Square869120 Contest #4 解説

中学3年の双子 E869120, square1001による有志コンテスト

コンテストの概要

- 問題数…8問（第1回～3回目まで、全てそう）
- 制限時間…3時間20分（第3回より長い）
- Writer…E869120,square1001（これは第1回から変わらないメンバー）
- rated…なし
- というわけで、Square869120Contest #4はお楽しみいただけたでしょうか。お 楽しみ出来たら幸いです。

問題 (Overview)

- A問題 → Atcoder Handles
- B問題 → Buildings are Colorful!
- C問題 → Calender 2
- D問題 → Driving on a Tree
- E問題 → Enormous Atcoder Railroad
- F問題 → Find the Route!
- G問題 → Guess the Salary of Atcoder
- H問題 → Huge Kingdom: Atcoder

A問題：Atcoder Handles

square869120Contest #4 解説

問題概要

- N個のハンドルネームの配列が与えられるが、そのうちいくつかの文字が隠されて ‘?’ で表されている
- 自分のハンドルネームも与えられる (これは隠されていない)
- それをソートした時に何番目になるか求める
- 複数通りあり得る場合は全部求める

- 制約
- $N \leq 10000$
- $|S_i|, |T| \leq 20$

小課題1 - 隠された文字が一つもない (130点)

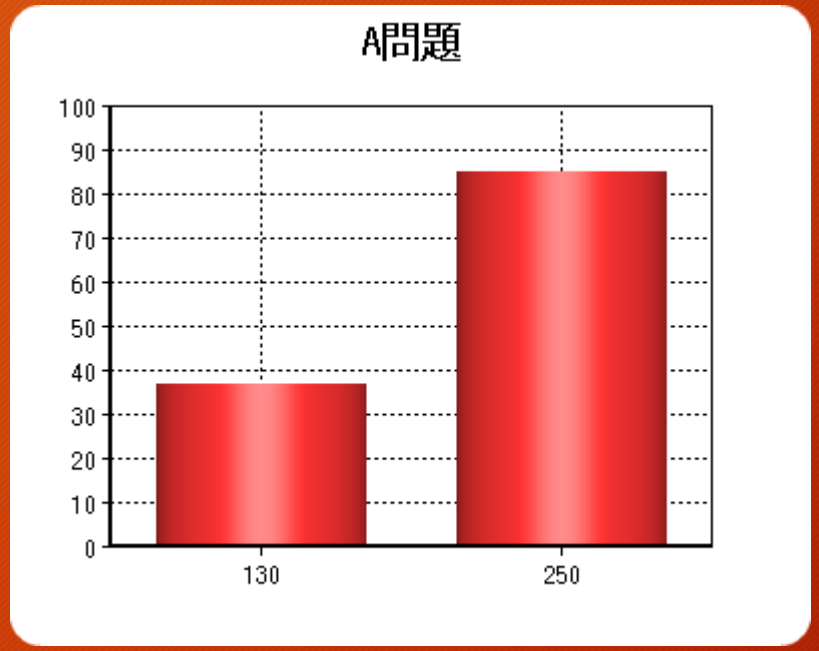
- リストに自分を含めたものの配列をソートする
- 自分のハンドルネームとあっているもののindexを全部出力すればOK
- 例：$S = \{\text{“}petr\text{”}, \text{“}tourist\text{”}, \text{“}rng\text{”}\}$ で、$T = \text{“}tourist\text{“}$ のとき、S に T を加えたものをソートすると $\{\text{”}petr\text{”}, \text{“}rng\text{”}, \text{“}tourist\text{”}, \text{“}tourist\text{”}\}$ となり、自分のハンドルネームと同じなのは3番目、4番目である。
- よって答えは3, 4となる。
- よってこの問題はソートすると $O(N \log N)$ で解ける。

小課題2 - 満点解法 (300点)

- 文字列 A, B (Aは '?' が含まれている可能性がある) に対し、'?' を埋めた時に $A \leq B$ となる可能性があるか、$A \geq B$ となる可能性があるかを考える
- $(S_1, T), (S_2, T), (S_3, T) \ldots, (S_N, T)$ の組しか比較しない、また T は '?' が含まれていないので、それぞれの結果の合計みたいなものを求める感じで範囲が求められる -> 答えの範囲が連続であることが証明できる！
- 左端は $S_i \geq T$ の可能性がある個数を使うと求められ、右端は $S_i \leq T$ の可能性がある個数を使って求められる
- 比較は全部 'a' を入れる、全部 'z' を入れると文字列の比較でできるので、$O(|S_i| + |T|)$ しかかからない
- よってこの問題は $O(N)$ で解けた

結果

- 250点：85人
- 130点：37人



B問題： Buildings are Colorful

Square869120Contest #4 解説

問題概要

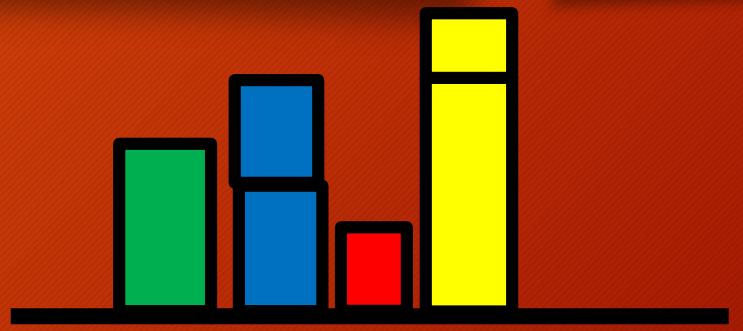
- N個の建物が並んでいます。
- 左からＩ番目の建物の高さは、最初 A[Ｉ] です。

- 【建物 i が見える条件】$a_j \geq a_i (j<i)$ となるような j が存在しない

- その時、左から見た時に K 個以上の建物が見えるようにするためには、最大で高さ何上げる必要がありますか？
- ただし、建物の高さを減らすことはできない。

問題概要

- 例えば、K=3, a={3,2,1,4}の場合、答えは3
- 右図のように上げれば良い
- 合計のコストは、2+1=3



- ≪制約≫
- 1≦N≦15
- 小課題・・・N=K [120点] , N≦5かつa[i]≦7 [90点]

小課題1 (120点)

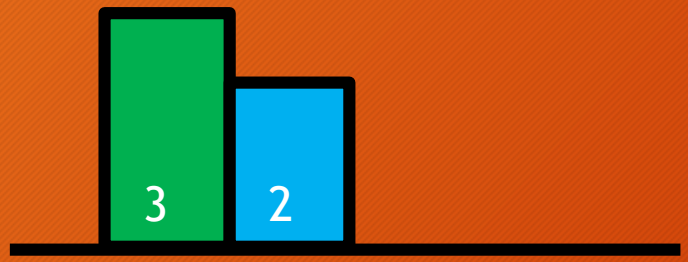
- 小課題1は、N=Kを満たします。
- そのため、すべての建物が見える必要があります。

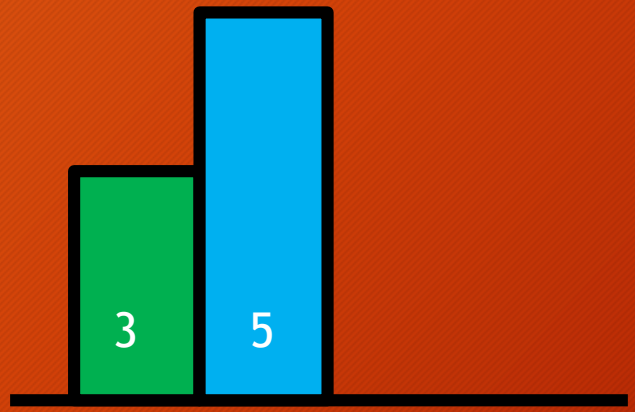
- さて、そのためにはコストが最小で何必要でしょうか？

- ≪ヒント≫ i番目の建物の高さが p のとき、次の建物の高さはp+1以上である必要があります。

小課題1 (120点)

- 例えば、以下の図を考えましょう。





- 左のパターンの場合、水色の建物を変える場合、高さ4に上げることが最適です。
- 右のパターンの場合、水色の建物を変える場合、高さを変えないことが最適です。

小課題1 (120点)

- よって、以下の2通りに場合分けできます。
- 前の建物の高さv ≧ 今の建物の高さ の場合、その建物の高さをv+1に上げる。
- そうでない場合、高さは変えない。

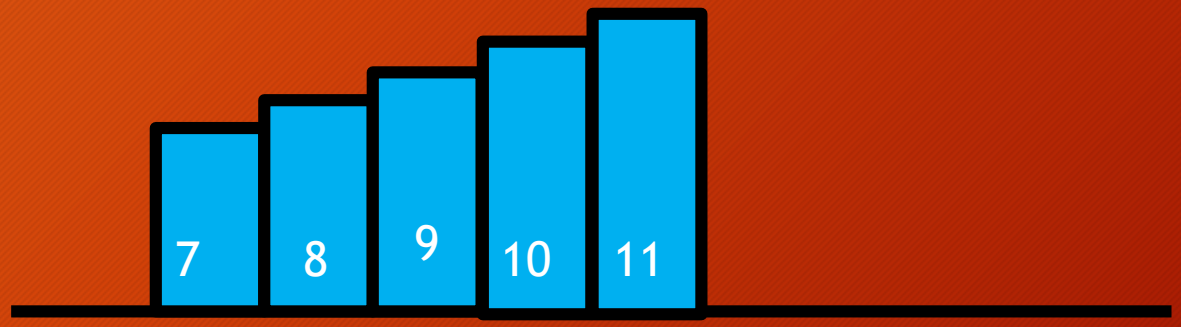
- それは、高さが低くなることによって後々損することがないことより、容易に証明できます。

- 左から見るので、建物を貪欲に見ていくときには左から見るようにしましょう。

小課題2 (90点)

- 今度は、N≦5, a[i]≦7です。

- どのような場合でも、高さを12以上にして最適になることはありません。
- 最大の場合でも、a[i]={7,7,7,7,7}のときで、最終的な高さは、{7,8,9,10,11} となります。



小課題2 (90点)

- というわけで、最終的な建物の高さとしてあり得るもの 11^5通りを全探索することを考えます。
- しかし、高さを減らすことはできないので、以下のような条件をみたすような場合の中で最小コストを求めることになります。
- ここでは、最終的な高さを b[i]とします。

- すべてのiについて、a[i]≦b[i]
- K個以上の建物が見える（これは問題文のとおりに実装すれば簡単）

- 計算量は、$11^5 * O(N^2) = O(N^2 * 11^5)$となり、間に合います。

考察1

- 今度は、N≦15, a[i]≦10億です。全列挙では到底間に合いません。
- そこで、以下のような考察をします。

- 結局見えないところは高さ変えても無駄なのではないか？

- ー見えてほしいところを全探索⇒2^N通り⇒間に合いそう！

小課題3 (140点)

- というわけで、以下のようなアルゴリズムで進めていきます。

- どこを見られるようにするか、を2^N通りで全探索（見えるところがK個以上のもののみ）
  - 左から順番に建物を見ていく
    - 見られないと決めたところは高さを変えない
    - 見られたいところは、（それより左の高さの最大値）+1に変える
    - ただし、今見ている建物の高さが高すぎる場合、高さを変えない場合がある
- そのように、貪欲に全列挙を行うことができます。

- 計算量は、$O(2^N * N)$⇒間に合います。

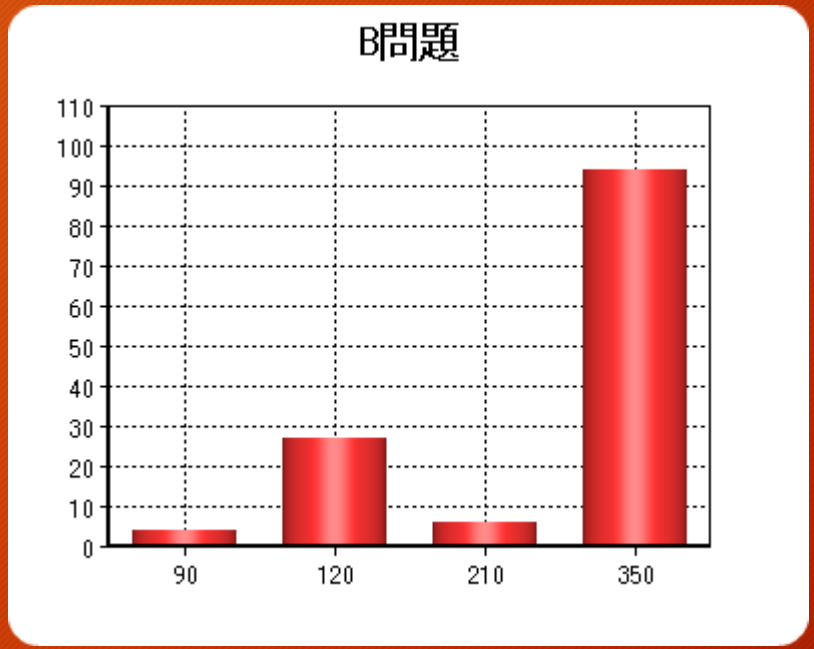
注意点

- long long を使うことに注意しましょう。
  - この問題では、答えの最大値が 150億程度となります。
  - int型では、約21億までしか値を表すことができません。

- 境界判定に気をつけましょう。
  - 今回の問題の場合、「…+1」などの境界を間違えてWAするというケースが考えられます。
  - それには注意しましょう。
  - 考察ノートなどに整理して問題を考えたりするとわかりやすいでしょう。

結果

- 350点者：94人
- 210点者：6人
- 120点者：27人
- 90点者：4人



C問題：Calendar 2

square869120Contest #4 解説

問題概要

- $n \times 7$ のマス目があり、各マス目は最初白くマス (i, j) には整数 $7i + j - 8$ が書かれている
- すぬけ君は、m で割った余りが $a_1, a_2, a_3, \ldots, a_q$ のマスを黒く塗る
- そのとき、白い部分の連結成分の個数は何個になるか?

問題概要

- 制約
- $1 \leq n \leq 10^{12}$
- $1 \leq q \leq m \leq 10^{5}$
- $7n$ は m で割り切れる

小課題1 (100点)

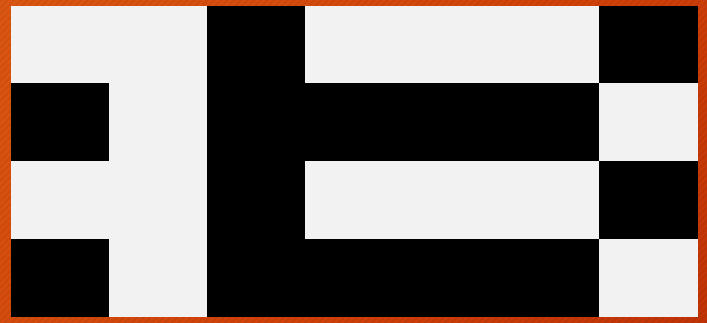
- 制約は $n \leq 100000$ なので、愚直にマスを黒く塗ってDFSで連結成分の数を求めれば答えが出てくる
- 計算量は $O(n)$

小課題2，3 (240点)

- m は7の倍数であるという制約がある
- $7n$ は m で割り切れるので、最初の $\frac{m}{7} \times 7$ の部分を $\frac{7n}{m}$ 個下につなげた形になる
- 1個、2個、3個… とつなげていったときに、何か規則性は見えないか?

