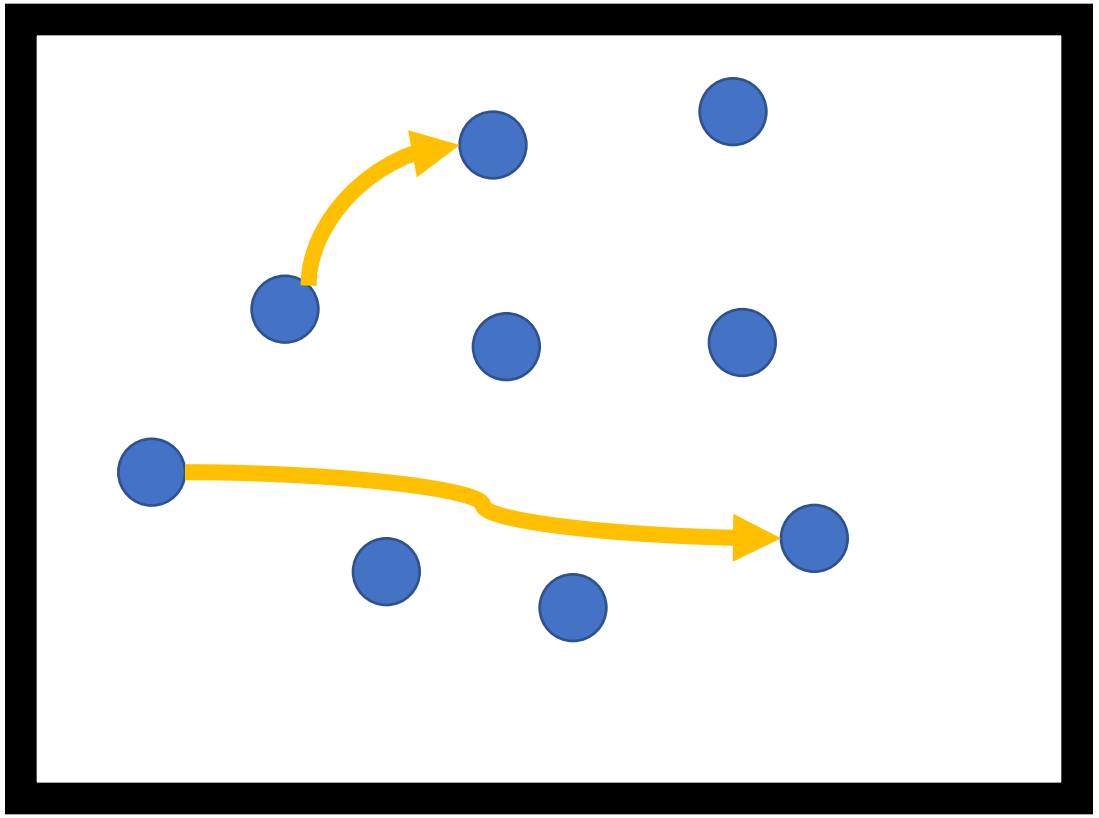
条件: 矢印でつなぐときに、ループを作らない(最後の階ではできるので、そこは特別に処理する)
孤立点が存在しないようにつなぐ

↑は包除原理で $O(N^3)$ で計算できる

E – Espionage

！！！有向にすると楽です！！！

- では、この遷移はどうすれば……



一つ上の階はこんな感じになる

E – Espionage

！！！有向にすると楽です！！！

- まとめ
  - 包除原理 で遷移行列が作れる $O(N^3)$
  - かわりに DP を 2,3 個して遷移行列を作っても良い $O(N^3)$ か $O(N^3 \log N)$ か $O(N^4)$ どれでも間に合う

  - 行列を累乗する
  - 最後の階を全部つなぐ