CODE FESTIVAL 2015 沖縄ツアー
解説

AtCoder株式会社 代表取締役
高橋 直大

A問題　Automatic Map Generator

A問題　問題概要

AtCoder

- H, W, Kが与えられる。
- H×Wのグリッドに陸を配置して、陸の連結成分の個数をK個にできるか判定せよ。
- できるならば実際に配置の仕方を一つ出力せよ。

- 制約
- 1 ≦ H,W, ≦ 100, 1 ≦ K ≦ 10,000

A問題　アルゴリズム

AtCoder

- 大量に敷き詰められる例を考える
- #.#.#.#.

  ....

  #.#.#.#.
- とりあえず↑のように奇数行目の奇数文字目にのみ陸を配置すると、連結成分の個数は((H+1)/2) * ((W+1)/2) 個になる
- これより小さくしたいときは、いくつか '#' を '.'にすればよい

A問題　アルゴリズム

AtCoder

- 果たしてこれが上界だろうか？
- H × W のグリッドを2行ごと、2列ごとに分割
- AABBCCDDE
  AABBCCDDE
  FFGGHHIIJ
- この分割されたものの個数は前述の((H+1)/2) * ((W+1)/2)に等しい

A問題　アルゴリズム

AtCoder

- さらに 1 つ陸を追加しようとしても、すでに1つ入っている領域に2つ目を追加することになる
- 同じ領域に2つあっても(どの組み合わせでも必ず)ひとつながりになってしまうので、K を増やすことはできない
- よって先ほどの値 ((H+1)/2) * ((W+1)/2) が上界となる。

A問題　アルゴリズム

AtCoder

- よって、
- K > ((H+1)/2) * ((W+1)/2)のとき“IMPOSSIBLE”
- そうでない時は、可能
  - 出力例としては、前述のものがある

B問題　Beware of the Sogginess!

1. 問題概要
2. アルゴリズム

B問題　問題概要

AtCoder

N種類の沖縄そば (麺a_iグラムとスープb_iグラム) のうちいくつかを買い、それぞれに対し独立に待つことで 麺a_i+tグラムとスープb_i-tグラムにすることができる。麺Xグラム以上、スープYグラム以上食べるには最低何種類買えばいいか？

- 制約
- $1 \leqq N \leqq 50$, $1 \leqq a_i, b_i, X, Y \leqq 10{,}000$

B問題　アルゴリズム

AtCoder

- 状況の整理
- a_iの合計がA、b_iの合計がBとなるように買って好きなように待つと、麺の量とスープの量はどのような値をとりうるか？
- 麺の量: A ～ A+B
- スープの量: B ～ 0
- 合計は常に A+B

B問題　アルゴリズム

AtCoder

- スープと麺の合計は変わらないので、とりあえずこの合計が X+Y グラム以上となるように買う必要がある
- その中で、麺が足りないときは、適宜待つことで麺を増やすことができる
- しかし、スープは減る一方
- 　→　スープの合計が Y グラム以上となるように買う必要がある

B問題　アルゴリズム

AtCoder

- ということで次のような問題になった
  - B[i], C[i] (=A[i]+B[i]) が与えられる。このうちいくつか選んで ΣB[i]≧Y, ΣC[i]≧ X+Y となるようにしたい。選ぶ個数の最小は？
- 上の問題はDPで解ける、しかし
  - dp[s][t]:= ΣB[i]=s, ΣC[i]=tとする最小個数
  - O(NX(X+Y)) → これは計算量的に厳しい

B問題　アルゴリズム

AtCoder

- 漸化式の工夫
- dp[s][t]:= s個でΣB[i]=tとなるときのΣC[i]の最大
- 上の漸化式を更新していけば $O(N^2Y)$
- これで間に合う

