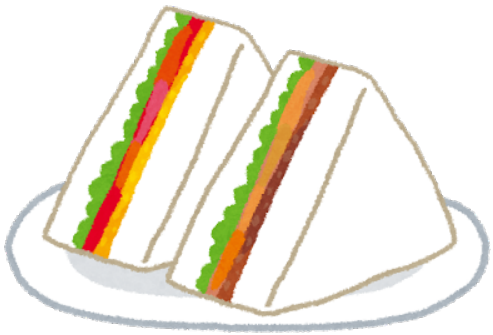
サンドイッチ (Sandwich)
解説

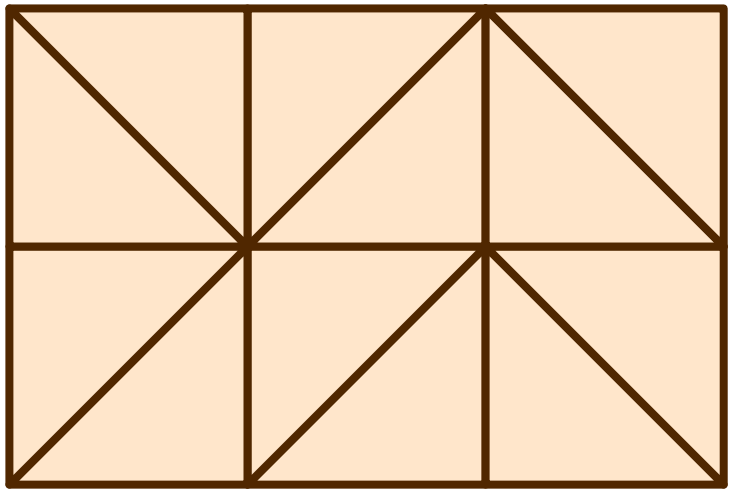
JOI 2016 春合宿 day2

snuke



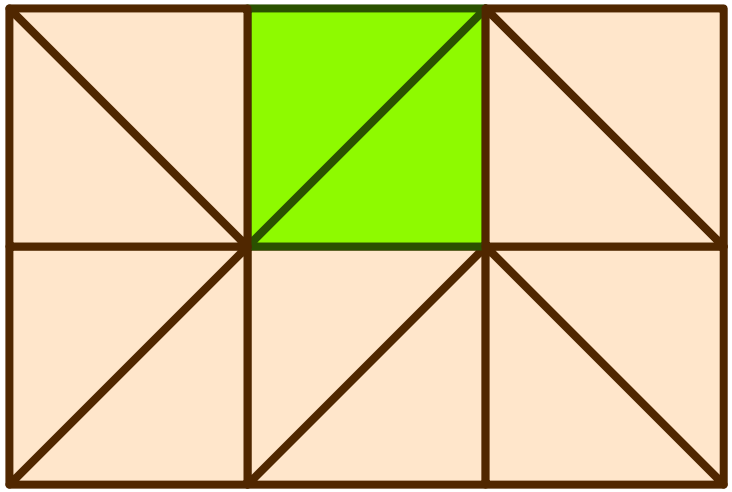
問題概要

- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？



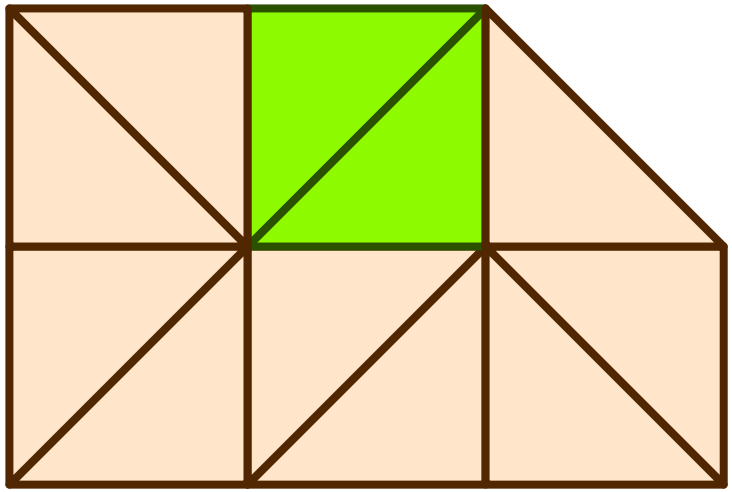
問題概要

- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？



問題概要

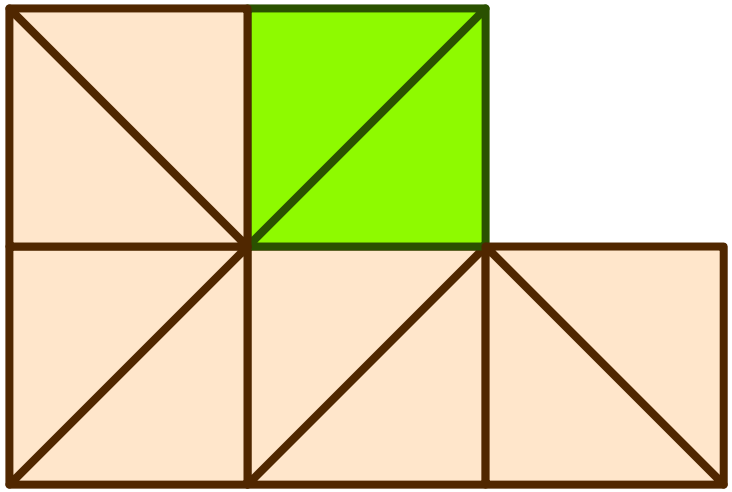
- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？



やってみよう!

問題概要

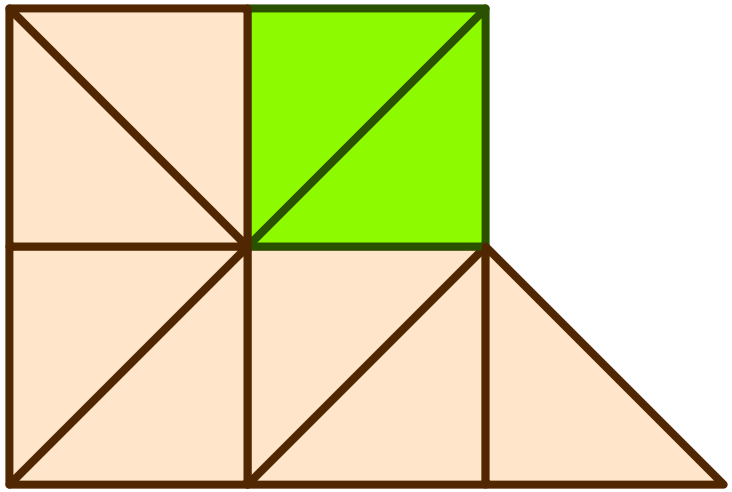
- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？



やってみよう!

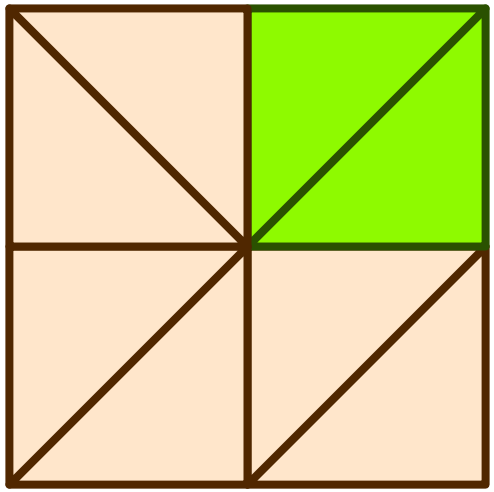
問題概要

- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？



問題概要

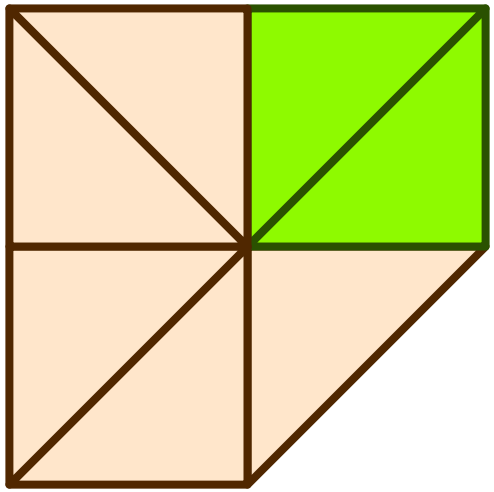
- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？



やってみよう!

