AtCoder Regular Contest 021
解説

AtCoder

AtCoder株式会社

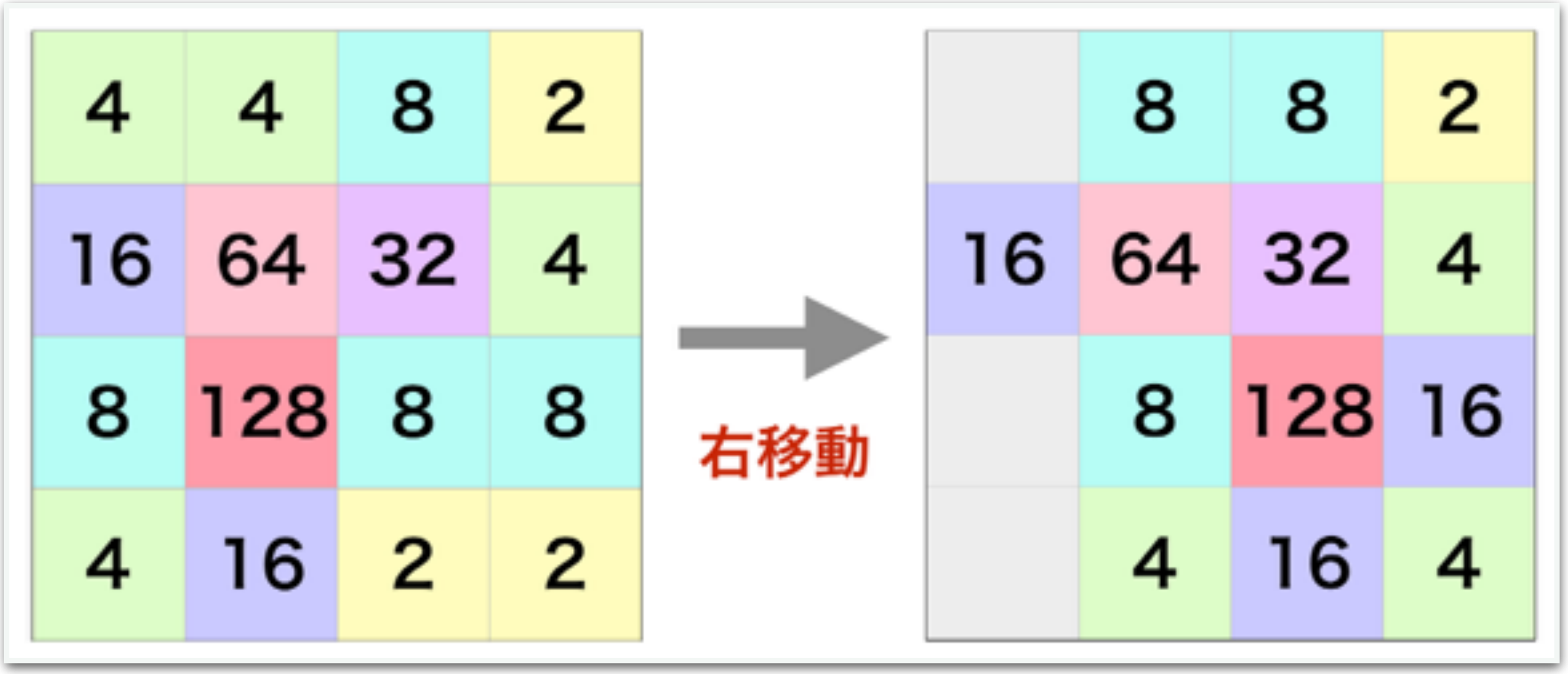
A問題

DEAD END

A - 問題概要

問題

タイルを滑らせることで同じ数をぶつけて
大きい数にしていくゲームの盤面が与えられる。
ゲームオーバーの状態かどうかを判定せよ。



A - 重要なポイント

与えられる盤面では**空いているマスはない**！

$A_{r,c}$ は 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 のいずれかである。

↓

同じ数をぶつけるような操作が必要

しかも、**隣り合う数**をぶつけるしかない

A - 解き方

整理すると……

上下左右に隣り合う同じ数が ある　→　**`CONTINUE`**

ない　→　**`GAMEOVER`**

実装

・**2 重ループ**を回す（添字に注意）

・if 文を書きまくって全部調べる（列挙漏れに注意）

(きちんとループで書けるようになっておいた方がいい)