小課題2，3 (240点)

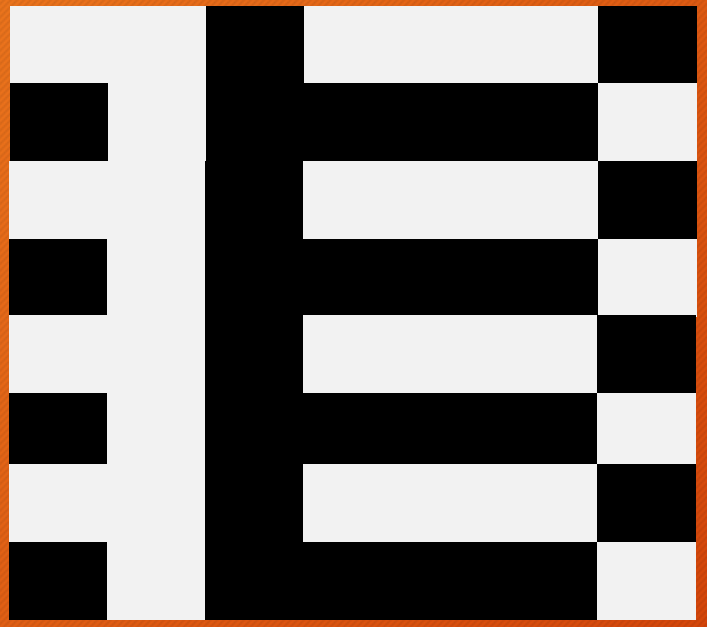
3個



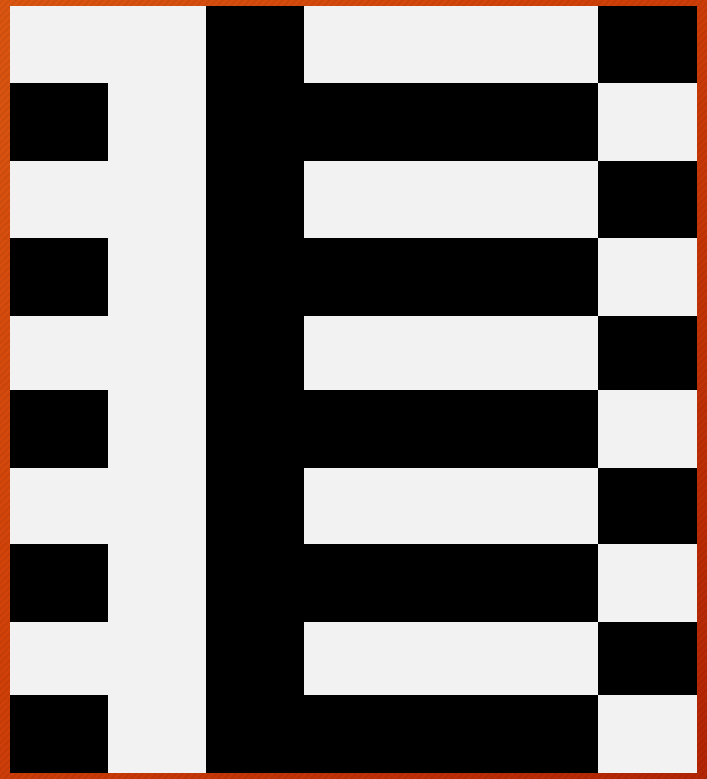
5個

7個

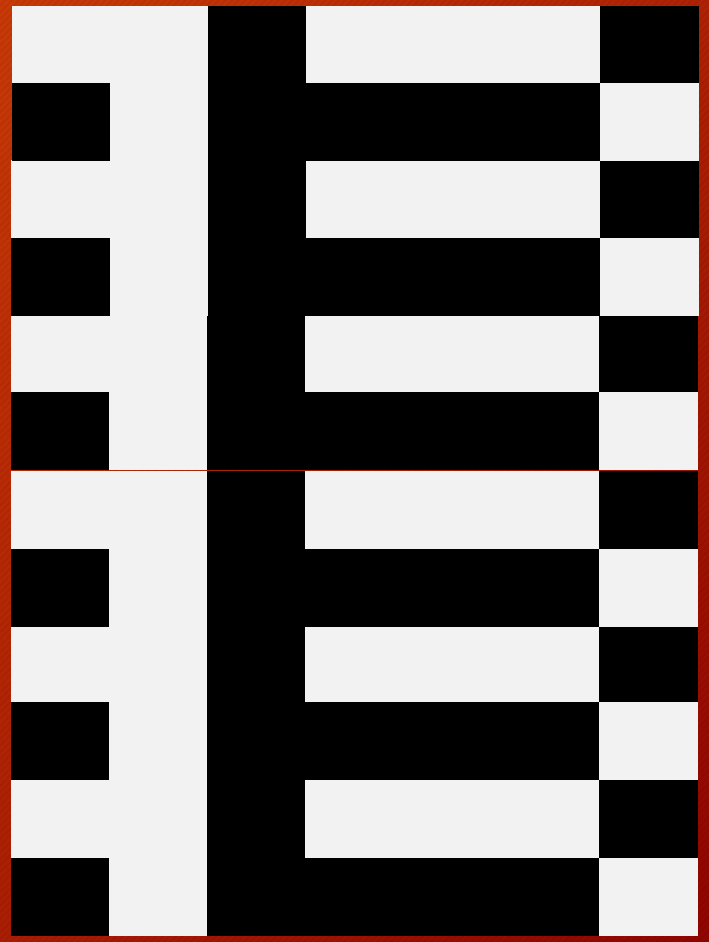
小課題2，3 (240点)



9個



11個



小課題2, 3 (240点)

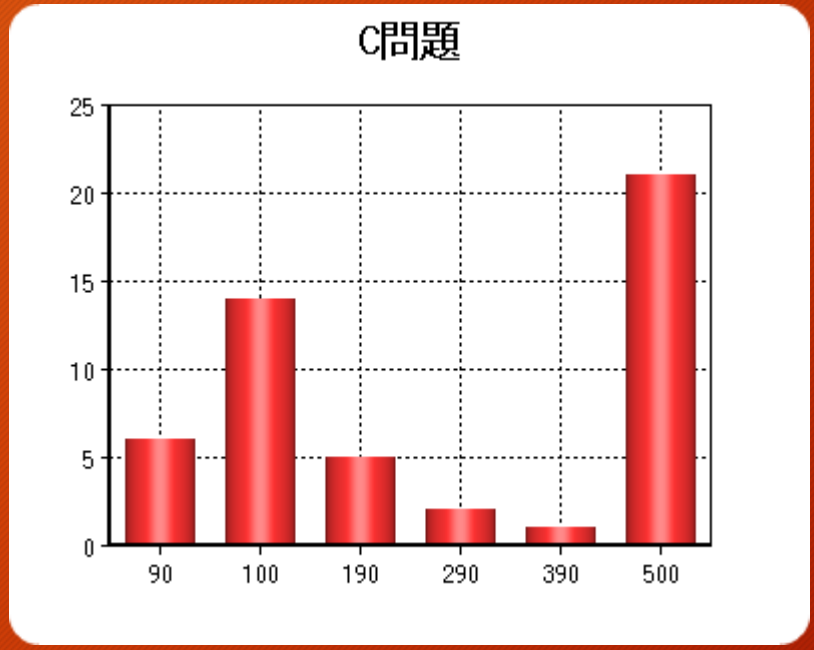
- x 個つなげたときに連結成分 $ax + b$ になる (ただし a, b は定数) 、またこれは証明できる (ここでは省略)
- よって、$x = 1, 2$ の場合が分かれば、a, b が分かるので、連結成分数も分かる
- 計算量は $O(m)$

満点解法 (450点)

- m が 7の倍数でない場合は、$x = 7$の倍数 の場合で計算すると m が7の場合に帰着できる
- よってこの問題は $O(m)$ で解けた

結果

- 500点者:21人
- 390点者:1人
- 290点者:2人
- 190点者:5人
- 100点者:14人
- 90点者:6人



D問題： Driving on a Tree

Square869120Contest #4 解説

問題概要

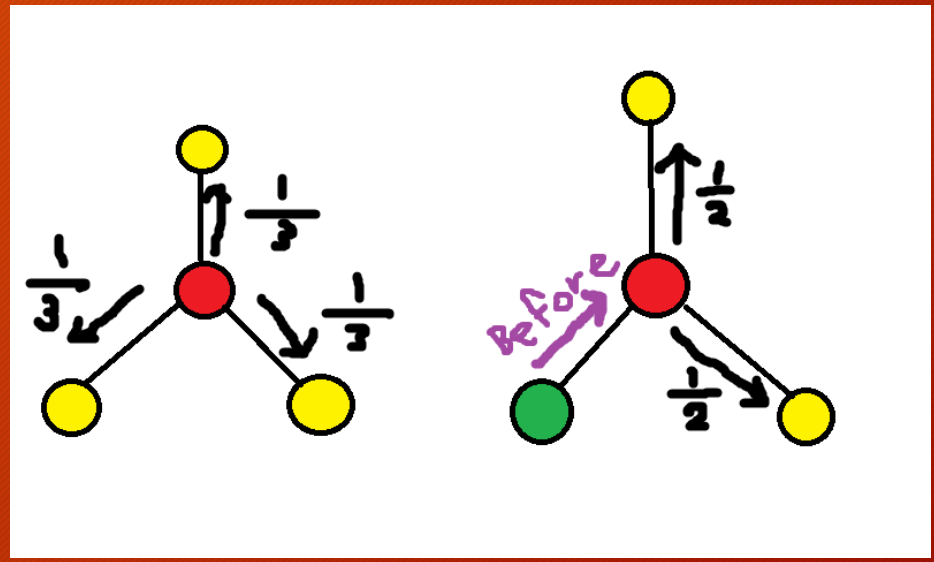
- 木が与えられます。
- うさぎは木上で以下のような操作を行います。
  - 隣り合った頂点に動く。しかし、U-ターンはできない。
  - U-ターンしないで動ける頂点がない場合、そこで操作は終了となる。
  - どこに動くかは等確率にランダムに選ぶ。例えば、次に動ける頂点がp個である場合、それぞれの頂点に1/pの確率で動くことになる。

- それぞれの出発する頂点についての、動く回数の期待値を求めなさい。

問題概要

- 右図のような木の場合、
  - 期待値は {2.0, 1.0, 2.0, 2.0}となる

- 制約
- 小課題1 [190点]: 線のような形である
- 小課題2 [220点]: N≦1000 を満たす。
- 小課題3 [390点]: N≦150000を満たす。

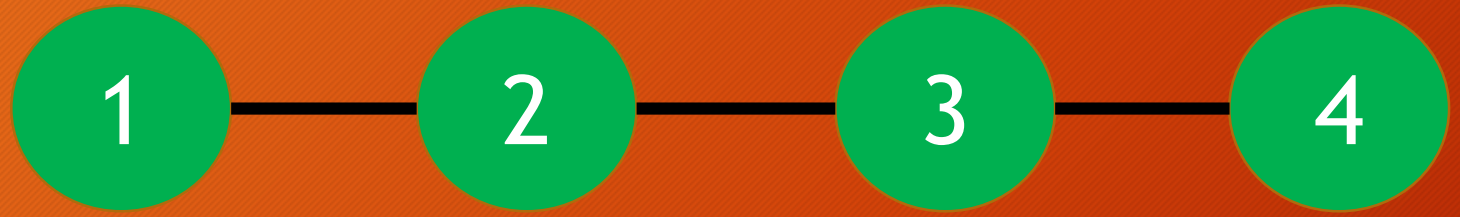


小課題 (190点)

- その制約は、木が線のようになっているという事です。
- さて、N=4, N=5の場合について考察してみましょう。
- どのような規則性が見いだせるでしょうか。

小課題1 (190点)

- N=4の場合



- 頂点1から ➡ 右に行く=3.0 期待値=3.0
- 頂点2から ➡ 左に行く=1.0 右に行く=2.0 期待値=1.5
- 頂点3から ➡ 左に行く=2.0 右に行く=1.0 期待値=1.5
- 頂点4から ➡ 左に行く=3.0 期待値=3.0

小課題1 (190点)

- N=5の場合

- 頂点1から ➡ 右に行く=4.0 期待値=4.0
- 頂点2から ➡ 左に行く=3.0 右に行く=1.0 期待値=2.0
- 頂点3から ➡ 左に行く=2.0 右に行く=2.0 期待値=2.0
- 頂点4から ➡ 左に行く=1.0 右に行く=3.0 期待値=2.0
- 頂点4から ➡ 左に行く=4.0 期待値=4.0

小課題1 (190点)

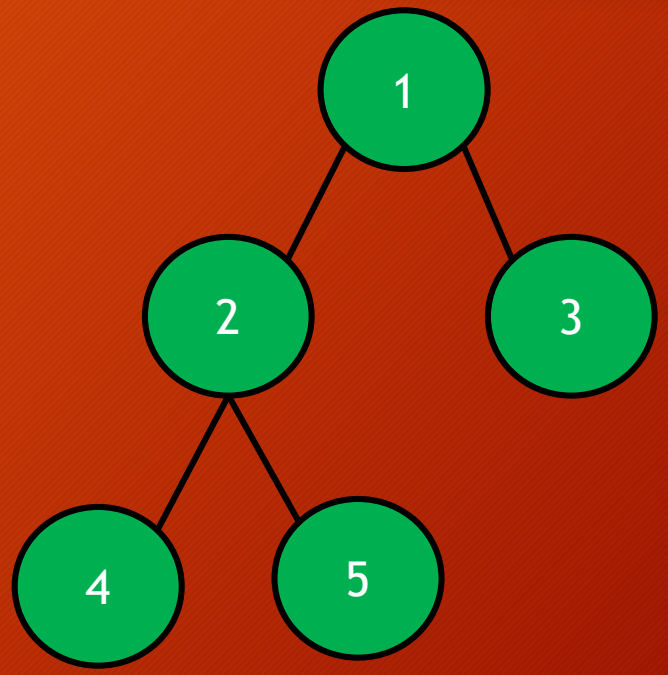
- 以上をまとめると、そんな感じになりそう
  - 端である場合、期待値は N-1
  - そうでない場合、期待値は (N-1)/2
- 端であるかどうかの判定は、次数が1か2か、という判定でできる
- ⇒O(N) 190点

小課題2 (220点)

- 今度は、一般の木の場合における期待値を求めたい
- 制約は N≦1000
- $O(N^2)$が間に合う、$O(N^2 logN)$も間に合うかもしれない

小課題2 (220点)

