技術室奥プログラミングコンテスト #3 解説

平成 30 年 7 月 14 日

筑駒パ研 一同

コンテストの概要

- 問題数 … 10 問 （第 1 回 ～ 第 3 回まで全てそうです）
- 制限時間 … 4 時間
- Writer, Tester … ynymxiaolongbao, autumn_eel, Thistle, waserabi_29, Ryo2016, keymoon, TAISA_, E869120
- Rated … No (unrated)

- これは、筑駒パ研が主催するコンテストです。
- 技術室奥プログラミングコンテストは、今まで 2015, 2016 年に開催されました。先輩方が引退したため、2017 年度は開催しませんでしたが、筑駒パ研内で 2018 年度のコンテストをやろうという動きが高まったため、今年は開催することに決めました。
- 参加していただき本当にありがとうございます。また、AtCoder 社には感謝します。

全体の講評

問題	配点	Solved	Submitted	Submissions	予想難易度
A: 時差	100	198	201	237	100
B: 鰻と忖度	200	179	193	347	200
C: 新入生歓迎数列 –Easy	300	131	169	459	300
D: 巨大チェスボード	400	97	114	191	400
E: デフレゲーム	500	78	91	121	500
F: 天使とふすま	700	53	70	85	600
G: パソコンの買い替え	700	5	24	151	1000
H: 新入生歓迎数列 – Hard	1000	6	36	140	1000
I: 王国と M 種類の店	1100	2	17	50	1400
J: 円の重なり	1300	0	11	35	-

A問題　時差

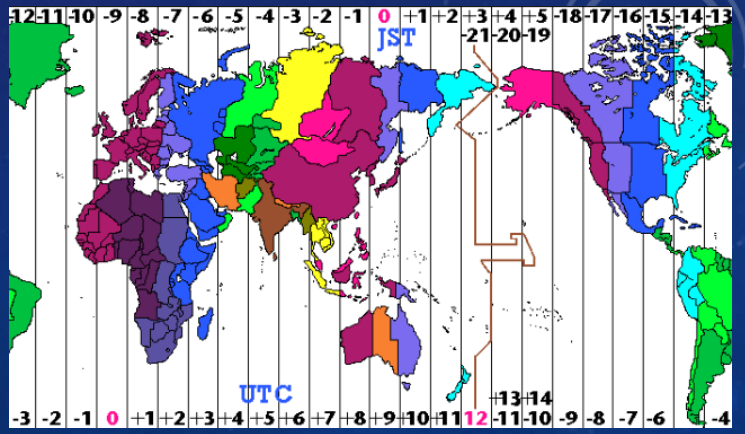
技術室奥プログラミングコンテスト #3

Writer: waserabi_29 (@waserabi)

問題概要

- 2 地点の経度が ‘E 90’ や ‘W 120’ のような形で与えられる。
  - それぞれ、東経 90 度・西経 120 度を意味する

- 2 地点間の時差を求めよ。
  - ただし、時差は日付変更線をまたがないように差を取ったものだとする
  - 日付変更線は東経 180 度にあるものとする



解法

この問題は二つの地点がともにWやEなのか、それとも
W, Eの組み合わせなのかというところがポイント

＜経度が同じ記号の場合＞
・二地点の経度の値の差を求めて15で割るだけ。
・値の差は例えばC++ではabs関数を使って求めることができる。

＜経度が違う記号の場合＞

・　要するに経度というのは東経/西経0度線から何度離れているかということなので西経はその線から西に行った分、東経は東に行った分と考えられる。

・つまり経度の差は二地点の経度の値を足すことで求められる。

サンプルプログラムは以下のようになる

```
1. #include <iostream>
2. #include <cmath>
3.
4. using namespace std;
5.
6. int main()
7. {
8.     char c1,c2; int a,b;
9.     cin>>c1>>a>>c2>>b;
10.    if (c1==c2) cout<<abs(a-b)/15<<endl;
11.    else cout<<(a+b)/15<<endl;
12.    return 0;
13. }
```