ARC #021 B 問題解説

解説:城下 慎也 (@phidnight)

問題概要

- N 個の非負整数 $B_1, B_2, \ldots, B_N$ が与えられる。
- $B_i = A_i \hat{} A_{i+1}$(ただし $A_{N+1} = A_1$) である。

(以降、^ を排他的論理和(xor) の記号として表す。)

- $B_1, B_2, \ldots, B_N$ として考えられるものが存在するなら辞書順最小を求めよ。ないならないことを指摘せよ。
- $2 \leqq N \leqq 100{,}000$

排他的論理和の性質(例)

- A, B, C を非負整数とすると
- 結合則$(A \hat{} B) \hat{} C =$ $A \hat{} (B \hat{} C)$ が成立する。
- 交換則$A \hat{} B =$ $B \hat{} A$ が成立する。
- $A \hat{} 0 =$ A (零元)
- $A \hat{} A =$ 0 (それ自身とxorすると零元になる。)
- これらの性質を利用してこの問題を考える。

考察

- A_1を固定して考えてみる。
- B_1 =A_1^ A_2よりA_2=A_1^ B_1

(∵ A_2=0^A_2=(A_1^A_1)^A_2=A_1^(A_1^A_2)=A_1^ B_1)

- 同様に、B_2 =A_2^ A_3よりA_3=(A_1^B_1)^B_2

(∵ A_3=A_2^B_2=前述のA_2を代入)

- 一般にA_i=A_1^B_1^B_2^…^B_{i-1} ($i \geqq 2$)が成立することを示す。

考察

- A_i=A_1^B_1^B_2^…^B_{i-1} (i ≧ 2)が成立する。

(∵A_i=(A_1^A_1)^(A_2^A_2)^…^(A_{i-1}^A_{i-1})^A_i

=A_1^(A_1^A_2)^ (A_2^A_3) ^…^(A_{i-1}^A_i)

=A_1^B_1^B_2^…^B_{i-1})

- 以上のことより、A_1が定まれば、A_2からA_Nまでのすべての値がB_1からB_{N-1}までの値を用いて、一意に定まることがわかる。

解の存在判定について

- 使用されなかったB_Nについては、先ほどの計算で出てきた値とB_N= A_N^ A_1が矛盾する場合がある。
- A_N^ A_1= B_1^ B_2^…^ B_{N-1}となるのでB_NはA_1に依存せずこの値に一致するはずである。
- 矛盾したならば解が存在しない。
- 矛盾しないなら、A_1を任意に指定した後、解が一意に定まる。

辞書順最小

- 解が存在した場合に、辞書順最小なものを求める必要がある。
- 先頭の値 (A_1) ができるだけ小さいほうが良い。
- 今回の場合では、A_1=0のケースがただ 1 通りだけ存在するので、その1 通りが辞書順最小の値となる。(A_2以降を比較する必要はない。)
- これを計算して答えが得られる。

まとめ

- B_N=B_1^ B_2^…^ B_{N-1}が成立しないなら解なし、成立するならA_1= 0と置いて順に解を計算する。
- 排他的論理和の性質を利用した問題は様々なものがあるので、排他的論理和の問題には慣れておこう!