C問題　Cat versus Wolf

1. 問題概要
2. アルゴリズム

C問題　問題概要

AtCoder

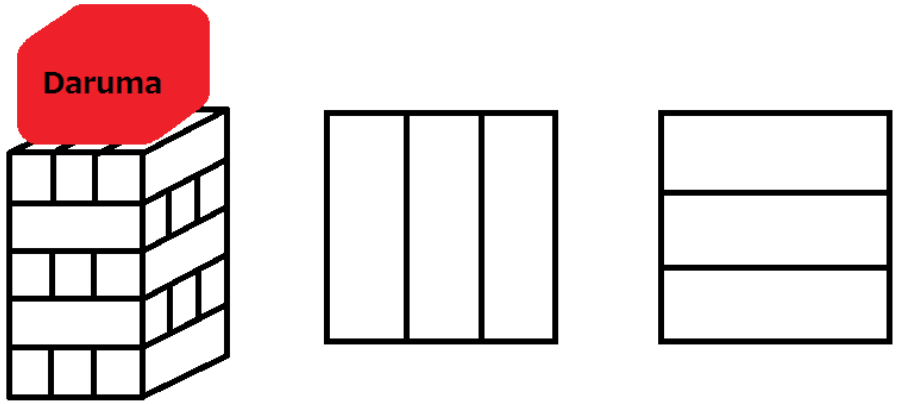
- 下図のように積まれたブロックを2人交互に取り去っていく

不安定になるように取り除いてはいけない

取り除けなくなったら負け

最善にプレイすると先手後手どちらが勝つか？

- 制約
- $1 \leqq N \leqq 50,000$



C問題　アルゴリズム

AtCoder

- 取り除くことができるブロックは2種類に大別される
  - その段に全てのブロックがあるとき(ブロックAと呼ぶことにする)
  - その段の一方の端のブロックがすでに無いとき(ブロックBと呼ぶことにする)
- ともに1つもないとき、負け
- それぞれのブロックを取り除くと、これら2種のブロックの個数がどう変化するか考える

C問題　アルゴリズム

AtCoder

その段に全てのブロックがあるとき(A)

　真ん中を取る→ Aが1減る

　片方の端を取る→Aが1減る、Bが1増える

一方の端がすでにないとき(B)

　もう一方の端を取る→ Bが1減る

C問題　アルゴリズム

AtCoder

- 操作がこの3種類であるということから、次のことがわかる
  - A, Bの残り個数がともに偶数のとき、手番を行うとどちらか一方が奇数個になる
  - A, Bの残り個数のいずれかもしくは両方が奇数のとき、適切に操作するとともに偶数にできる
  - A=0, B=0 (ともに偶数)のとき負け

C問題　アルゴリズム

AtCoder

- よって、与えられた盤面からA,Bの個数を計算し、A,Bがともに偶数のときはその番の人は負け、いずれかもしくは両方が奇数のときはその番の人が勝つ。
- それまでに取られたブロックの個数の偶奇も数えれば2人のどちらが勝つかも求められる。
- O(N)

D問題　Dictionary for Shiritori Game

1. 問題概要
2. アルゴリズム

D問題　問題概要

AtCoder

- 辞書に載っている語でしりとりをする
- ただし同じ単語は何回でも言える
- 最初は文字0から始まる単語を言う
- 最善を尽くしたとき、先手が勝つか？後手が勝つか？永遠に終わらないか？
- 文字の種類数 ≦ 100,000
- 単語の個数 ≦ 200,000

D問題　アルゴリズム

AtCoder

・まず、自明に負ける例としては、その文字から始まる単語が無い文字が自分の番で来たとき

•(そのような文字がないなら、終わらない)

•ということは、相手に「自明に負ける例」の状況を起こさせれば勝てる

・相手に「相手に「自明に負ける例」の状況を起こさせれられる」状況にされてしまったら負け

……

D問題　アルゴリズム

AtCoder

- 先ほどのことを考えると、「自明に負ける点」から逆に勝敗を決めていくのがよさそうとわかる。
- 逆から探索してみよう

D問題　アルゴリズム

AtCoder

- そこからだと負ける頂点を -1 、勝つ頂点を 1 として探索をしたい。
- ある頂点から -1 に向かう辺がある頂点
  - → 1となる
- ある頂点から向かう辺が全て 1 となっている頂点
  - → -1 となる