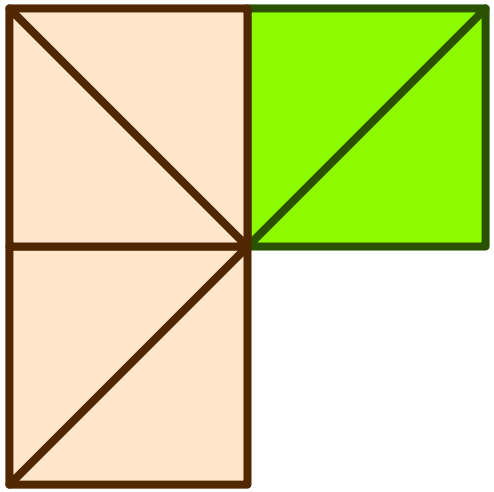
問題概要

- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？



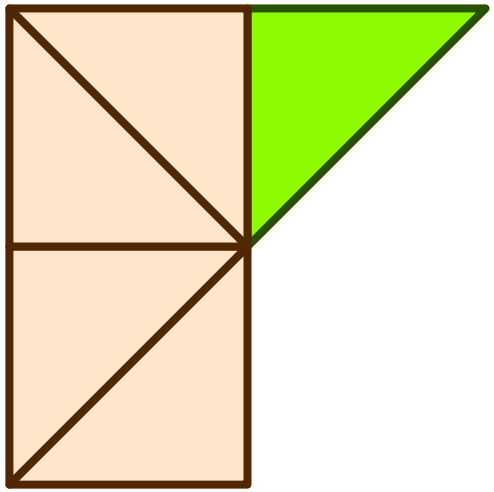
問題概要

- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？



問題概要

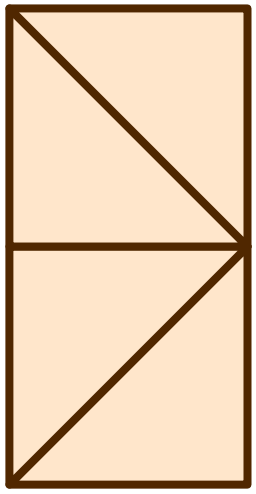
- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？

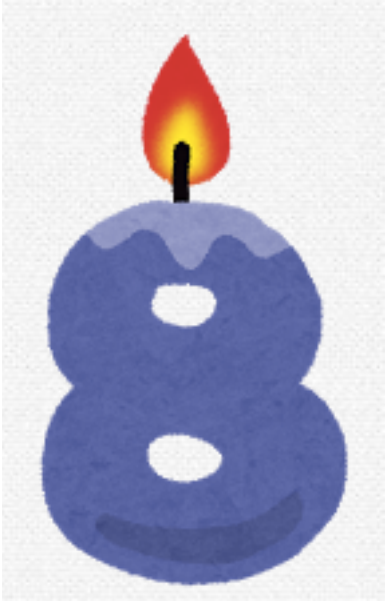


やってみよう!

問題概要

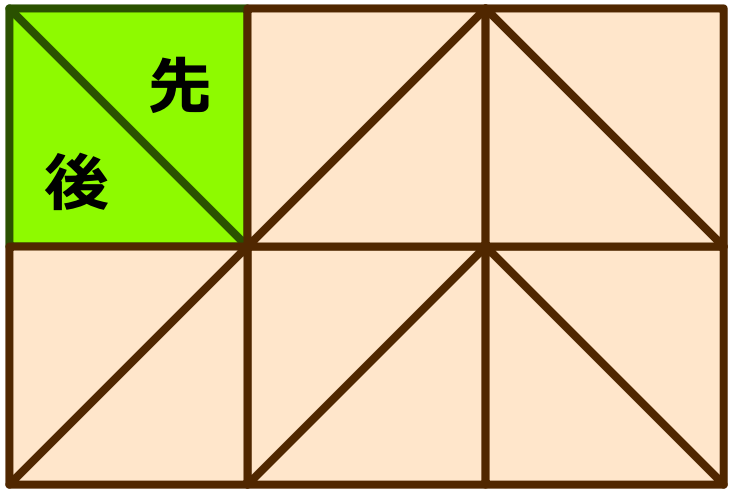
- R行C列のマス目にサンドイッチが並んでいる
- 斜辺または他の二辺が空いてるサンドイッチは取れる
- 各マスについて、そのマスのサンドイッチを両方取るために取る必要のあるサンドイッチの個数の最小値は？