ARC #021 C 問題解説

解説:城下 慎也 (@phidnight)

問題概要

- 高橋君は建物を K 回増築したい。
- 高橋君は N 軒の建物を所有している。
- 増築には、増築する建物について、現在の価格だけ費用がかかり、増築で一定値値上がりする。
- 合計費用として考えられるものの最小値はいくらか。
- $1 \leqq K \leqq 100{,}000{,}000$
- $1 \leqq N \leqq 100{,}000$

部分点解法 その1

- 毎回どの建物を増築するかをN 通りから選ぶ解法だと O(N^K) 通りになってしまう。
- 考えてみると、異なる建物同士の増築の順番を考慮しなくてもよいことが分かる。
- つまり、「建物1 を増築しまくる」→「建物2 を増築しまくる」→… という順番を付けてもOK。
- すると、dp[i][j]=(i 番目の建物までの増築が完了したときに、今まで j 回増築している場合の最小費用) とする DP を組めば O(K^2N) で計算することができる。これで 30 点が得られる。

部分点解法 その2,3

- 実は、毎回「現在の費用が最小なものを増築し続ける」という貪欲法で正解を得ることができる。
- 以降の数枚でそのことを示す。
- この貪欲法を毎回すべて見る方針で実装すれば O(KN) で解ける。計40点が得られる。
- priority_queue を使用すればO($K \log N$) となり、通算55点が得られる。

貪欲法が適用可能なことの証明

- 建物の増築回数と価格を照らしあわせた表は例えば次のようになる。

ID	A	D	0 回増築	1 回増築	2 回増築	3 回増築
1	50	40	50	90	130	170
2	40	80	40	120	200	240
3	70	50	70	120	170	220
4	60	60	60	120	180	240

貪欲法が適用可能なことの証明

- 建物の増築回数と価格を照らしあわせた表は例えば次のようになる。

ID	A	D	0 回増築	1 回増築	2 回増築	3 回増築
1	50	40	50	90	130	170
2	40	80	40	120	200	240
3	70	50	70	120	170	220
4	60	60	60	120	180	240

- ここから K 個良いものを (途中が飛んでも良い) 選ぶアルゴリズムは貪欲法で実装できる。

貪欲法が適用可能なことの証明

- 先ほどの貪欲アルゴリズムは左詰め。というのも、同じ ID 内で、X 回増築の部分を採用しないで X+1 回増築の部分を採用しているなら、代わりに X 回増築の部分を採用すればより費用が下がる。
- 左詰めに採用されているのであれば、これは今回の問題で実行可能。
- この値より下回る解は存在しないので、これが最適解となる。

満点解法に向けて

- この方針では $1 \leqq K \leqq 100{,}000{,}000$ という制約に対して充分高速でない。
- 今までの手順をどうにかして高速化したい。
- 更に考察してみよう!

アプローチを変えてみる

- 貪欲法で最後に選んだ増築の費用をVとしてみる。
- 費用がUの増築に対して、
- $V > U$なら、その増築は必ず行われる。
- $V < U$なら、その増築は決して行われない。
- $V = U$なら、合計がK回となる分だけ増築が行われる。

満点解法

- 先ほどのVを考えてみると、適当にV'を定めたとき、V'未満、V'より大きい、V'に等しい費用の増築がそれぞれ何個あるかを計算すれば、VとV'の大小関係が分かる。
- よって、Vを二分探索することで計算することができる。
- Vが分かれば、V未満の増築(M個とする)をすべて採用し、Vと等しい増築を$K-M$個採用すれば最適解を計算できる。
- O($N \log L$) (LはVの上限) で計算できる。
- 100点が得られる。