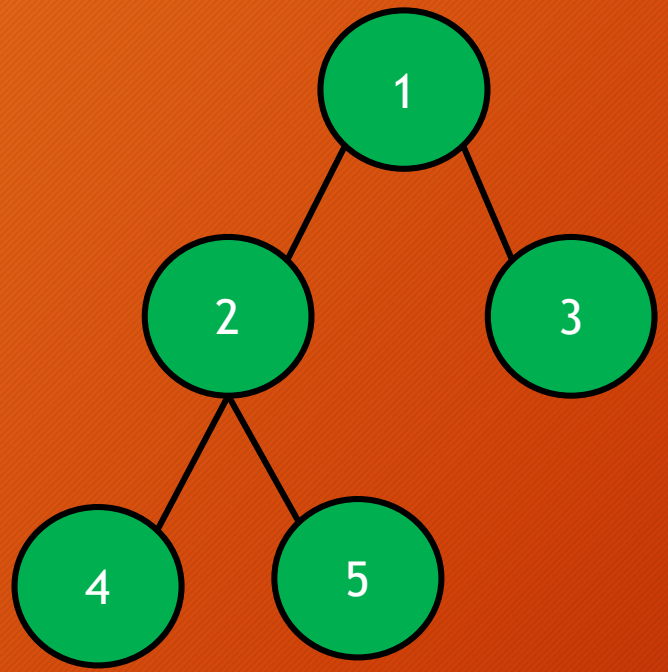
- そこで考えるのが、木DP
- スタート地点を根として木を作ることを考える。

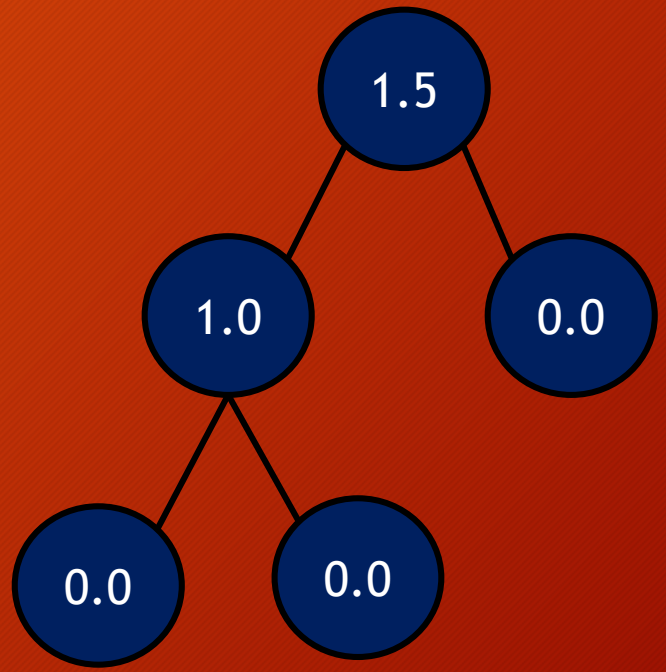


- 下から順に木DPをすることを考える。
- ⇒頂点 i の値は以下のようになる
  - $\mathrm{dp[i]} = \sum \frac{dp[\text{頂点}i\text{の子孫}]}{\text{頂点}i\text{の子孫の個数}} + 1$
    - ランダムにうさぎは動くので、その平均を使った式は成り立つ
    - U=ターンはできないので、上に行けないことは自明

小課題2 (220点)

- 例えば、その図の場合、答えは 1.5
- 最後の部分では、 (1.0+0.0)/2 +1.0 = 1.5という計算式





小課題2 (220点)

- それを全ての頂点について全探索
  - 頂点の数は N 個
  - 木DPにかかる時間: O(辺の数) = O(N)
- 全体計算量は$O(N^2)$ ➡ 220点

- 小課題1には N≧1001のケースもあるので注意すること。
- 小課題1についても場合分けすると、410点！！！！！

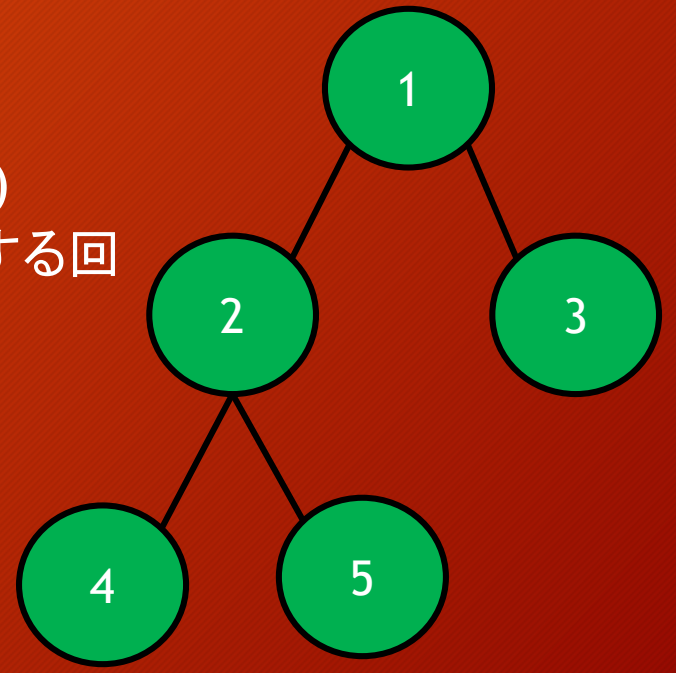
小課題3 (800点)

- 制約は、N≦150000
- いくら定数倍高速化しようとも、$O(N^2)$は間に合わない
- ⇒O(N) やO(N log N)などを考えなければならない



考察 1

- まず、頂点1が根であるようにDFSを行うことを考える
- その時、最初に下にしか行けない場合、全ての頂点についてO(n)で処理できる
- 一回下に行ってしまうともう一度上に行けない
  - ⇒小課題2の頂点1が根の場合と同じように木DPできる：O(n)
  - ここからは、dp[i] = 頂点 i からずっと下に行くことでの移動する回数の期待値　とする
- 問題は、最初に上に行く場合



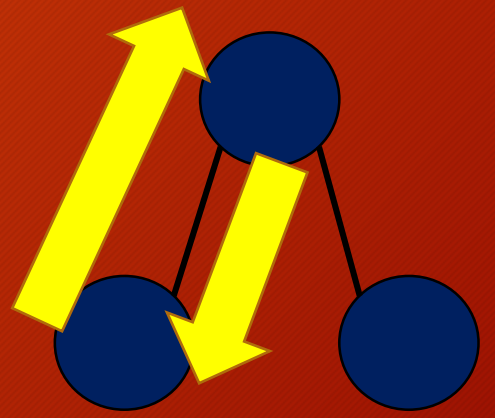
考察 1

- 最初に上に行く場合、どのように処理するか
- 上に行くので、前の結果を利用できそう　➡上から順にDPしそう

- とりあえず、dp2[i] = 頂点iから出発し、最初上に行く場合の期待値　と考える。
- dp2の結果を再利用できるテクを考える

考察1

- 頂点 i の直接の祖先を頂点 j とする
  - 根付き木に直接の祖先は 1 個しか存在しない
- 頂点 j から上に行く場合の期待値・・・ dp2[j] + 1
- 頂点 j から下に行く場合（切り替える場合）の期待値：dp[j] + 1



- ただし、U-ターン禁止なので、右図の場合に気を付けなければならない
  - 右図の場合の期待値は、dp[i] + 2
- よって、頂点 j の次数を p とすると、
- dp2[i]=$\frac{(dp2[j]+1)+(dp[j]+1)*(p-1)-(dp[i]+2)}{p-1}$ となる

小課題3 (800点)

- dp2を求める操作は、上から順に処理する
- ただし、dpが求まっていなければならないので、以下の順番で処理する必要がある。
  - dpを求める操作　➡ dp2を求める操作

- また、頂点 i での期待値は以下のようになる。
- $E = \frac{dp[i]*(p-1)+dp2[i]}{p}$
  - dpは下に行く場合の期待値、dp2は上に行く場合の期待値
  - pは自然数

小課題3 (800点)

- 計算量解析
- 頂点1を根としてDFSするのに O(N)
- dpを求めるのにO(N)
- dp2を求めるのにO(N)

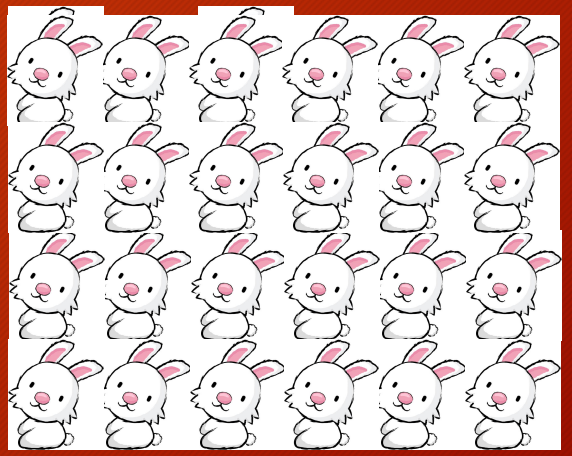
- 合計で O(N)+O(N)+O(N)=O(N)！！！！！！！！！！

小課題3 (800点)

- DFSを使って、上から、もしくは下から順番に処理する問題はほかにもたくさんあります。

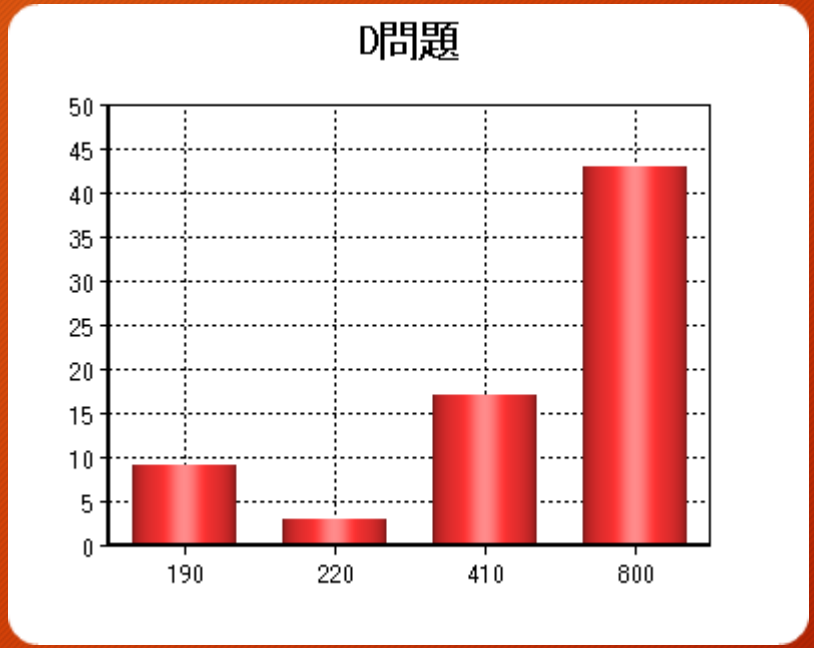
- ≪そのような問題≫
- AGC004：Teleporter / DFSを使います。
- JOI 2009 合宿：Advertisement / DFSを使います。

- その他いろいろ



結果

- 800点者:43人
- 410点者:17人
- 220点者:3人
- 190点者:9人



E問題：Enormous Atcoder Railroad

square869120Contest #4 解説

問題概要

- 駅がN+1個一直線上に並んでいる (それぞれの駅を0～N とする)
- 普通電車は駅 i と駅 $i+1$ の間を1分で移動する
- 急行電車は駅 S_i と駅 S_{i+1} の間を1分で移動する ($0 = S_0 < S_1 < \cdots < S_K = N$)
- 準急電車を新設する。駅 T_i と T_{i+1} の間を1分で移動するが、次のような条件を満たすように数列 T を決める必要がある
  - $0 = T_0 < T_1 < T_2 < \cdots < T_{|T|-1} = N$
  - 急行が停車する駅は必ず準急も停車する
  - 駅0からどの街にも X 分以内でいけるようにする
- そのとき、条件を満たす数列 T の個数は何通りか? ($mod\ 10^9 + 7$)

問題概要

- 制約
- $K, X \leq 2500$
- $S_{i+1} - S_i \leq 10000$

小課題1 (120点)

- 準急の止まる駅の集合は最大でも 2^{N-1} 個しかない
- => $N \leq 15$ なのでこれを全探索できる！

- 駅0からの距離は $N+1$ 頂点 $3N$ 辺以下のグラフに対しての最短経路問題なので、BFSすることによって $O(N)$ で解ける
- => 計算量は $O(2^N \times N)$ で解ける

小課題2 (210点)

- 制約は $S_{i+1} - S_i \leq 15$ => 急行と急行の間について全探索できそう！

- 考察1：急行の停車駅間での問題に帰着できないか?
- 例：$N = 13, K = 4, X = 4, S = [0, 4, 8, 11, 13]$ のとき

- その場合、駅1から9にすべて対して「駅0までの最短経路が4以下」または「駅10までの最短経路が3以下」の条件が満たされていればOK

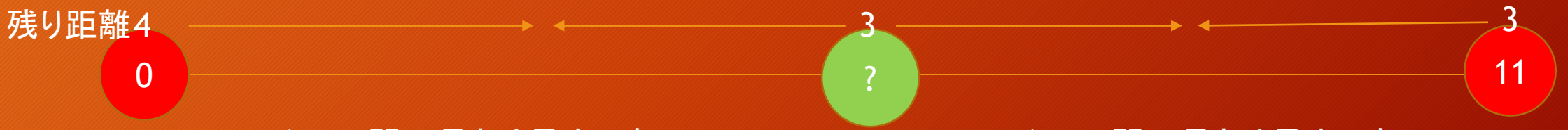
小課題2 (210点)

- あとはそれぞれの区間に対してかけ算すればよいので、全体の計算量は $O(2^{S_i - S_{i-1}} \times (S_i - S_{i-1}) \times K)$ くらいでできる
- $K \leq 15$ であるので、この問題は時間制限に間に合う計算量で解くことができる

小課題3 (470点)

- $K \leq 40, S_i - S_{i-1} \leq 40$ => 多項式で解ければよさそう
- =>全探索はできなさそう、動的計画法が使えるか?