D問題　アルゴリズム

AtCoder

- 辺を逆向きにしてBFSをする
- 「自明に負ける頂点」を -1 にしてスタート
- -1 の頂点から到達可能な頂点は全て 1
- 1の頂点から到達可能な頂点には、BFSで行き先の頂点を参照するたびにカウントしていく
  - カウントが「その頂点に入る辺の個数」となったらその頂点を -1 にして追加

D問題　アルゴリズム

AtCoder

- 最終的に、しりとりのスタートの頂点が 1 か -1 かどちらでもないかで答えが分かる。
- 1のとき: 先手が必勝
- -1 のとき: 後手が必勝
- どちらでもないとき: 永遠に終わらない
- 計算量は O(N+M)

E問題　Enormous XOR Rectangle

1. 問題概要
2. アルゴリズム

E問題　問題概要

AtCoder

- H*W のグリッドが与えられる。このグリッドには左上から順に、0-originで番号が振られている。
- すなわちi行j列目には (i-1)*W+(j-1) が書かれている。
- 部分長方形を選んでその長方形内の全ての値のbitwise xorを最大化せよ。
- 1≦H,W≦1,000,000,000

E問題　アルゴリズム

AtCoder

- 自明な上界を考える
- $2^{n-1} < H*W \leqq 2^n$のとき、グリッドに書かれた全ての数は2進表記でn桁以下
- よって、このとき答えが 2^n以上になることはない。
- よって、2^n-1となるような部分長方形を見つけたらそれが答えになる。

E問題　アルゴリズム

AtCoder

- 答えが 2^n-1となる簡単なケース:
- $2^{n-1}-1$と2^{n-1}が左右に隣り合っている場合
- この2つを部分長方形に選ぶと、xorした値は2^n-1となる。
- 多くのケースで$2^{n-1}-1$と2^{n-1}は左右に隣り合っていることがわかる。

E問題　アルゴリズム

AtCoder

- $2^{n-1}-1$と2^{n-1}が左右に隣り合っていない場合
- そもそもこのとき、$2^{n-1}-1$がグリッドの右端にあることになる。0～$2^{n-1}-1$まででグリッドの総数は2^{n-1}
- 素因数分解を考えると、このとき W が 2^m ($0 \leqq m \leqq n-1$) であることが容易に分かる。
- このときは、上位n-mビットと下位mビットを分けて考えてみる。

E問題　アルゴリズム

AtCoder

- $2^{n-1}-1$と2^{n-1}が左右に隣り合っていない場合
- 自明な下界は 2^n-2^m
  - $2^{n-1}-2^m$と2^{n-1}は上下に隣り合っている
- これより大きな答えを達成するには、奇数行を選び、上位n-mビットを全て1にする必要がある。
- 右端の列のみ選んだとしても変わらないので、次からそうしたとして考える。

E問題　アルゴリズム

AtCoder

- $2^{n-1}-1$と2^{n-1}が左右に隣り合っていない場合
- すると、上位n-mビットのみを考えた1次元の問題に落ちる。
- 0 ～ H-1 が一列に並んでいる。連続する奇数個を選んで全てのxorした値を $2^{n-m}-1$にできるか。

E問題　アルゴリズム

AtCoder

- $2^{n-1}-1$と2^{n-1}が左右に隣り合っていない場合
- 上のビットから考えていく。
- 最上位ビットは奇数個選ばれるから、最上位ビットが0のものは大きいものから偶数個選ばれ(0個もありうる)これらのxorは0なので、ここは今後無視できる。
- ということは、その次のビットが1のものを奇数個選ぶには、最上位と次のビットがともに1のものを奇数個選ぶ必要がある。

E問題　アルゴリズム

AtCoder

- $2^{n-1}-1$と2^{n-1}が左右に隣り合っていない場合
- そうするとやはり2番目のビットが0になるものは偶数個選ばれるので、無視ができる
- すると、大きいほうから3番目のビットについても、上位3つのビットが全て1のものを奇数個選ぶ必要が出てくる
- 下から2番目のビットまで同様のことがいえる。