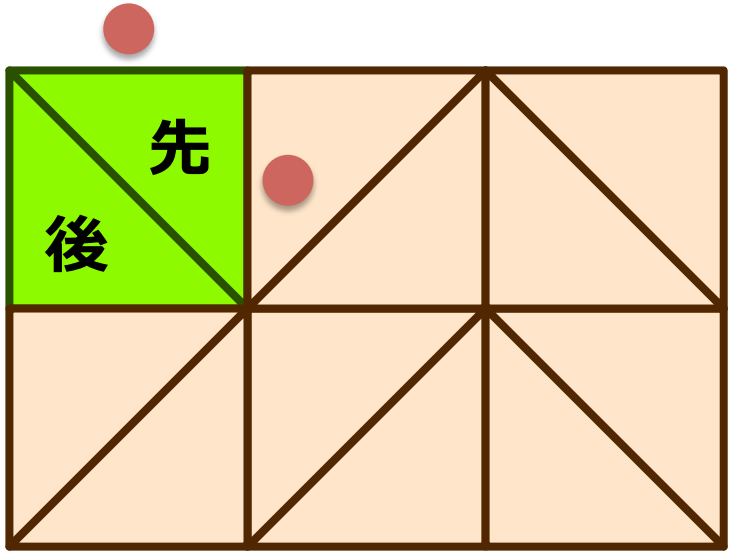
小課題 1 (35点)

- 目的の２個のサンドイッチの取り方は２通りある
  (A) 上側のサンドイッチを先にとって下側を次に取る
  (B) 下側のサンドイッチを先にとって上側を次に取る
- (A)についての最小値が求められれば、(B)についても同様な方法で求めることができるので、(A)のみを考える



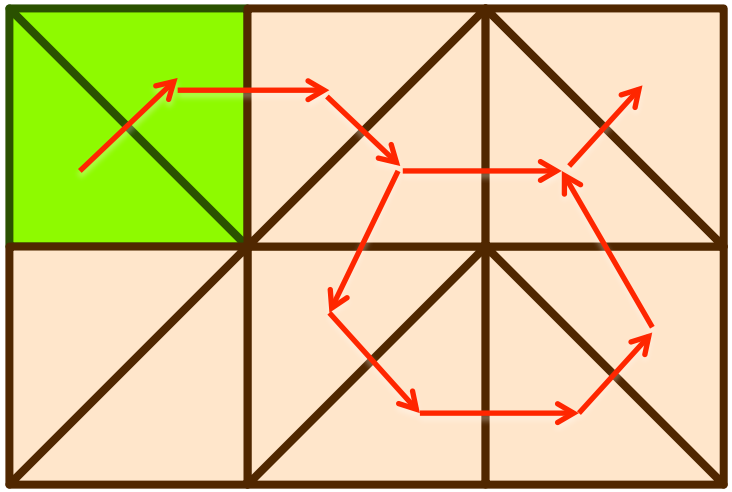
小課題 1 (35点)

- 上側のサンドイッチを取るためには、赤マルの位置のサンドイッチを先に取らなければならない
- このように「Xを取るためには先にYを取らなければいけない」という状況を、「XがYに依存している」と呼ぶことにする



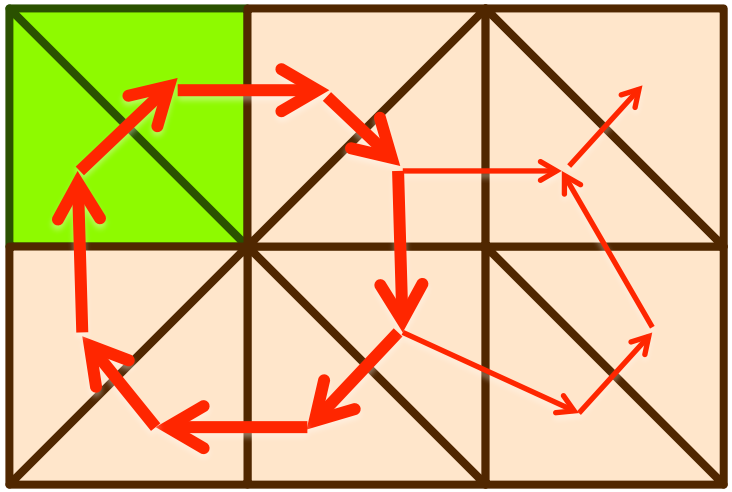
小課題 1 (35点)