- 考察2:各区間に対して準急の停車駅数を固定したとき求められるか?
- 両端が残り距離4、残り距離3とすると、この区間の準急が3つのとき残り距離が4, 3, 3 となる



その区間の長さは最大でも8
(4, 3, 2, 1, 0, 0, 1, 2, 3)

その区間の長さは最大でも7
(3, 2, 1, 0, 0, 1, 2, 3)

小課題3 (470点)

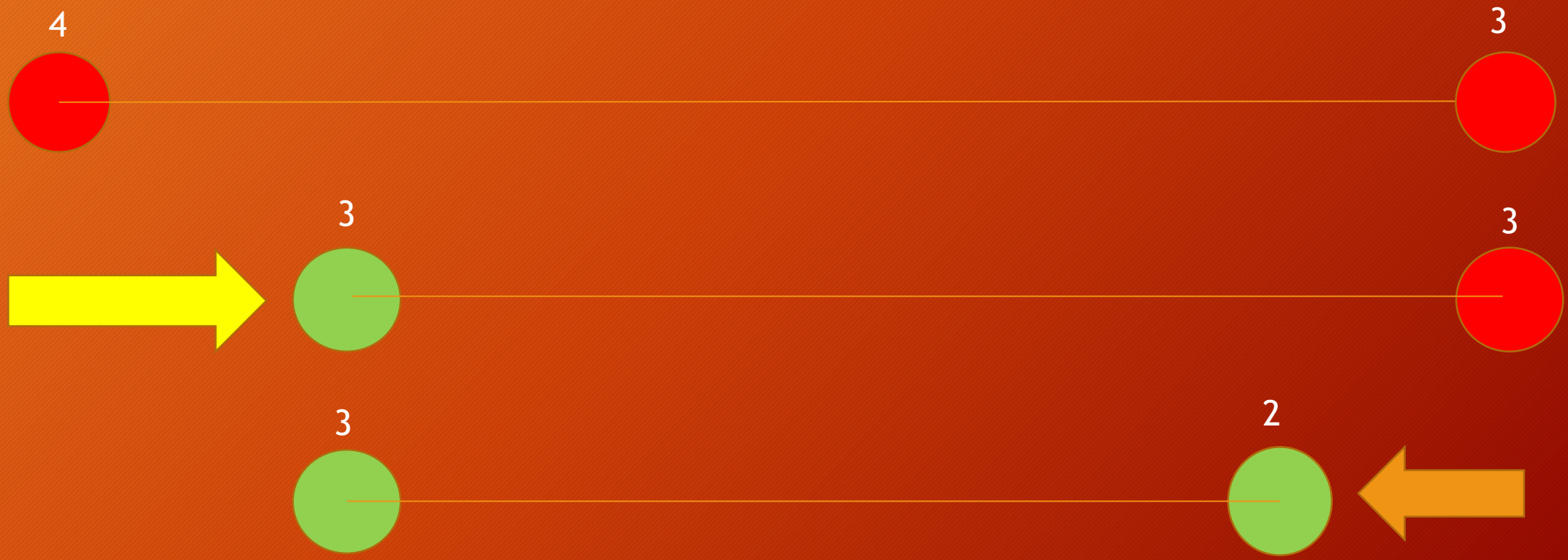
- よってこれは単純な部分和問題に帰着できる
- 準急の停車駅数がを L とするときこの問題は動的計画法と累積和を用いて $O\left(L(S_{i+1}-S_i)\right)$ で解ける
- 準急の停車駅数は $S_{i+1}-S_i$ 通り程度あるので、一つの区間に対して $O\left((S_i-S_{i-1})^3\right)$ の計算量で解ける
- 区間は K 個あるので、計算量 $O(K(S_i-S_{i-1})^3)$ で解ける (累積和を使わなくても4乗になって間に合うので解ける)

満点解法 (1000点)

- $N, K \leq 2500, S_i - S_{i-1} \leq 10000$
- もう少し工夫はできないか?

満点解法 (1000点)

- 考察3:



満点解法 (1000点)

- 考察3:

満点解法 (1000点)

- このように遷移すると重複なしで高速に求められる！
- $dp_{i,j}$ = 両端の残り距離の和が i , 両端の間の駅数+1を j とすると、次のような漸化式ができる
- $dp_{0,1} = 1, dp_{i,j}\ (j \leq 0) = 0$
- $dp_{2i,j} = dp_{2i-1,j-1} + dp_{2i-1,j-2} + \cdots + dp_{2i-1,j-2i} + int(j \leq 2i+1)$
- $dp_{2i+1,j} = dp_{2i,j-1} + dp_{2i,j-2} + \cdots + dp_{2i,j-2i-1} + int(j \leq 2i+2)$
- ここでは, int(trueまたはfalse) をtrueの場合1、falseの場合0をとる関数とする。

満点解法 (1000点)

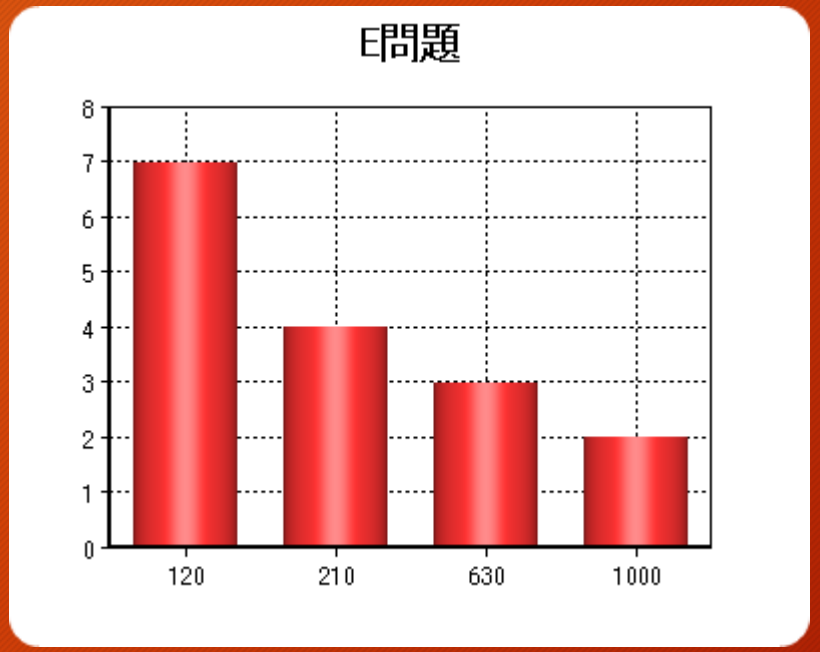
- こうすると、使う準急の駅数が決まっていなくても簡単に求めることができる
- この方法でやると、DPテーブルは、$O(X(S_{i+1}-S_i)^2)$ で求めることができる
- 累積和を使うと、$O\left(X(S_{i+1}-S_i)\right)$ で求めることができる

満点解法 (1000点)

- 各区間に対して愚直に求めていくと、$O\left(XK(S_{i+1}-S_i)\right)$ で解くことができる => これでは間に合わない！
- DPテーブルを前処理で求めておくと各区間につき $O(1)$ で判定できるので、計算量は $O(X(S_{i+1}-S_i)+K)$ となり、計算量に間に合わせることができる

結果

- 1000点者:2人
- 630点者:3人
- 210点者:4人
- 90点者:7人



F問題： Find the Route!

Square869120Contest #4 解説

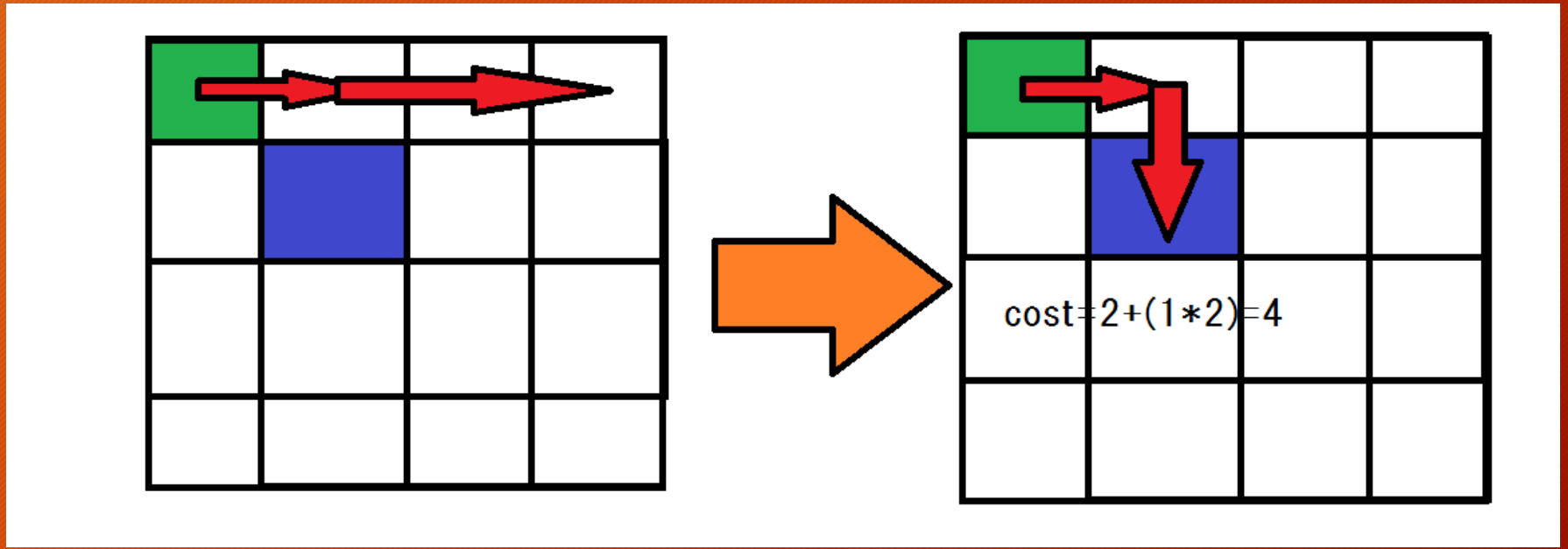
問題概要

- H*Wのグリッドがあります。
- N個の矢印があります。
- E869120という者は、場所 (sx, sy)から場所(gx, gy)まで矢印をたどっていきたいです。
  - しかし、今の盤面だといけない可能性が高いです。
  - 盤面を変えるにはコストがかかります。

- ゴールにたどり着くためのの最小コストを求めなさい。

問題概要

- 入力例
- その場合
- Mincost=4



- 制約：　小課題1→H=1　　小課題2→H,W≦80
- 　　小課題3→H,W≦600 小課題4→H,W≦10^5 全部においてN≦min(HW,70000)

小課題1 (190点)

- H=1，簡単に見えそう
- fの値（長さを変えるコスト）は共通
  - ⇒何個分長さを変えるか(p)について探索することを考える
  - それぞれのpについて、最小コストを記録しておくことを考える

- dp[今のマス目][今までで何個分長さを変えたか] = その状態での最小コスト を考える
- ただし、H=1の場合、最大でも W 個分しか長さを変える必要がない
  - dp[W][W]で空間計算量$O(W^2)$

小課題1 (190点)

- さあ、時間計算量は？
- 1つの状態につき、(そのマスに矢印がある場合)
  - 向きを変える場合・・・W個の状態に遷移(どのマスに飛ぶか)
  - 向きを変えない場合・・・W個の状態に遷移(どのマスに飛ぶか)

- $O(W^2) * O(W) = O(W^3)$で間に合う [定数が速いので]

小課題2 (360点)

- 同じ矢印を2度通るのは損
  - 実際に、同じ矢印を2回変えるのは許されていない
  - 変えることが許されていたとしても損であることには変わりない
  - 矢印を変えるのではなく、矢印 / 矢印でない場所 を通るためにコストが必要なのだと置き換えることができそう

- 小課題1の解法だと、残念ながら $O(H^3W^3)$でTLE
- グラフの最短路なので、**ダイクストラ法**が使えそう

小課題2 (360点)

- ダイクストラ法が使えそうなら、まずそのようなことを考える
  - dist[x][y] = マス(x, y)までたどり着くための最短コスト
- そのとき、各状態について、遷移は以下の2(H+W)通り
  - 北方向、南方向に進む・・・それぞれW通り
  - 東方向、西方向に進む・・・それぞれH通り
- 合計で、計算量は$O\big(HW(H+W)\big)\log HW(H+W)$
- 空間計算量は $O\big(HW(H+W)\big)$

小課題2 (360点)

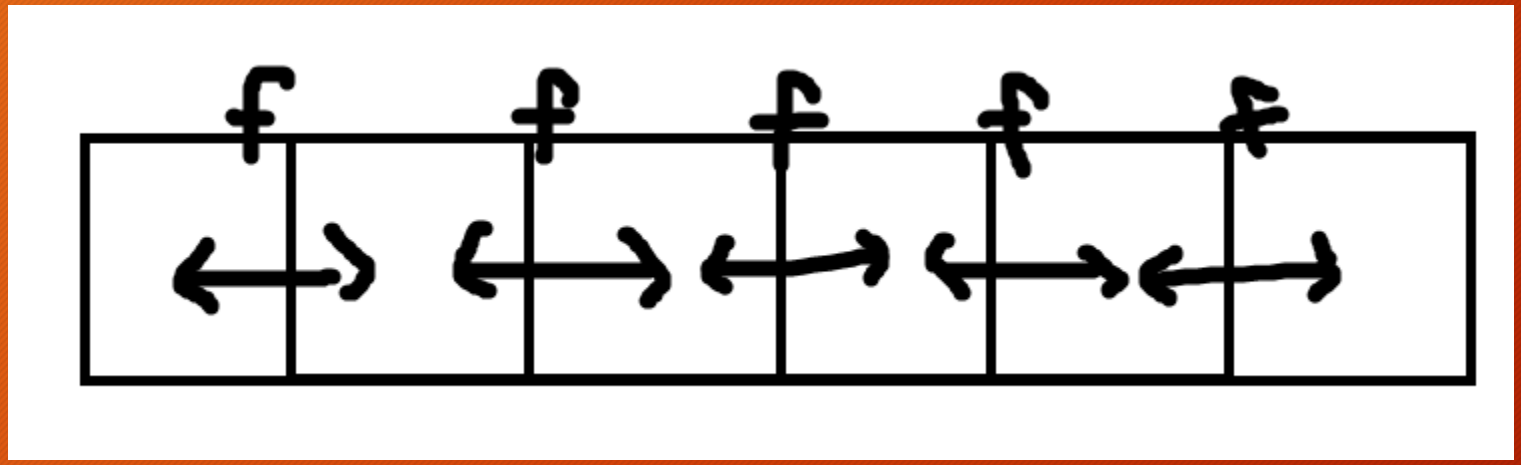
- 注意点
- 1. 配列を dist[88][88]とかにすると　小課題2 だけ通り、170点になります。その場合、dist[608][608]くらいまでは持っておきましょう。
- 2. 答えが “-1”になる場合に注意です。入力例にはありませんでした。

小課題3 (720点)

- 今度は H,W≦600
- オーダーが厳しい上に、MLEすら小課題2の方法だと厳しい

小課題3 (720点)