E問題　アルゴリズム

AtCoder

- $2^{n-1}-1$と2^{n-1}が左右に隣り合っていない場合
- ただし最後のビットは例外的に、隣接する偶数個の数をとってきてもxorが1となりうるので、n-m個のビット全てが1であるものが存在するとは限らない。
- 以上より、$H=2^{n-m}-1$ or 2^{n-m}である必要がある。
- 上記二つの場合は、実際に答えを 2^n-1にすることが可能(1行目から$2^{n-m}-1$行目までの右端を選ぶ)

E問題　アルゴリズム

AtCoder

- 答えをまとめると、
- $2^{n-1}<H*W\leqq 2^{n}$とする。
- Wが2のべき乗数でないとき、もしくはWが2のべき乗数で、Hもしくは H+1が2のべき乗数の場合
  - 答えは $2^{n}-1$
- そうでない場合
  - 答えは $2^{n}-W$

F問題　Falconry

1. 問題概要
2. アルゴリズム

F問題　問題概要

AtCoder

- 2次元座標平面上のN個の点に3羽の鳥が訪れる。
- それぞれの点に対して、最低1羽が訪れればよい
- 全ての点に訪れるために必要な3羽の移動距離の合計を求めよ
- N ≦ 18

F問題　アルゴリズム

AtCoder

- 3羽……？
- とりあえず1羽の場合を考えてみる
- 普通のTSPなので bit DP で解ける
- 「それぞれの鳥に対して 2^N 通りの訪れ方に対して最短距離を求める → 3つをまとめる」、という方法ではどうしても $O^*(3^N)$ 程度になってしまう

F問題　アルゴリズム

AtCoder

- 3羽の鳥が同時に出発すると考えると複雑
- 合計時間なので、1羽ずつ順に移動していき、前の鳥が訪れる場所を全て訪れてから次の鳥が移動を開始すると考えていっても変わらない
- すると、dp[i][j][S]:= i番目の鳥が現在jにいて、訪れた頂点集合がSとなる最短距離
- というTSPの変形のbitDPで解ける。

F問題　アルゴリズム

AtCoder

- 計算量はTSPと同じ $O(2^N \cdot N^2)$
- 定数が重いが、Time Limitが長いので間に合う。
- このような問題では '3' という定数にとらわれず、より一般的な例を考えることが有用なこともある。

G問題　Gourgeous Vases

1. 問題概要
2. アルゴリズム

G問題　問題概要

AtCoder

- 2つの花瓶があり、片方は空。もう片方に青い花がA本と赤い花がB本入っている。(A≧B)
- 花を1つずつもう空の方の花瓶に移していく。
- 2つの花瓶どちらについても、常に青い花の本数が赤い花の本数以上でないといけない。
- 特定の本数の組み合わせにしてはいけない
- 移し方は何通りあるか？
- 1≦A≦B≦100,000　(ダメな配置の数)≦20

G問題　アルゴリズム

AtCoder

- このままでは図にしにくいので、よくある経路の個数を数える問題に言い換えてみる
- 2次元のグリッドで (0,0) から (A,B) まで (+1,0) と (0,+1)の2種類で行く方法は何通りあるか？
  - ただし通る点(i,j)は全てi≧jかつA-i≧B-j
  - (p[i],q[i]),(A-p[i],B-q[i])は通れない

G問題　アルゴリズム

AtCoder

- まず、N個の障害物がある状況はややこしいのでこれを次のようなDPで解消する。
- 以下、障害物と(A,B)にそれぞれ適当に頂点番号をつけたとする
- dp[i]:=番号のついた頂点のうち、頂点iに最初に到達する方法の個数