- XがYに依存しているとき、X→Yという辺を貼ってみる
- 辺が貼られたものを逆順に、取れるものから取っていけば緑のサンドイッチを取ることができる



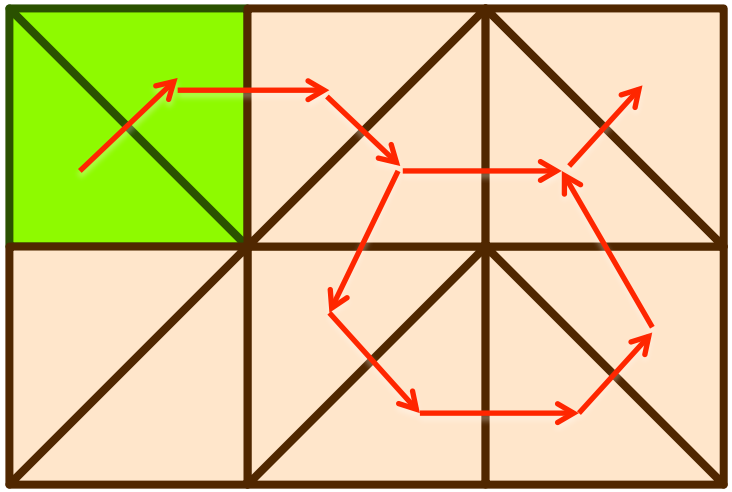
小課題 1 (35点)

- このようにサイクルができる場合は取ることが出来ない



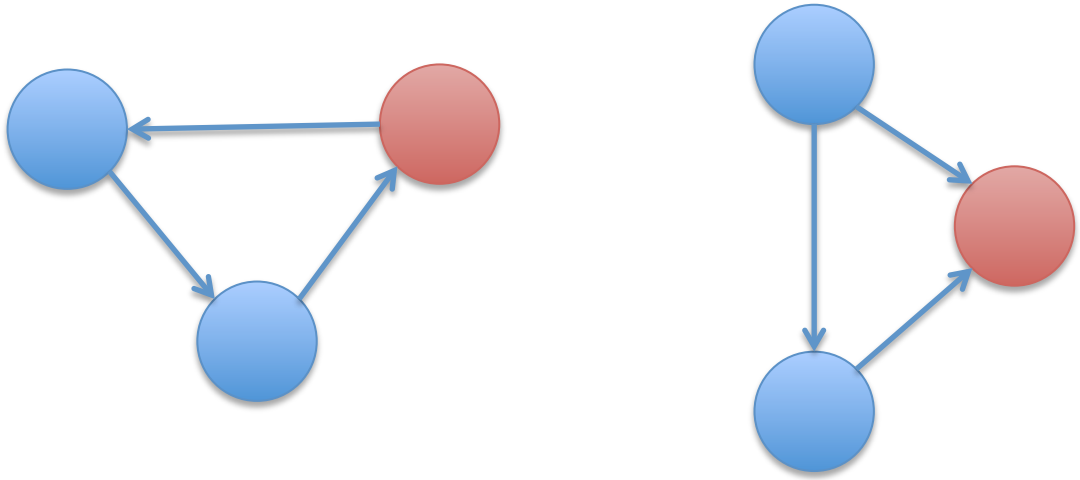
小課題 1 (35点)

- 各マスについて、(A),(B)パターンを試す
  - 依存関係を列挙していき、辺を貼る
  - 辺が貼られたサンドイッチの個数を求める
  - ただし、サイクルが存在する場合は取れないと判定する
- 計算量：O((RC)^2)



小課題 1 (35点)

- 具体的にどう計算するか
  - 単にbool visited[MAX_V]のような配列で状態を管理していると、「サイクル」なのか「単に２回訪れただけ」なのかが判定できない



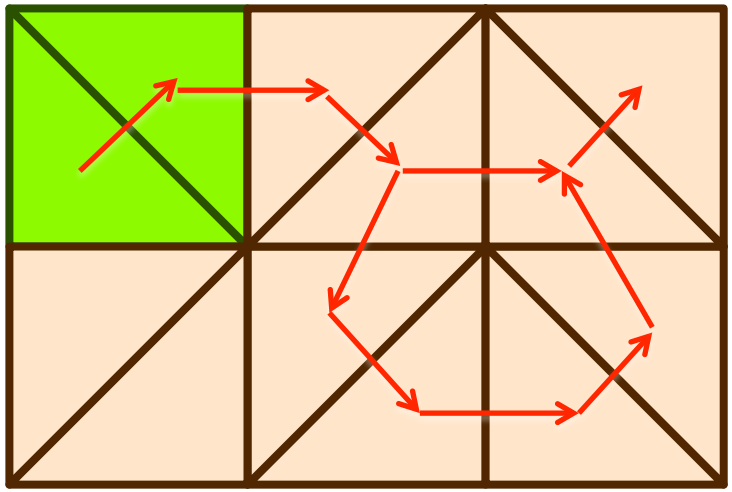
小課題 1 (35点)