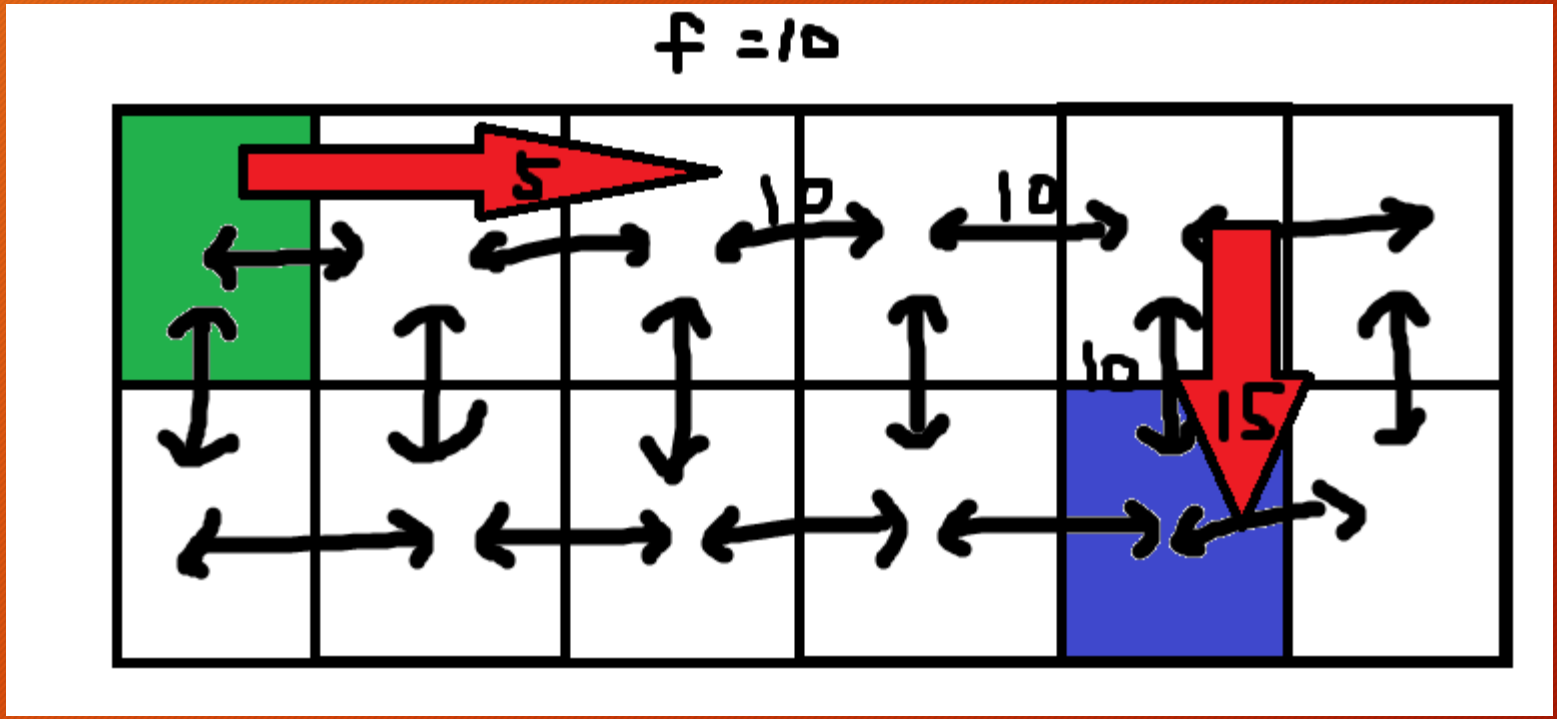
- そこで、fの値が共通、というものを使う



- という訳で、隣り合った辺にコスト f の辺を張るのでよさそう
- 矢印のもともとの辺はコスト0で残しておき、向きが変わったやつはコストe[i]にする
  - **それは嘘解法**

小課題3(720点)

- そういう場合に死ぬ



小課題3 (730点)

- 向きを記録するのが良さそう
  - 0 - 北向きに進んでいるという状態
  - 1 - 東向きに進んでいるという状態
  - 2 - 南向きに進んでいるという状態
  - 3 - 西向きに進んでいるという状態
  - 4 - 静止状態（次のステップでは、コスト f の辺ではなく矢印の辺で動く）

- それで、dist[x][y][向き]でダイクストラしたら良さそう
  - 矢印の辺は、向き=4の状態から出る（各方向 [向き変えも含む] について） : 一方通行
  - 長さfの辺は、dist[x][y][d]からdist[x+dx[d]][y+dy[d]][d]に向けて出る (0≦d≦3)
  - あと、dist[x][y][d]からdist[x][y][4]に戻す辺も必要

小課題3 (730点)

- そうやって、方向を5方向に分けて、ダイクストラ法をすると・・・
- **O(HW log HW)**

- 注意点
  - 1. 定数が重いです。
  - 2. 3パターンの辺を全て張らなければ、WAします。

小課題3 (730点)

- 実は、この問題は定数倍高速化で通すことが可能です。

- 辺を張らないダイクストラ法を考えます。(MLE回避のため)
- つまり、辺を張らず、priority_queueから出てきた状態から H+W個に直接遷移することを考えます。
  - そうすると、空間計算量はクリア
  - 時間計算量も自然に減った⇒1500msくらいで通る
- それを参照 http://ideone.com/GmCVoD

小課題4 (1150点)

- H,W≦100000になってしまいました。
- 小課題3の解法ですら、空間計算量 O(HW)かかってしまうので、MLEしてしまいます。

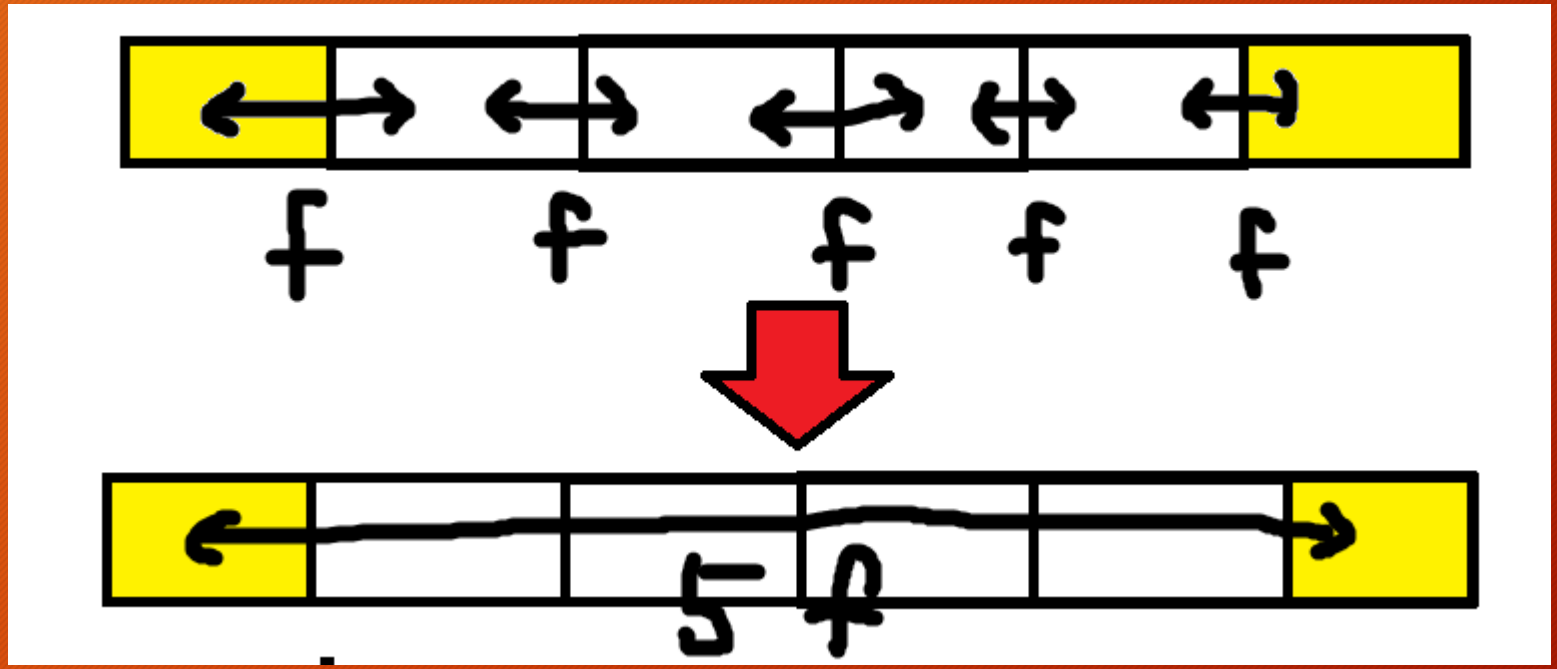
小課題4 (1150点)

- さあ、辺の張り方を工夫しましょう。
  - 矢印 + 矢印の向き変え は辺の数 4N なので良いとします。
  - 問題なのは、コスト f の辺の張り方です。数が HW 個になってしまいます。
  - それにしても無駄が多いです。無駄をどうやって削るか。

- まず、絶対必要な頂点というのを考えます。
  - まず、矢印の始点、終点は絶対必要です。
  - 向きを変えた（長さは変えない）矢印の終点も必要です。
  - それ以外必要ありません。

小課題4 (1150点)

- また、必要な頂点が限られた場合、以下のように長さ f の辺を減らすことができます。(黄色：必要なマス)



小課題4 (1150点)

- 絶対必要な頂点を p とすると、最大でも長さ f の辺の代わりとなる辺の数は 4p 個くらい
  - それは自明
- p=N*5=350000くらいなので、必要な辺の数は 1400000個くらい　：　O(N)本
- 矢印 + 矢印の向き変えはもっと少なくて、N*5本くらい　：　O(N)本

- →計算量は、$O(N \log N)$→ ~~通ったぁ！~~ それは分からない
  - 定数が重い(*25とかある)ので、コードや言語によっては通らないことがあります。

小課題4 (1150点)

- 実は、向きを 5 から 3 に減らすことができます。
  - 0 - 北か南に進んでいる
  - 1 - 東か西に進んでいる
  - 2 - 静止状態（次は矢印に乗る）
- 証明
  - 静止状態からでないと矢印に乗ることができない
  - 長さ f の代わりの辺を北→南 と行っても、結局もとのマスに戻るだけで損をするので、場合分けする必要はない
  - を使うことで証明ができる

$O(15N \log N)$ くらい

小課題4 (1150点)

- 注意点
- 1. 定数が重いです。
- 2. long long型にして、オーバーフローに注意しましょう。
- 3. 途中で座標が負になることがあります。配列外参照などにご注意ください。

結果

1150

1150

720

360

190

190

G問題：Get the Salary of Atcoder

square869120Contest #4 解説

問題概要

- N頂点の頂点0を根とした根付き木が与えられる。また、p_i を頂点 i の親とすると、$i \geq 1$ のとき $p_i < i$ が成り立つ。また、各頂点には最初 a_i が書かれている。また、Q個の次のようなクエリが与えられる
- タイプ1：v_i, d_i, x_i が与えられ、v_i の部分木の中で v_i との距離が d_i 以内の頂点に書かれた値を x_i 加算する
- タイプ2：v_i, d_iが与えられ、v_i の部分木の中で v_i との距離が d_i 以内の頂点に書かれた値の合計を求める
- タイプ3：c を現在の頂点数とするとき、p_c, a_c が与えられ、頂点 c を新たに追加する

制約

- $N \leq 400000$
- $Q \leq 40000$
- $0 \leq x_i \leq 1000$ などなど

小課題1 (170点)

- $N \leq 5000, Q \leq 5000$
- 愚直にやると間に合う
- クエリ1：DFSを使って 計算量 $O(N+Q)$ ($N+Q$ は頂点数)
- クエリ2：DFSを使って 計算量 $O(N+Q)$ ($N+Q$ は頂点数)
- クエリ3：木を隣接リストで表現すると計算量 $O(1)$ で処理できる

小課題2 (240点)

- $N, Q \leq 400000$
- $p_i + 1 = i$ (木はリストのようになっている)

- 木はリストのようになっているので、これを使えないか?

小課題2 (240点)

- 木はパスなので配列と考えることができる
- するとクエリは次のように置き換えられる
- クエリ1：区間 $[v_i, \min(v_i + d_i + 1, c))$ に x_i を足す。
- クエリ2：区間 $[v_i, \min(v_i + d_i + 1, c))$ に書かれている整数の合計を求める。
- クエリ3：c に1を加算し、a_cをセットする

小課題2 (240点)

- これは単純な Range Add Query / Range Sum Query で処理できる
- 分からない人は 蟻本のBITを使った解法とか kagamizさんのブログ (http://kagamiz.hatenablog.com/entry/2012/12/18/220849) とかを見よう

- これで各クエリが$O(\log(N+Q))$ で処理できたので、計算量は $O(Q\log(N+Q))$ => よって解くことができた

小課題2 (240点)

- 小課題1と組み合わせれば 410点 が取れる
- $N \leq 5000, Q \leq 5000$ のときに小課題1のコード、そうでないときに小課題2のコード、とすればよい

考察 - 数列のクエリに帰着する

- 木を数列のクエリに帰着することはできないか?
- 数列のクエリにすると単純に考えることができそう

考察 - 数列のクエリに帰着する

- 木をDFSをして、訪れた順を記録する (ord)
- するとある頂点について、この頂点の部分木は数列ordの区間になる (これも記録しておく, l_i と r_i)
- 深さもあるので、この順に深さも並び変え (これを$depth_i$とする, クエリが全部終わったときの深さ rd_i も記録する)、a_i も並び替えると、数列のクエリに帰着できそう
- まだ追加されていない頂点を $depth_i = inf$ とすると実装が楽そう（クエリ3で a_i を更新しなくてすむ)

考察 - 数列のクエリに帰着する

- 結局必要なのは次のようなクエリになった
- タイプ1：$depth_j \leq depth_{v_i} + d_i \ (l_{v_i} \leq j < r_{v_i})$ を満たす j に対して a_j に x_i を加算する
- タイプ2：$depth_j \leq depth_{v_i} + d_i \ (l_{v_i} \leq j < r_{v_i})$ を満たす j の a_j の合計を求める
- タイプ3：$depth_c$ を inf から rd_c に変える
- ここからは $l_i = l_{v_i}, r_i = r_{v_i}, d_i = depth_{v_i} + d_i$ とする
- 愚直にやると $O\big(Q(N+Q)\big)$ 。どのように高速化するか?

小課題3 (500点)

- クエリ1と2しかない
- 愚直では間に合わない
- どのようにするか?

- まずは Q が少ない場合の高速な解法を考えてみよう (Q=500くらい?)