G問題　アルゴリズム

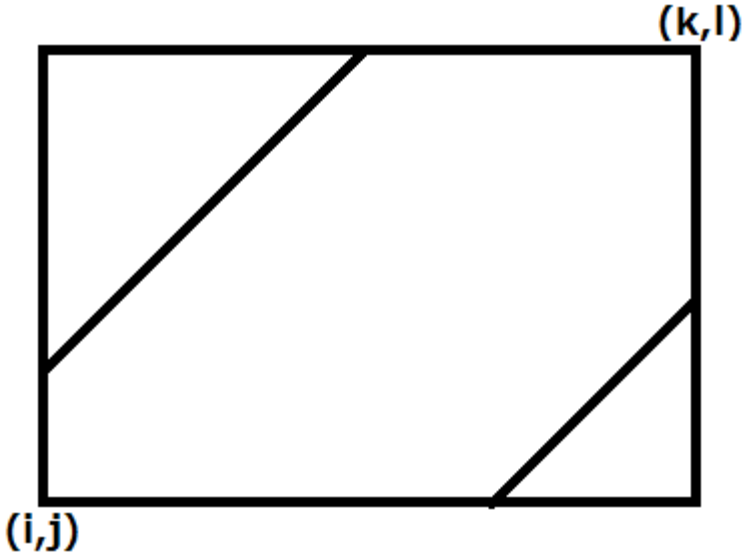
AtCoder

- 更新は次のようになる
  - dp[i]=(スタートから点iまで(障害物を考慮せず)行く方法の個数) – Σ(dp[j]・(点jから点iまで(障害物を考慮せず)行く方法の個数))
  - 地点iに到達する方法から地点iの前にすでにどこかに訪れてしまった方法を引いていると考えると分かりやすい
- このパートの計算量は $O(N^2)$

G問題　アルゴリズム

AtCoder

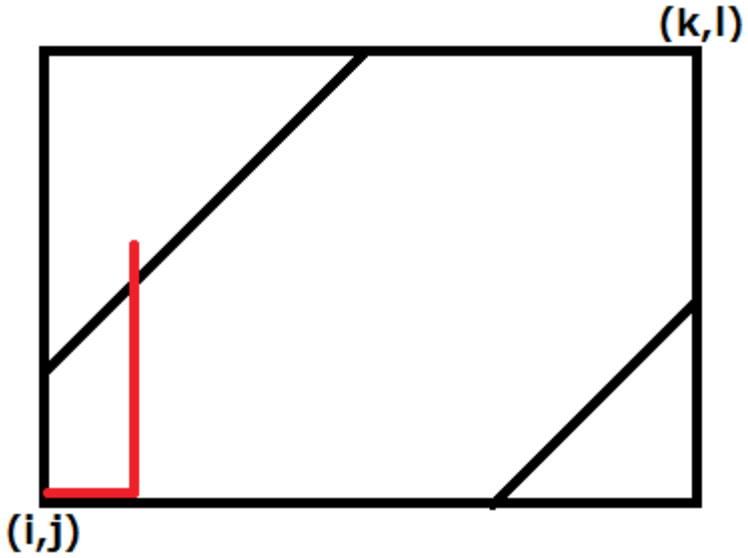
- これで2つの大小関係を満たしながら(i,j)から(k,l)まで移動する経路の個数を求められればよくなった。
- これは下図のような経路である。