- 具体的にどう計算するか
  - DFSを用いると簡潔に実装できる

```
int state[MAX_V]; // 0 で初期化
int dfs(int v) {
  if (state[v] == 1) return INF;
  if (state[v] == 2) return 0;
  state[v] = 1;
  int sum = 1;
  for (int u : to[v]) {
    int res = dfs(u);
    if (res == INF) return INF;
    sum += res;
  }
  state[v] = 2;
  return sum;
}
```

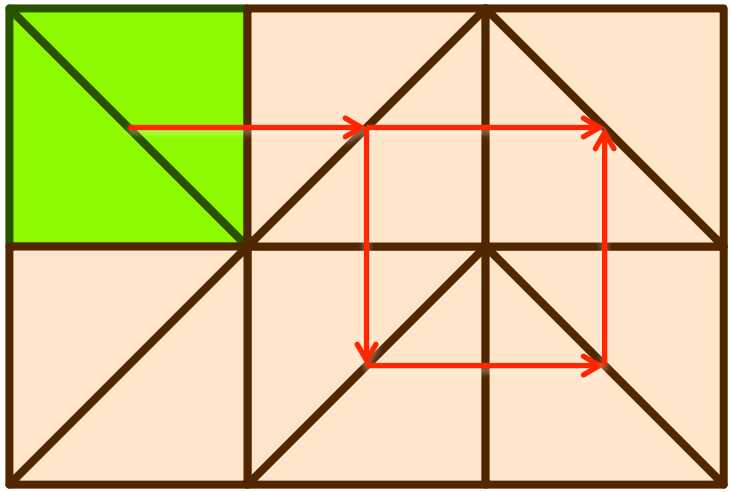
実装の簡略化

- サンドイッチが三角形なのは扱いにくいので、マス単位で考えることにする



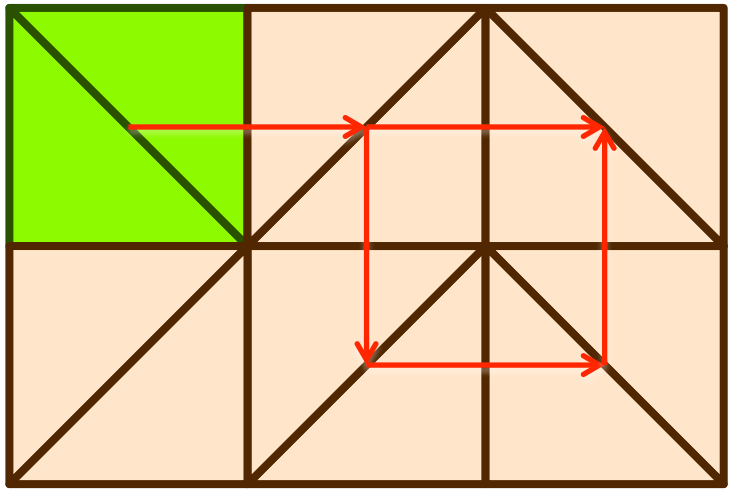
実装の簡略化

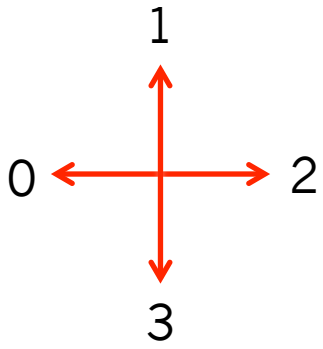