- そのとき l_i, r_i に出現する数は $O(Q)$ 個しかないので座標圧縮ができそう (座標圧縮した時に $O(Q)$ 個に分かれるが、これをここでは区間という)

小課題3 (500点)

- l_i, r_i は区間の一番端にある => $O(Q^{0.5})$ 個の区間に対して処理してやればよい
- クエリを処理する前に各区間に対して二分探索できるようにあらかじめ d_i を区間ごとにソートしておく
- するとクエリに対して d_i を二分探索できるようになる (これを ptr とする)

小課題3 (500点)

- 各区間に対してやるべきことは、次のようなことである
- タイプ1: $b_0, b_1, b_2, \ldots, b_{ptr-1}$ に x_i を加算する
- タイプ2: $b_{ptr}, b_{ptr+1}, b_{ptr+2}, \ldots, b_{|b|-1}$ の合計を求める (最初に累積和を記録すると最初の a_i の状態での合計も求められるので最初に$b_i = 0$ で初期化する)
- これは Range Add Query / Range Sum Query ではないか? (これはBITを使って実装すると定数倍が軽いです)
- これによって問題が $O\big((Q^2+N)\log N\big)$ で解けた
- ソートを工夫して線形でやることもできて、これだと $O(Q^2 \log N + N)$

小課題3 (500点)

- 最終的な a_i も復元できないか?
- BITでそのまま計算すると $O(N \log N)$ 、DPで計算すると $O(N)$ で数列 a_i の値が求まる
- これも含めると計算量 $O(Q^2 \log N + N)$

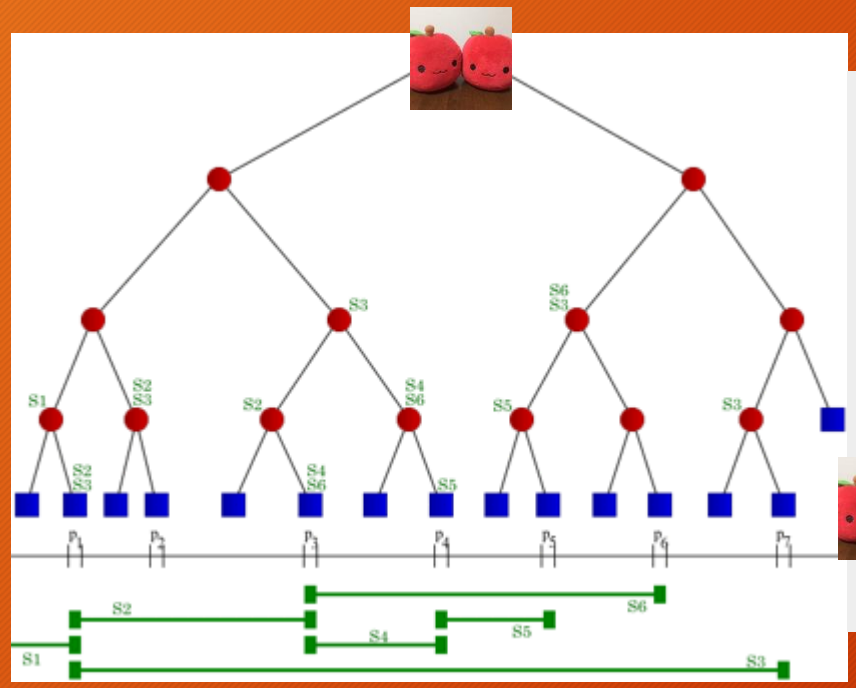
小課題3 (500点)

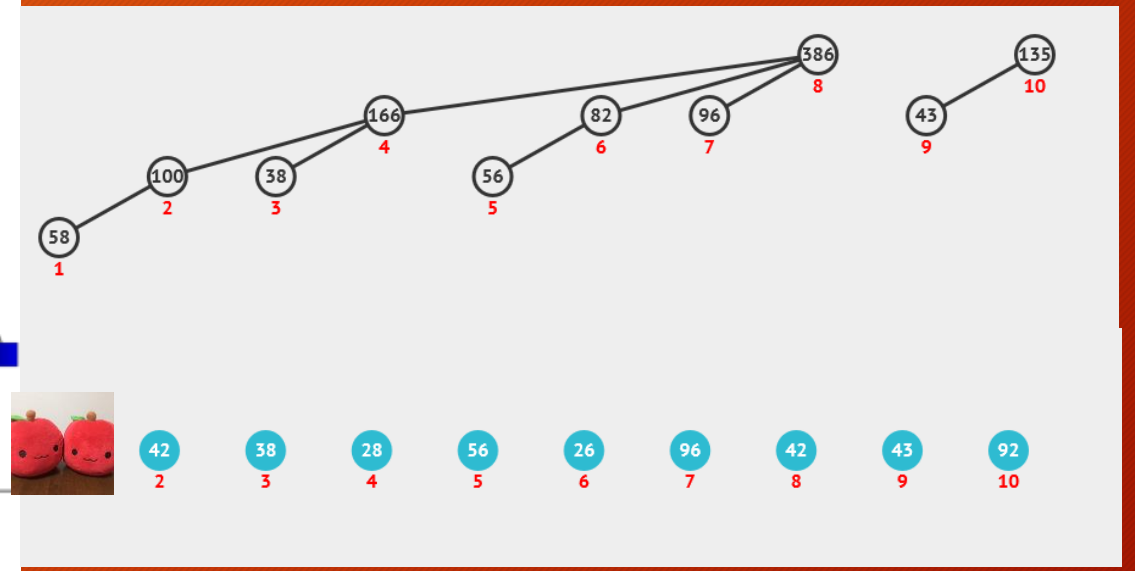
- 最終的な a_i の値も求まったしクエリを平方分割できそう
- 先ほどの方法で、クエリを $O(Q^{0.5})$ 個ごとに処理すると、$O(Q^{0.5})$ 回 × $O(Q \log N + N) = O(Q^{1.5} \log N + Q^{0.5} N)$ で処理できる
- $N = 400000, Q = 40000$ なので、計算量的に十分間に合いそう、定数倍が遅くても間に合う
- 定数倍を速めるためには、平方分割ではなく $\frac{Q^{0.5}}{5}$ 個程度の区間に分割するとできる

満点解法 (1400点)

- もう少し工夫すればクエリ3も処理できる
- クエリ3は、$l = ord_c, r = ord_c + 1$ としてこれも座標圧縮にいれると解きやすい (クエリ3の処理の区間が必ず1になる)
- 区間の長さが1なので、$depth_i$ を変えてももう一度ソートする必要はない
- よってクエリ3には区間を探す2分探索で $O(\log N)$ しかかからないので、効率的に解くことができる (クエリ1, 2の実装は変えなくてもよい)
- 小課題3と同様 $O(Q^{1.5} \log N + Q^{0.5} N)$ で解くことができる

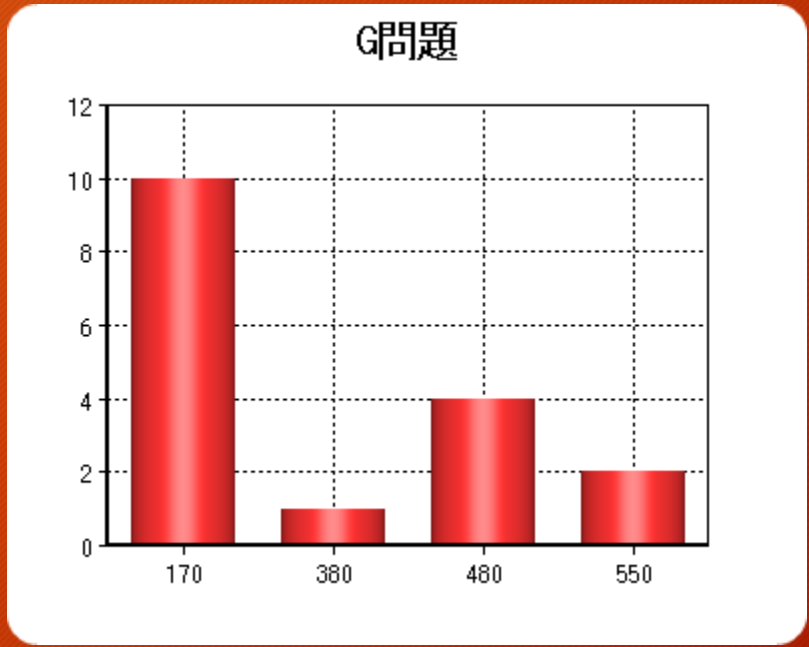
満点解法 (1400点)


S1
S2
S3
S4
S5
S6
p1
p2
p3
p4
p5
p6
p7


58 1
100 2
38 3
166 4
56 5
82 6
96 7
386 8
43 9
135 10
42 2
38 3
28 4
56 5
26 6
96 7
42 8
43 9
92 10

結果

- 1450点者:0
- 550点者:2人
- 480点者:4人
- 380点者:1人
- 170点者:10人



H問題： Huge Kingdom Atcoder

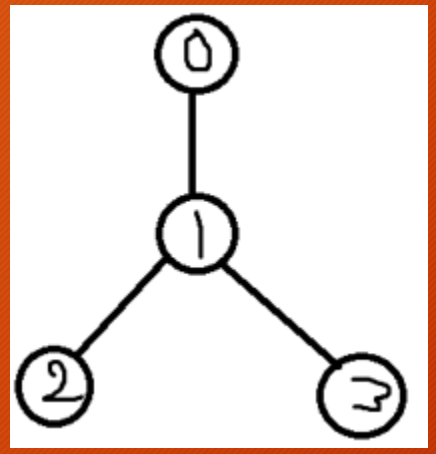
Square869120Contest #4 解説

問題概要

- 木が与えられます。 N頂点です。
- その時、あなたは以下のような質問をして、木を当てなければなりません。
  - 文字列S(N文字)を出力する。その時、Si=1の時、i番目の街を黒く塗り、Si=0の時、i番目の街を白く塗ることを意味する。
  - N頂点のグラフGを考える。
  - 王国の道の両端の街が、両方とも黒く塗られている場合、グラフGにその辺を追加する。
  - グラフGの各連結成分についての、「木の直径」を2乗した値の総和が返される。
- その時、できるだけ少ない質問回数で構造を当てなさい。

問題概要

- 制約：N=200
- グラフは木である。
- 入出力例は右へ

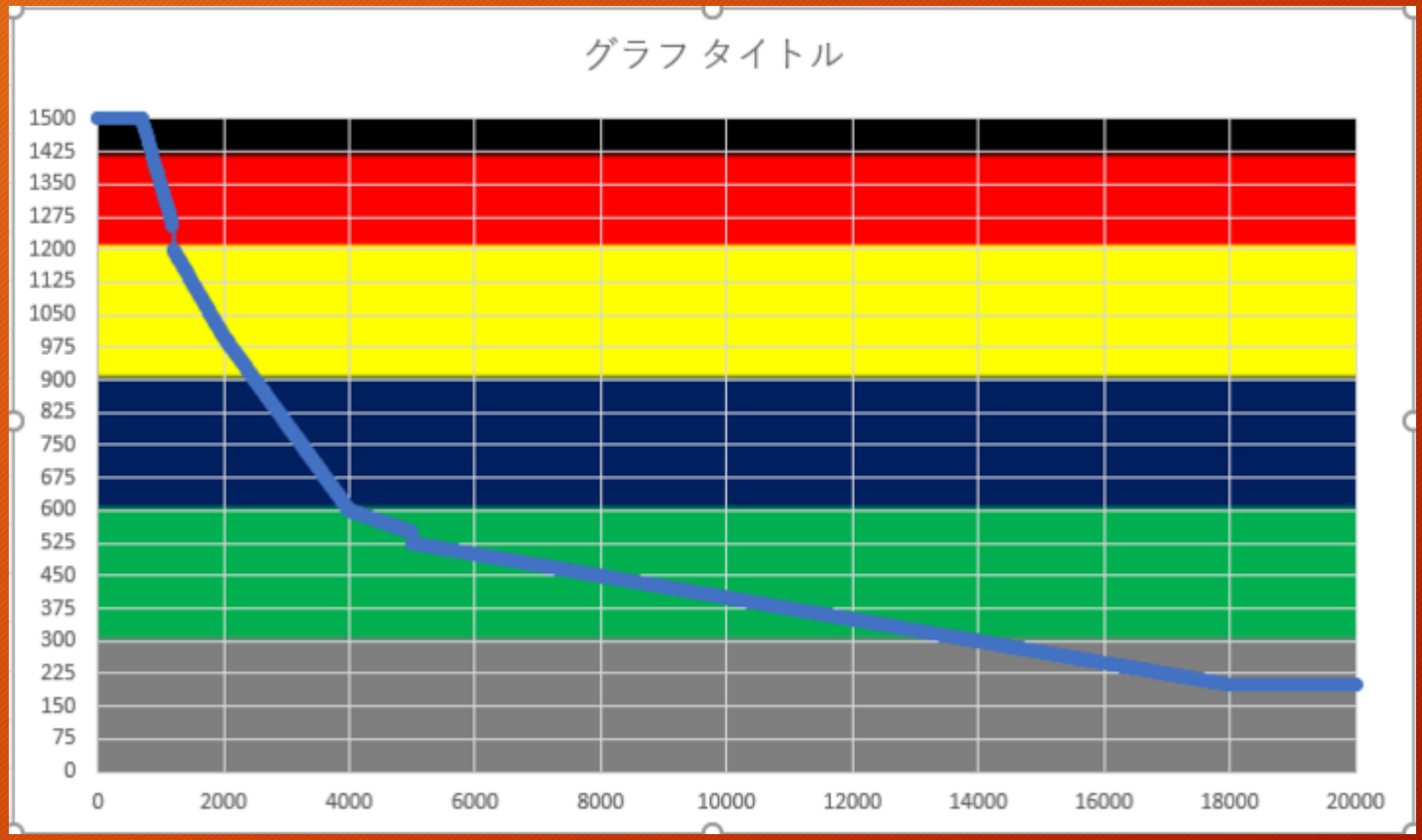


そのグラフの場合→

入力	出力
4	
	? 1111
4	
	? 1101
4	
	? 1001
0	
	? 1100
1	
	? 1011
0	
	! (0,1) (1,2) (1,3)

得点

- 得点はそんな感じ　ー　この問題は[マラソンタスク](#)です



200点解法

- まずは最初の200点から説明していきます。
- 全ての頂点対について、「その道があるかないか」を質問することを考えます。
  - i と j を結ぶ道があるかないかは、”0…010…010…0”というような文字列で質問することで質問可能
  - 1の位置は i文字目と j文字目
  - もし道がない場合 “0”、ある場合 “1”と返される　ー　それは自明
    - 道がない場合、グラフGに1本も辺が張られない　ー　0と返される
    - 道がある場合、グラフGに(i,j)の辺が1本張られる　ー　1と返される