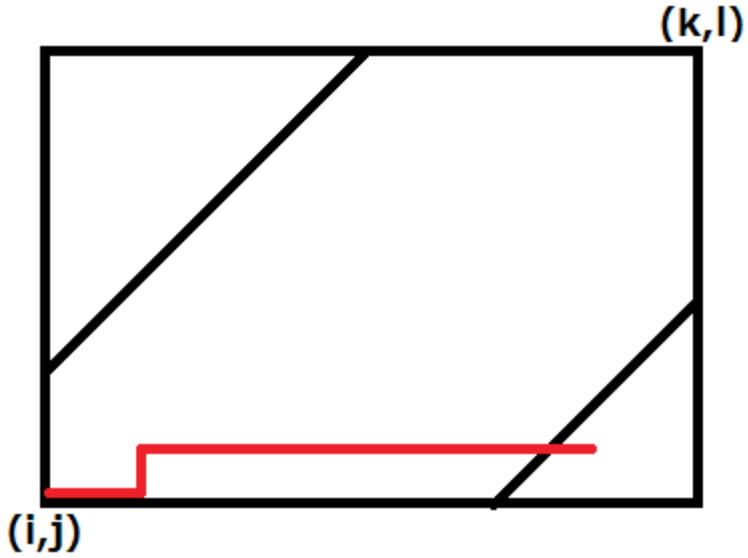


G問題　アルゴリズム

AtCoder

- 次のようなはみ出る場合を取り除きたい



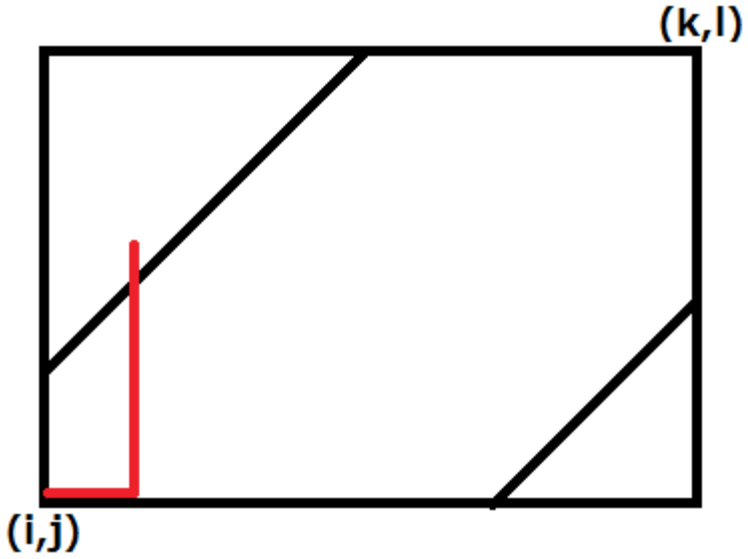


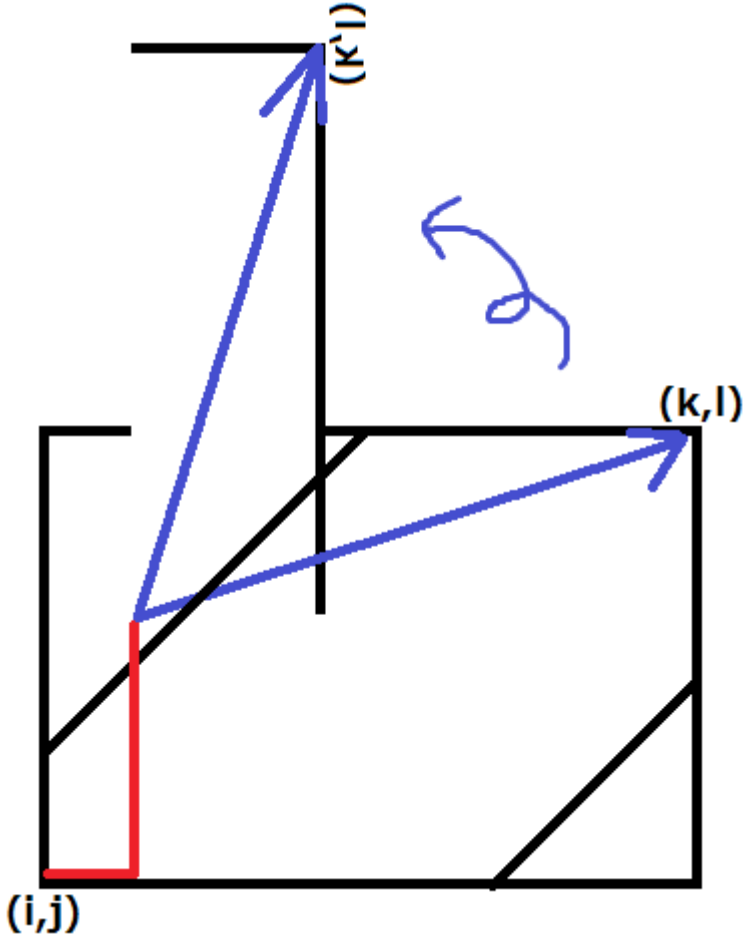
左のケースをとりあえず考える

G問題　アルゴリズム

AtCoder

- 次のようなはみ出る場合を取り除きたい





はみ出た先から目的までの経路を折り曲げる
→上にはみ出るケースに1対1対応する

G問題　アルゴリズム

AtCoder

- 右にはみ出るときも同様
- ただし、どちらにもはみ出るというケースを重複して取り除いていることになる！
- そこで先ほどと同様に「上にはみ出てから右にはみ出る」「右にはみ出てから上にはみ出る」という方法の数を足す
- また重複が出るので「上、右、上」「右、上、右」を引く…… →繰り返すとこのような重複が0通りになるところに行き着く