D問題
だいたい最小全域木

D - 問題概要

問題

200 次元空間に 5 000 個の点がある。

2 点を結ぶコストを (1 - コサイン類似度) とするとき

できるだけコストの小さい全域木を求めよ。

最適なものの 1.01 倍のコストに収まれば OK。

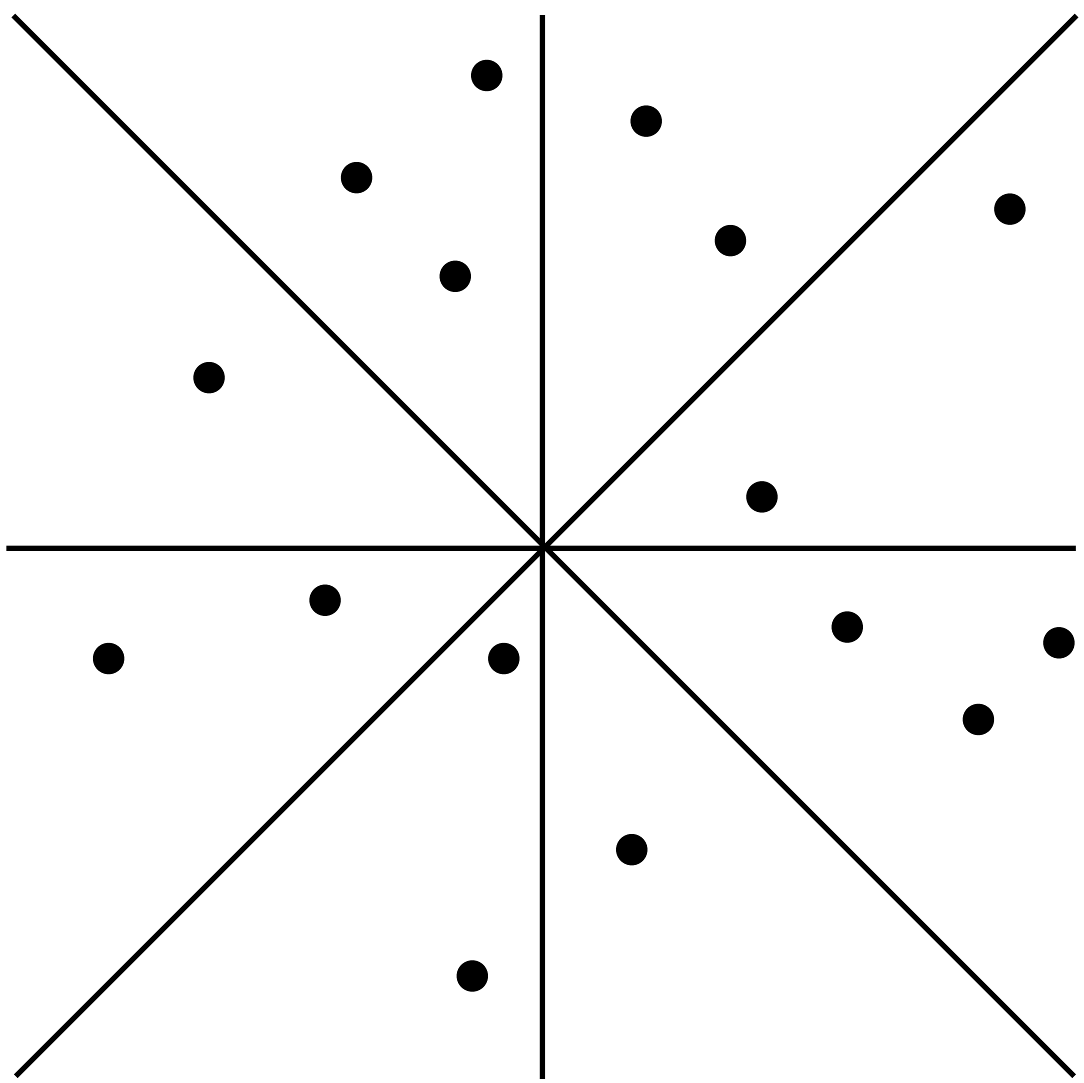
D - 方針

普通にやると……
問題自体は最小全域木だが、
全点間のコストを計算するのは時間がかかりすぎる。

最小じゃなくてもいい！
「最適なものの 1.01 倍のコストに収まれば OK。」
だった！
→ 必要そうな 2 点間だけ計算するようにしたい

D - 2次元の場合

この問題が 2 次元空間の場合は……



←こんな感じで
グループ分けして

同じか、近いグループの
点たちだけ見る

D - 200 次元でも同じことを

・ランダムなベクトルを K 個用意する
・各点とランダムベクトルの**内積が正か負か**で
　K ビットのハッシュのようなものが計算できる
・このハッシュで近いものだけを考えることにする

ハッシュ同士は**ハミング距離**(異なるビットの個数)
で距離を測れば爆速！！！！！

D - K の選び方

今回の場合は点がランダムに分布しているので

・**点の i 番目の座標の値が正か負か**

で 200 ビットのハッシュを作ると、
均等にグループ分けできて良いです。