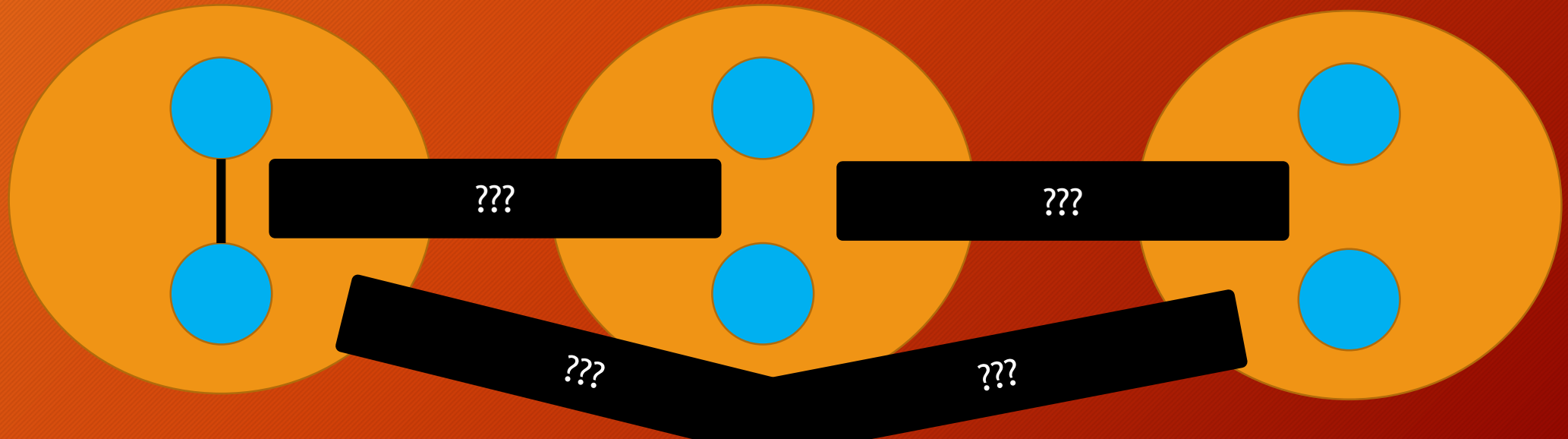
200点解法

- “0…010…010…0”というような質問を、1≦i＜j≦nを満たす全ての i, j について行うことを考える
- 結果が1の時にのみ、「そこに辺があること」を出力
- 質問回数は$O(\frac{1}{2}N^2)$回 ➡ N=200のとき20000回
- 出力量は、$\frac{1}{2}N^2 * N = \frac{1}{2}N^3$ バイト➡N=200のとき4000000B ➡ printfを使うこと。

200点

328点解法, 588点解法

- 328点解法　－　これは、$O(\frac{1}{2}N^2)$ではできません。
- しかし、20000 個の頂点対を探索しているのにもかかわらず 200 個しか辺がないのは無駄だと感じませんか？
- そこで、以下のことを考えます。



328点解法, 588点解法

- まず、N 個の頂点を p 個ずつに分けます。
- グループ i の中での値をa_iとします。
  - ここでいう「木」は、グループ i 内でのグラフ（両端がグループ i であるものだけに辺を張ったもの）
  - それは、”0...01...10...0”という1回の質問で聞くことができる
- あとは、グループ i とグループ j を結ぶ頂点があるかどうかを調べます。
  - 無い場合、”0...01...10...01...10...0” (‘1’ = グループiとjに含まれる頂点)の値は、$a_i + a_j$となります。
  - ある場合、それを超える値になります。 ― 2乗の性質を利用する

328点解法，588点解法

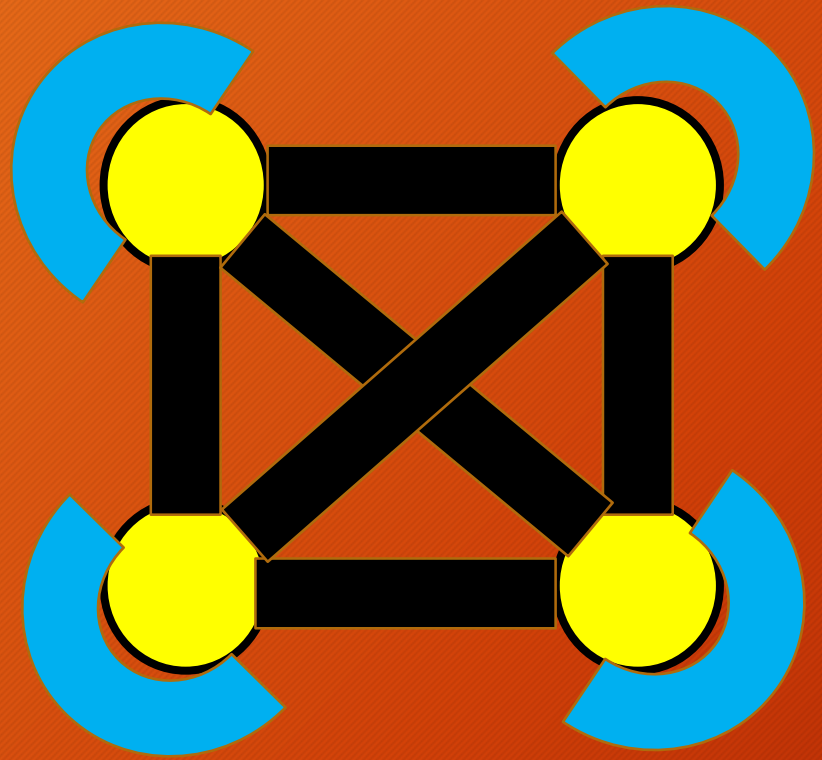
- それを利用して、以下のようなアルゴリズムで解くことができます。

- グループｉ内の値a_iを求める
  - a_i ≧1のとき、グループｉ内を結ぶ辺があることを意味する。
  - あった場合、そのような辺を全探索する [$\frac{P(P+1)}{2}$回の質問：Pはグループ数]
- 1≦i＜j≦B (Bはグループ数) について、グループｉとｊの間を結ぶ辺があるかないかを全探索 ➡ $\frac{B(B-1)}{2}$通り
  - H(質問の返答) =$a_i + a_j$の場合、辺がないので、スキップ
  - そうでない場合、辺があることを意味する⇒そのような辺を全探索
  - 全探索は、その部分だけ200点解法と同じ感じでやる [P^2回の質問：Pはグループ数]

そこで1辺ずつ全探索する意味・・・
どこに辺があるか特定するため

328点解法，588点解法

- 解き方を図で表してみると、そんな感じになります。



意味

そのグループ内を結ぶ（両端ともそのグループ内の）辺があるかどうか判定

あったら$\frac{P(P+1)}{2}$通りを全探索

2つの指定されたグループ内を結ぶ辺があるかどうかを判定

あったらP^2通りを全探索

328点解法，588点解法

- 計算量の見積もり（グループ数をPとする）
  - グループ内を結ぶ辺を全探索・・・P通り・・・①
  - 違うグループを結ぶ辺・・・$\frac{1}{2} * \mathrm{P}^2$通り・・・②
  - そのうち、辺があって詳細な全探索が必要なものは精々 N 通り
    - ⇒全探索の合計コストは最大で$\max(\frac{N^2}{2}, N * \left(\frac{N}{P}\right)^2)$通り

- ⇒P=sqrt(N)くらいが良さそう⇒ P=20とする
- 最悪計算量 $\min(20000, 200 * 10^2 + 20 + 20^2)$=20000回くらい
- 実際に、ランダムケースなのでもう少し良い結果が出る 328点

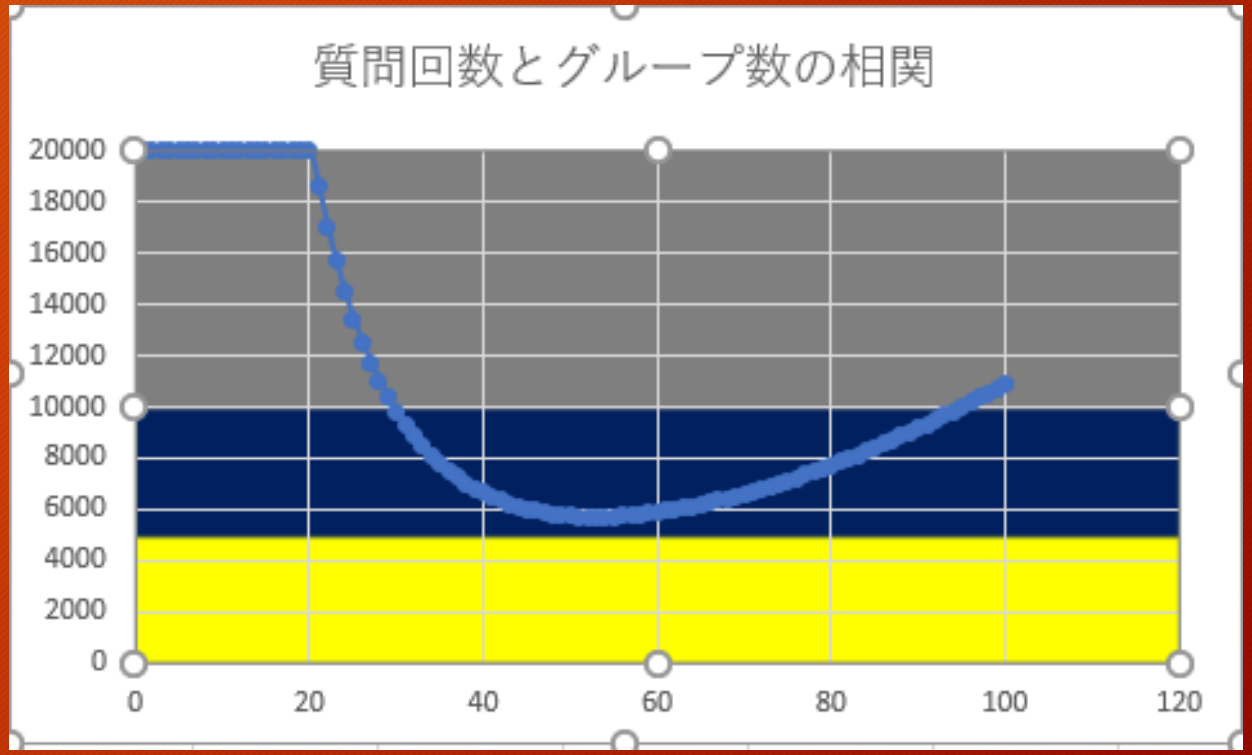
328点

328点解法，588点解法

- 実際に、もっと良い P はあります。
- Excelでグラフを作ってみると・・・

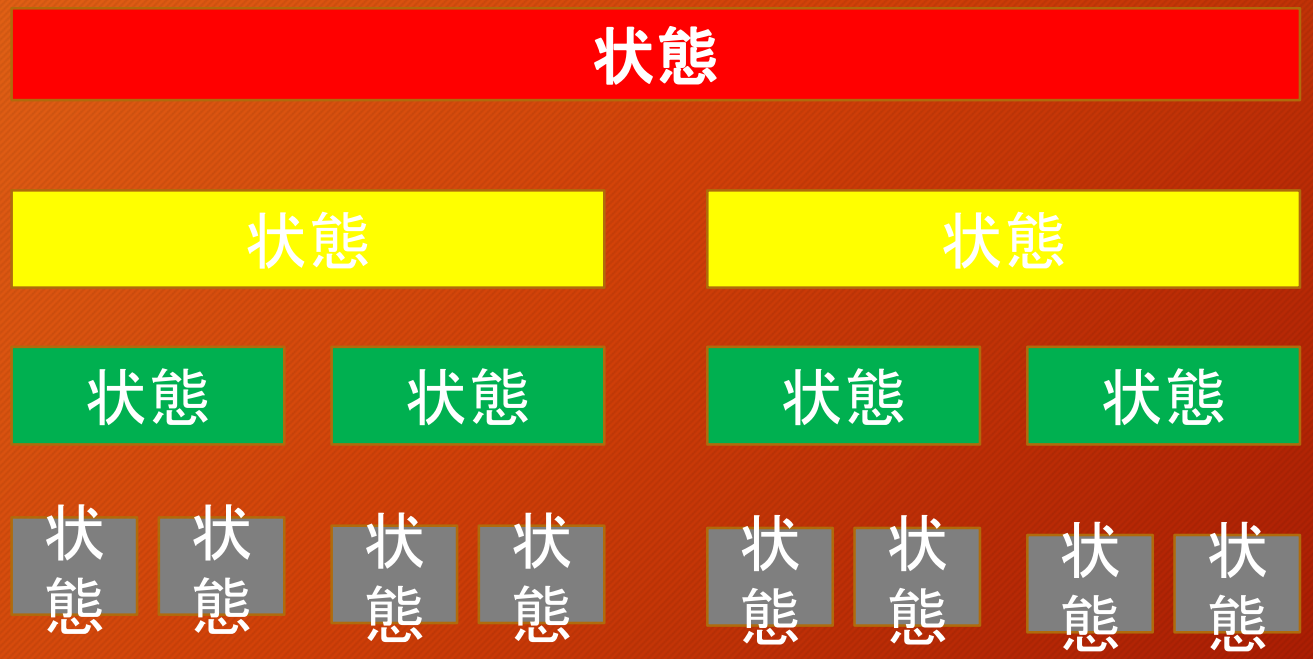
- P=50くらいが一番良さそう
  - つまり、4個ずつに分ける
- ⇒最悪質問回数は約 6000 回