G問題　アルゴリズム

AtCoder

- 後半のパートの計算量は、重複を取り除く回数に依存する
- 1段進むには、逆のほうにはみ出す必要がでてくる。これは最低3回動く必要がある
- 全部で A+B 回動くので、O(A+B)でこれを計算できる
- 前のパートと合わせると $O(N^2(A+B))$
  - (実は重複を何度も取り除く必要があるときは動ける範囲が細長いので愚直なDPで解けるため、ほんの少し計算量は改善できる。)

H問題　Happy 2015

1. 問題概要
2. アルゴリズム

H問題　問題概要

AtCoder

- 1次元の数直線上においてN個の閉区間 [L[i],R[i]] が与えられる。
- これらの 2^N通りの選び方を試したとき、一つ以上の区間で覆われる区間の集合は何種類あるか？
- $1 \leqq N \leqq 2{,}000$

H問題　アルゴリズム

AtCoder

- まずは座標圧縮をする(すると、座標の最大は4000程度になる)
- 次のようなDPの漸化式を考える。
- dp[i]:= 座標i-1が直近の区間の右端となり、開区間(i-1,i)は照らされていない種類の数
- $dp[i]=\Sigma_{(j\sim i-1\text{の区間のみを照らせるj})}(1+\Sigma_{k<j}dp[k])$
- 累積和を使うことで、この更新は$O(N^2)$で可能

H問題　アルゴリズム

AtCoder

- それぞれの頂点の対に対して、1次元の数直線全体のうちその対の間(両端を含む)のみを照らすことができるかどうか、が高速に判定できれば先ほどの漸化式で $O(N^2)$で解くことができる

H問題　アルゴリズム

AtCoder

- 以下の方法で、計 $O(N^2)$でこれが求められる
  - 左端を決める
  - 「ここまでは埋まっている」という場所を持っておく(最初は決めた左端と同じ場所)
  - 右端の候補を左から順に見ていく。「ここまでは埋まっている点」とその場所の間に始点があり、その場所終わる区間があれば可能であり、かつ「ここまでは埋まっている」場所をその右端に更新できる

H問題　アルゴリズム

AtCoder

- この方法で、判定部分に累積和を使うとそれぞれの左端について O(N)で、すなわちO(N^2)
  全ての区間に対して O(N^2)で可能かどうかを決めることができる。
- 結果的に、全体で O(N^2)でこの問題を解くことができる。

I問題　Implementation Addict

1. 問題概要
2. アルゴリズム

I問題　問題概要

AtCoder

- N日間でなるべく多くの問題を解く
- 連続して問題を解き続けるとパフォーマンスが落ちる。具体的には i日連続で解いたときに解ける問題数は max(0,A-(i-1)B) 問
- すでに解かないと決めた日がM日ある
- 解く日解かない日を決めたときに解ける問題数の最大数は？
- $N \leqq 10^9$, $M \leqq 100{,}000$

I問題　アルゴリズム

AtCoder

- まず、あらかじめ決められた休日によって分断されリセットされるので、それぞれの休日ごとに独立に考えることができる。
- このようにして分けた区間内では、なるべく問題を連続して解く日数は均等にしたほうがよい
  - t日連続, t+2日連続して解くと解ける問題数よりも t+1日連続を2回繰り返したほうが解ける問題数が多くなる (前者での t+2日連続した日が後者での t+1日連続した日に対応)