- サンドイッチが三角形なのは扱いにくいので、マス単位で考えることにする



実装の簡略化

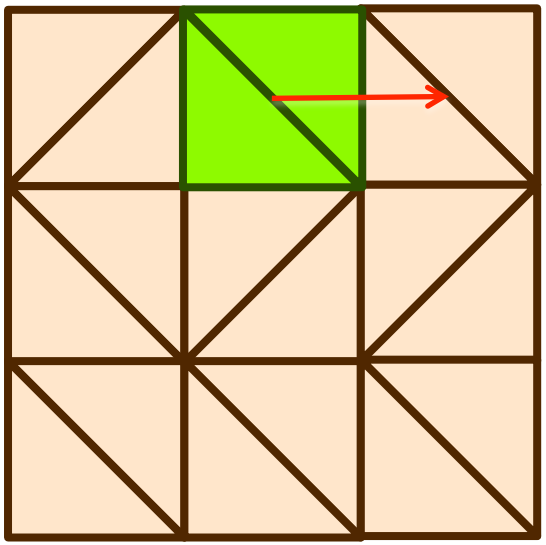
- マス目の隣接関係を扱うときのテクニック
  - dx[] = {-1, 0, 1, 0}, dy[] = {0,-1,0,1}
  - for(v=0~3) nx = x+dx[v], ny = y+dy[v]
- v という方向で‘Z’のマスに入ってきた時、そのマスから貼る辺は v と v xor 1 の方向のマス
  - ‘N’のマスなら v と v xor **3**





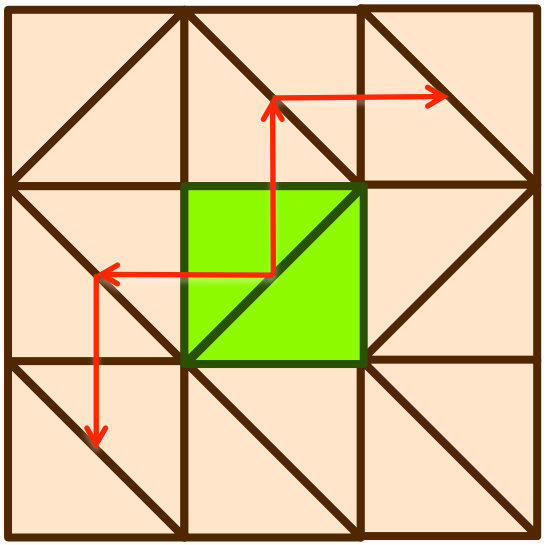
小課題 2 (満点)

- 列ごとに考える
- 列の上から順に依存関係を計算していく



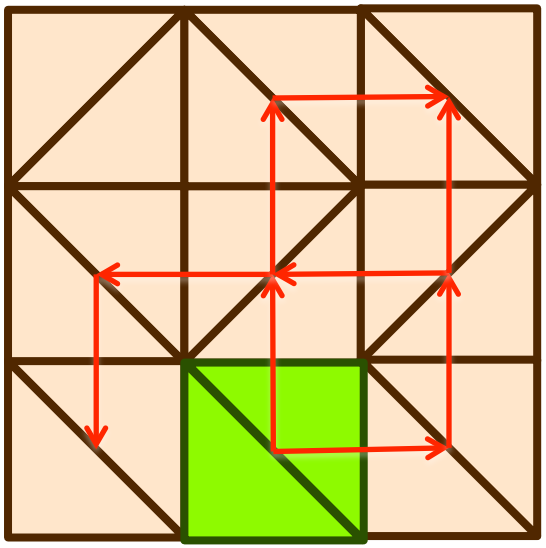
小課題 2 (満点)

- 列ごとに考える
- 列の上から順に依存関係を計算していく



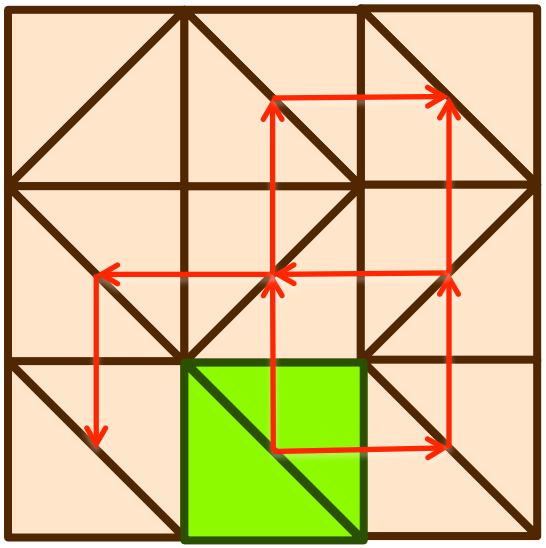
小課題 2 (満点)