588点



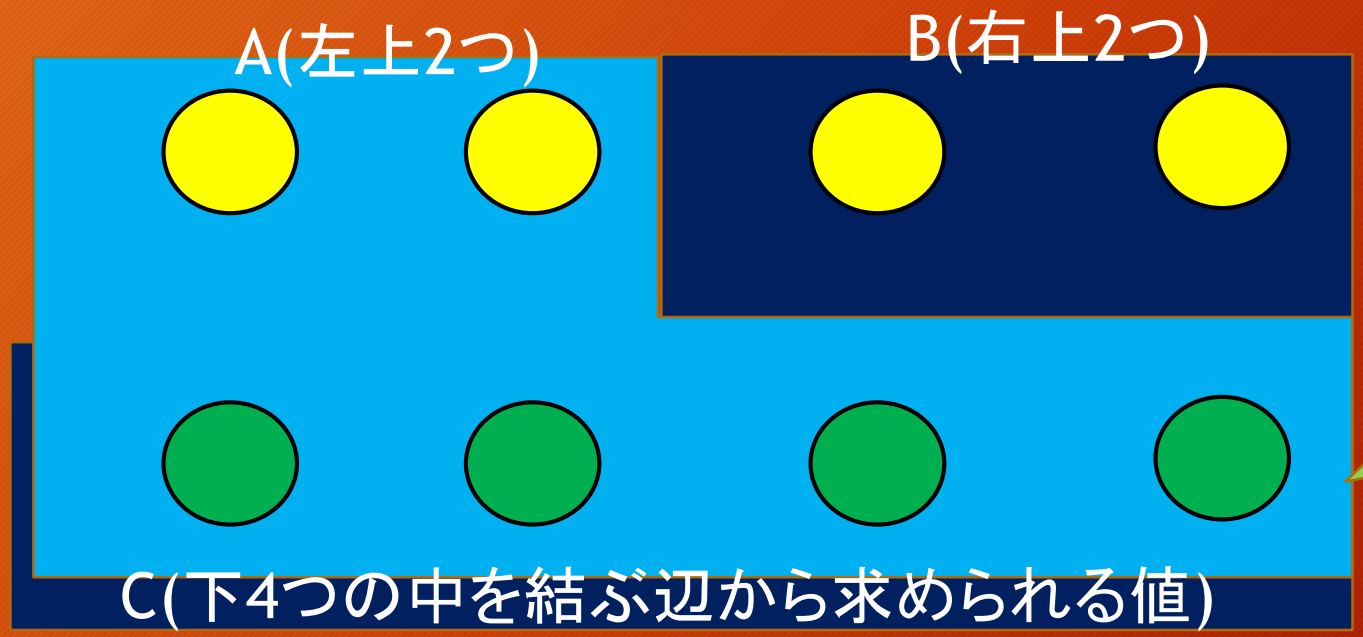
825点解法

- これは平方分割のアルゴリズム
- ⇒これを再帰でできないか・・・（s8pc-3-hでは再帰を使って 756点）



825点解法

- 実際に、再帰を使う方法は、存在します。
- イメージとすれば、そんな感じです。

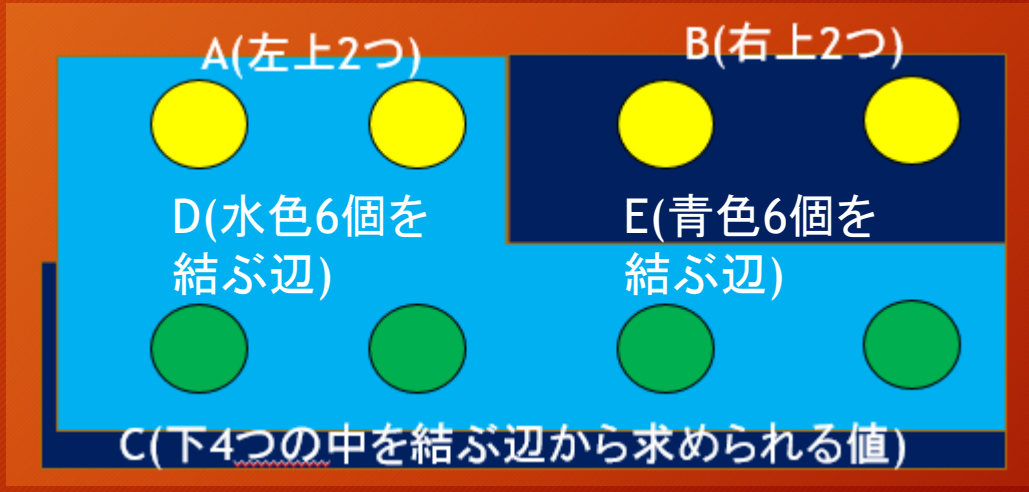


825点解法

- 水色の部分で、黄色と緑を結ぶ辺 [8個候補がある] が1つ以上あるかどうかは、D=A+Cかどうかで判定できる

- 青色も同じように、E=B+Cかどうかで判定可能

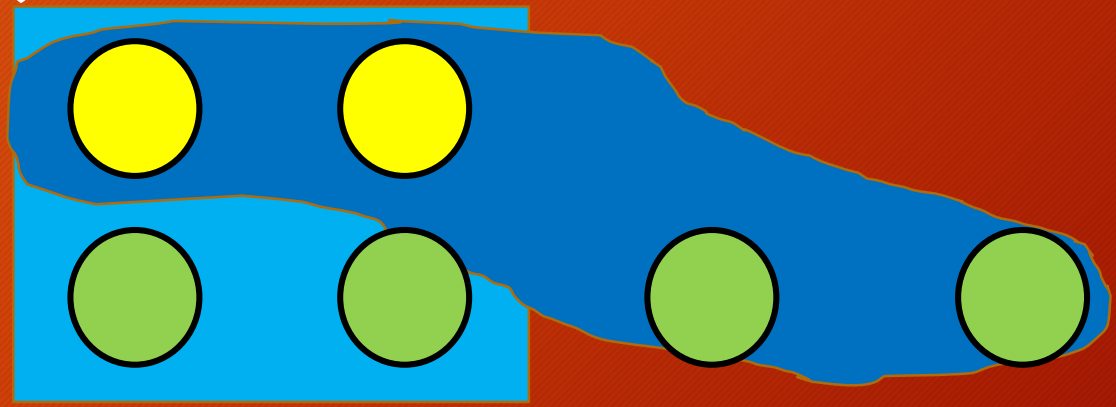
- 辺があったら、再帰的に深さを増やしていく
- なければ、その方向には深さを増やさない



←[4, 4] から [2, 4]に再帰を減らす
[a, b]は、[黄色の頂点の個数, 緑の個数]

825点解法

- [2, 4]を[2, 2]に減らすこともできます
- こんな感じです。

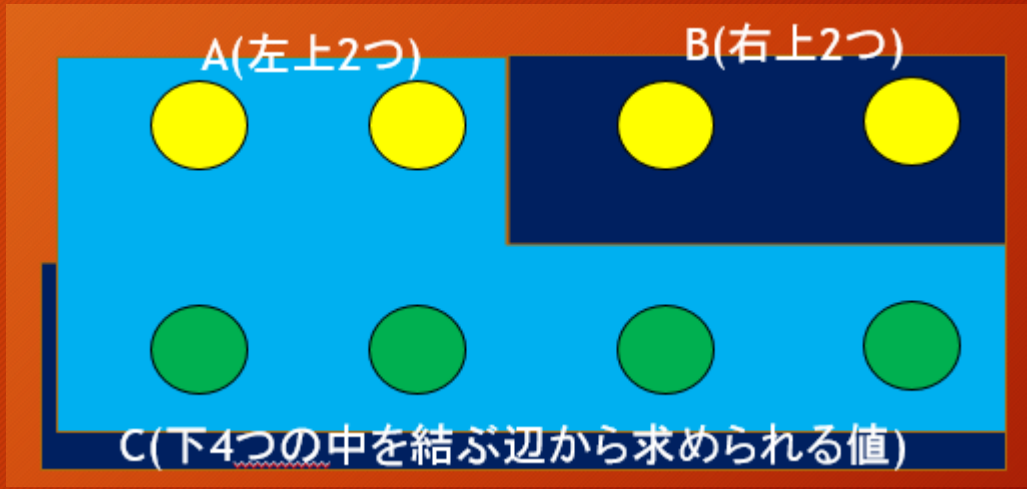


- よって、再帰では、
- 数を [4,4] -> [2,4] ->[2,2] -> [1,2] -> [1,1]と減らすことができます。
- [1,1]でも残っている場合、そこに辺があることを意味します。

825点解法

- さて、計算量解析をしましょう。

- まず、A, B, Cの質問回数は、右図のような感じです。
  - ⇒O(2n) ≒O(n)
  - ただし、同じ質問を2度繰り返したときに質問しないように、質問の内容を覚えておくことも重要
  - map<vector<int>>持っていると時間足りないので、Hashすることを勧める



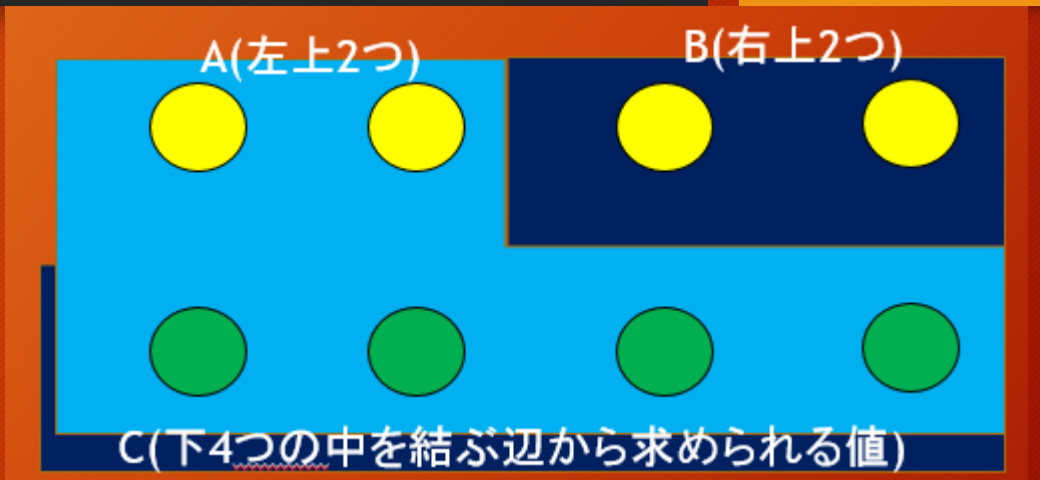
区間[0, 7]の中の辺だけで求められた値

区間[0, 3]　区間[4, 7]

区間[0, 1]　区間[2, 3]　区間[0, 3]　区間[4, 7]

825点解法

- D,Eの質問回数は、
  - 辺の個数が n 個
  - 再帰の深さが 2log n
  - ⇒最悪計算量は$O(4n\log n)$以下
- 実際は、深さ log nくらいまでは共通部分多そう
  - ⇒ $O(2n\log n)$くらいと見積もれる
- ⇒n=200のとき、$O(2n\log n)+O(2n)$≒3 200回



825点

943点解法

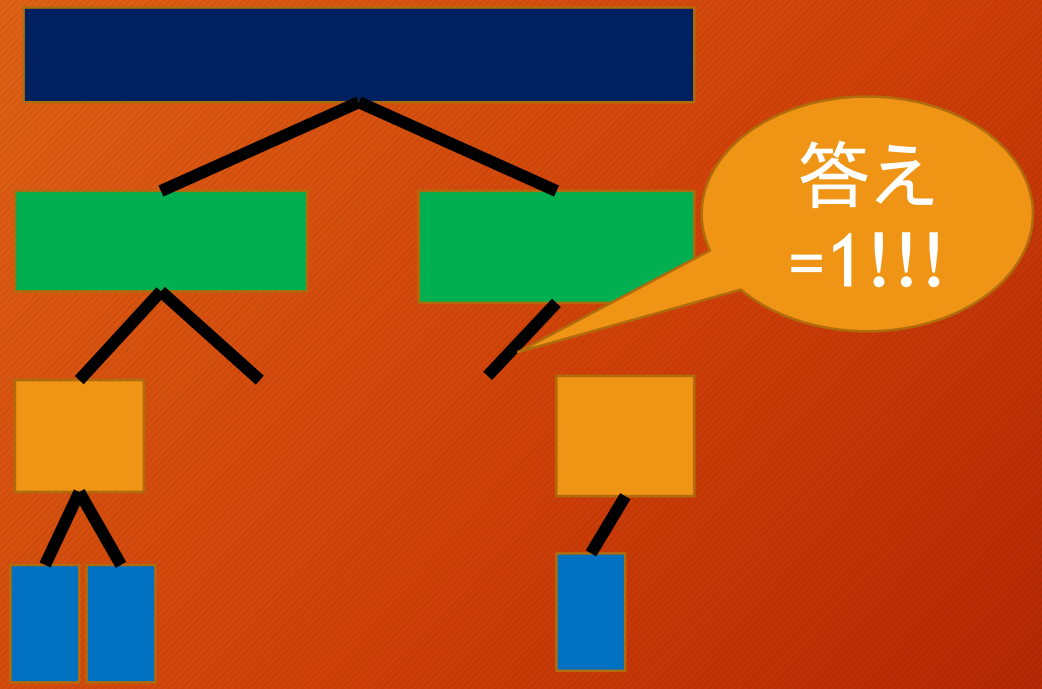
- それをさらに工夫
- 実際に、再帰の途中で答えが1以下になったら以下のことが成り立つ
  - 左側・右側両方さらに深く探索することはない　ー　つまりどちら一方かはここで枝刈りされる
- その性質を利用して、探索回数を以下のように減らせる
  - それより下の深さでは、左側の答えが1であれば左側だけさらに深く探索
  - そうでなければ、右側だけさらに深く探索

- そうすると、定数の 2 が 1 になる部分が多くなる
- ⇒ O(n) + o(1.5n log n)くらいになる

943点

943点解法，1072点解法

- 図でいえば、そういうイメージでやる



線が質問、長方形が状態を意味する。

一部を平方分割にしたりして、うまくパラメーター調整をすると、1072点が取れる。

実装はヤバイので要注意。

1072点

1072点以上

- Writer解はここまでである。
- 一応この問題はマラソン系問題なので、満点解が存在しなくてもよい。

投稿日時	問題名	ユーザ名	言語	得点	ソースコード長	状態	実行時間	メ
2017/04/06 18:11:16	H - Huge Kingdom: Atcoder	E869120	C++14 (GCC 5.4.1)	1073	5351 Byte	AC	86 ms	
2017/04/06 10:07:39	H - Huge Kingdom: Atcoder	E869120	C++14 (GCC 5.4.1)	1072	5229 Byte	AC	89 ms	
2017/04/06 17:58:15	H - Huge Kingdom: Atcoder	E869120	C++14 (GCC 5.4.1)	1072	5083 Byte	AC	86 ms	
2017/04/06 10:15:20	H - Huge Kingdom: Atcoder	E869120	C++14 (GCC 5.4.1)	1072	5337 Byte	AC	86 ms	

1200点以上

- その問題で1200点以上を取るためには、以下のようなアプローチが必要です。

- 木の通り数は、$\prod_{i=1}^{N-1} i(N-i)$くらいあります。大体2^{2600}くらいでしょう。
- つまり、各質問について2通りに場合分けする場合、仮に場合分けが確率半々くらいだとしても、2600回となります。かなりの工夫をしても、1200回（1200点）を超えることは簡単ではないでしょう。

1200点以上

- それを知るためには、以下のように場合分けする必要があります。
  - 0, 1の2通りではない
  - 返り値として考えられるものの通り数は40000 (/2?)通りくらいあると見積もれるので、これをまんべんなく使って場合分け
  - ‘1’の数を少なくするともったいないので、’1’の数を多くして探索
- しかし、それを場合分けするには、とても大きな量のデータが必要になります。それを如何にして最適に近く探索するか、というのがその問題の1200点以上解法の焦点です。

最後に

- 今回も、最終問題はマラソンでした。
- どうでしたでしょうか？　何点取れましたか？
- 面白ければ幸いです。

- ちなみにs8pc-3からは、最終問題にミニマラソンを入れることが続いています。

- という訳で、コンテストお疲れさまでした！！！！！！

結果

- tomerun→693点
- fjzzq2002→606点
- WA_TLE→540点
- nmnmnm…→388点
- autumn_eel→386点
- その他　正の点数を取られた方々→27名

The Last – 講評

Square869120Contest #4

講評

- 今回のs8pc-3, どうでしたか？
- 今回は、s8pc-1に比べればかなり難しいセットになっているはずです。
- 第4回目、の有志コンテストとなりましたが、s8pcはs8pcとしての傾向があると 思うので、過去問を解くとよいと思います。
- ちなみに、writerはJOIerなので、情報オリンピックの対策にも少しはなると思います。
- 皆さん、ご参加ありがとうございました！

全体結果

問題	提出数	提出者数	部分点者数	満点者数	配点	難易度
A	363	141	37	85	250	400
B	304	138	37	94	350	350
C	140	140	28	21	500	900
D	206	206	29	43	800	800
E	40	40	14	2	1000	1500
F	23	23	4	2	1150	1500
G	117	117	17	0	1450	2000
H	142	142	32	0	1500	2000