I問題　アルゴリズム

AtCoder

- すると、一つの区間に休みを何回入れるかを決めると、O(1)で解ける問題数を計算できる。
- さらに、休みの回数が√(日数)より多いときは、代わりに連続して解く日数の最大の長さを決めても O(1)で解ける問題数を計算できる
  - 均等に割り振ったときに解く連続する日数が同じとなる休日数の区間では解ける問題数が休日の数に関する一次関数になっているので、両端を見ればよい

I問題　アルゴリズム

AtCoder

- 以上のことから、すでに決められた休日で分断された区間の長さを X[0], X[1],…, X[M] とすると、先ほどの方法では全体で計算量がO(Σ√X[i])となる。これが最も大きくなるのはX[0],…,X[M]が均等なときで、ΣX[i]=O(N)であることから、全体での計算量は、O(√(MN))
  - →これでも間に合う

I問題　アルゴリズム

AtCoder

- さらに、もう少し解ける問題数の式を考えてみると、解ける問題数は休日の数に関して凸であることが分かる。
- すなわち、三分探索を用いることでも正解できる。(より高速な解法)

J問題　Jungle

1. 問題概要
2. アルゴリズム

J問題　問題概要

AtCoder

- N 本の木が与えられる。それぞれの大きさは t[i]である。
- このうち最大M本を切ることができる。切ると大きさが0になる。ただし、連続するK本の中で切られた木が2つ以上あってはいけない。
- 連続するK本の大きさの合計の最大値を最小化せよ。
- 1 ≦ K ≦ N ≦ 100,000 , 1 ≦ M ≦ N

J問題　アルゴリズム

AtCoder

- この問題を考えるにあたって、これから次のような状態を考え、これを扱っていくことにする。

現在i番目の木を見ていて、最後に切られたのが
i-K本目であるため、(個数制限を気にしなければ)
i番目以降の木は切ることが可能である。

J問題　アルゴリズム

AtCoder

- 解法アイデア①
- K個の和の最大値で二分探索をする。
- 必要な切る木の本数がM本以下ならばよい。
- dpでこの切る木の最小本数を求める。
- dp[i]:=(先ほどの状態のi本目)で必要な切る木の本数の最小値

J問題　アルゴリズム

AtCoder

- 解法アイデア①
- i+j(0≦j<K)番目の木を切ることを考えると、dp[i]からdp[i+j+K]に対して配るDPで更新することができる。
- 切らないときも同様に更新できる。
- また、コストを求めるときにスライド最小値をdequeを使って求めることで、これは線形で更新ができる。

J問題　アルゴリズム

AtCoder

- 解法アイデア①
- 計算量は、
  - 二分探索の分 O(log (ans))
  - 配るDPをする部分。O(NK)
- であり、全体で O(NK･log(ans)) となる。
- Kが大きいときに間に合わないので、Kが大きいときは別の方法を考える必要がある。

J問題　アルゴリズム

AtCoder

- 解法アイデア②
- Kが大きいときを考える。
- 実は、Mは(N+K-1)/Kより大きくても使い切れない。
- すなわち、このときMが小さくなる。
- Mの大きさに依存したDPを考えることにする。

J問題　アルゴリズム

AtCoder

- 解法アイデア②
- dp[i][j]:=i本すでに切っていて、j本目で先ほどの状態になったときのそれまでのK個の連続する大きさの和の最大値の最小値
- この更新は、iそれぞれに対し、最初から見てきたときのK個の連続する大きさの和の最大値を変数に持っておき、適宜dp[i-1][j]の値とのminを取る形でこの変数を更新していくことで、合計で線形で計算することができる。

J問題　アルゴリズム

AtCoder

- 解法アイデア②
- こちらの場合も、コストの計算でスライド最小値を使う。これもdequeで線形で計算できる。
- 全体として計算量は、O(MK)となる。

J問題　アルゴリズム

AtCoder

- 解法アイデア①と②をまとめる
- この問題は O(NK･log(ans)) もしくは O(NM)、すなわちO(N^2/K)で解くことができる。
- K^2 = N/log(ans) となる程度のKを境に解法①、②を使い分けることで、最もよくなる。
- K=50で使い分けるとすると、
  - 前者は 3000N=3億程度
  - 後者は N^2/50=2億程度　となり、間に合う。