- 列ごとに考える
- 列の上から順に依存関係を計算していく



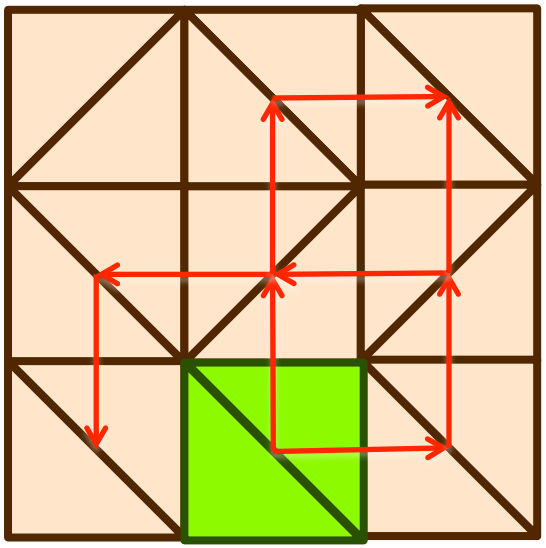
小課題 2 (満点)

- 列ごとに考える
- 列の上から順に依存関係を計算していく
- **依存関係の辺は単調増加！**



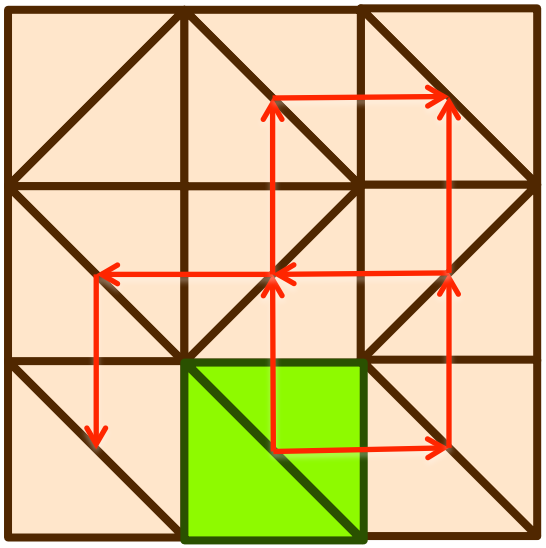
小課題 2 (満点)

- 列ごとに考える
- 列の上から順に依存関係を計算していく
- **依存関係の辺は単調増加！**
  - 上側のサンドイッチは必ず上方向に依存しており、
    上方向に依存されたマスは上方向に依存しているため



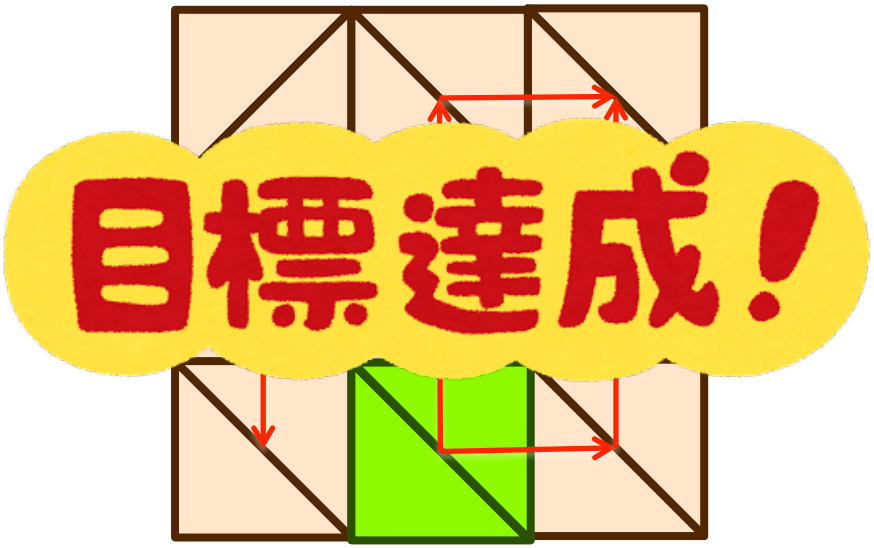
小課題 2 (満点)

- 列ごとに、上から順に依存関係を計算する
- 同じ列内では依存関係を初期化をせずに計算していく
- 依存関係の個数は高々O(RC)なので、列ごとにO(RC)で計算でき、全体でO(RC*C)となる



小課題 2 (満点)

- 列ごとに、上から順に依存関係を計算する
- 同じ列内では依存関係を初期化をせずに計算していく
- 依存関係の個数は高々O(RC)なので、列ごとにO(RC)で計算でき、全体でO(RC*C)となる



統計

0 15

35 4

100 1

統計

菩提樹の木の下で悟りを開くブッダのイラスト

0

15

巫女さんのイラスト

35

4

おからのイラスト

100

1