B問題　鰻と忖度

Made by Thistle (@thistleprogram)

想定解

6の倍数は、

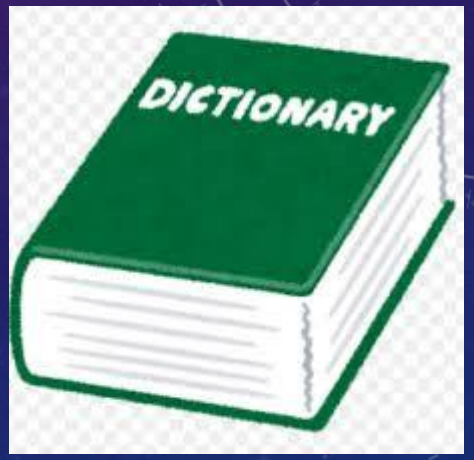
- 2の倍数：最後の桁の数字が偶数
- 3の倍数：全ての桁の数字の和が3の倍数

を適用することで判別できる。

11の倍数は、奇数桁目の数の和と、偶数桁目の数の和が等しいかどうかで判別できる。

証明は、ネットで調べたらたくさん出てくる。

解法②



巷のプロが空気を吸うより簡単に使いこなしている、

一桁ずつ、

㋑一つ前の桁までの結果を10倍したものに今の桁を足す

㋺それの余りを取る

を繰り返す、という解法がある。

参考：ABC030　「へんてこ辞書」https://www.slideshare.net/chokudai/abc030

解法③

Pythonなどの言語には、標準で多倍長の演算機能が実装されているので、
それを利用して計算をする。

新入生歓迎数列-Easy

技術室奥プログラミングコンテスト #3

Writer:TAISA_(@ta1sa_)

問題概要

・$\{A_1, A_2, \ldots, A_N\}$が書かれているN枚のカードがある

・これらの中から隣接する2枚のカードを選び、その2枚に書かれている整数の積が書かれた新しいカードに置き換える

・これを繰り返して、整数Pの書かれたカードを作れるか？

考察

・このままだとよくわからない

・結局、操作によってできる整数としてあり得るものは何だろうか？

考察

- $A_1, A_2, A_3, \ldots, A_N$
- $A_1*A_2, A_2*A_3, A_3*A_4, \ldots, A_{N-1}*A_N$
- $(A_1*A_2)*A_3, (A_2*A_3)*A_4, \ldots, (A_{N-2}*A_{N-1})*A_N$

…

- $A_1*A_2*A_3*\ldots*A_N$

「**数列Aの中の空でない連続する部分列内の要素の積**」なら何でも作れる！

(以下、これを「連続する部分列の積」と呼びます)

考察

・つまり、「**連続する部分列の中に積がPであるものは存在するか？**」という問題を解けばよい

・単純な全探索だと$O(N^3)$となり間に合わない

・「左端を全探索して積がPを超えるまで右端を伸ばす」という解法でも 、{1,1,1,...,1} のようなケースでは間に合わない

・この解法を高速化できないか？

解法①

・Aが1を含まなければ、$2^{30} > 10^9$より、右端を伸ばす方のループは高々30回しか回らない

・よって、Aからあらかじめ1を取り除いておいてから先ほどの解法を使うとO(NlogP)となり、間に合う

解法②

・部分列の左端を決めた時、Aに負の要素は存在しないので、右端を伸ばしていくと積は単調増加する

・尺取り法が使える！

・尺取り法とは、ある連続する部分列が条件を満たす間右端を伸ばし、満たさなくなったら満たすようになるまで左端を縮める、というようなアルゴリズム

・解法①とやっていることはほとんど同じ

サンプルコード

・尺取り法については、サンプルコードやABC032-C、ABC038-Cの解説を参考にするといいと思います

・サンプルコード(解法①)

・サンプルコード(解法②)

巨大チェスボード 解説

技術室奥プログラミングコンテスト #3

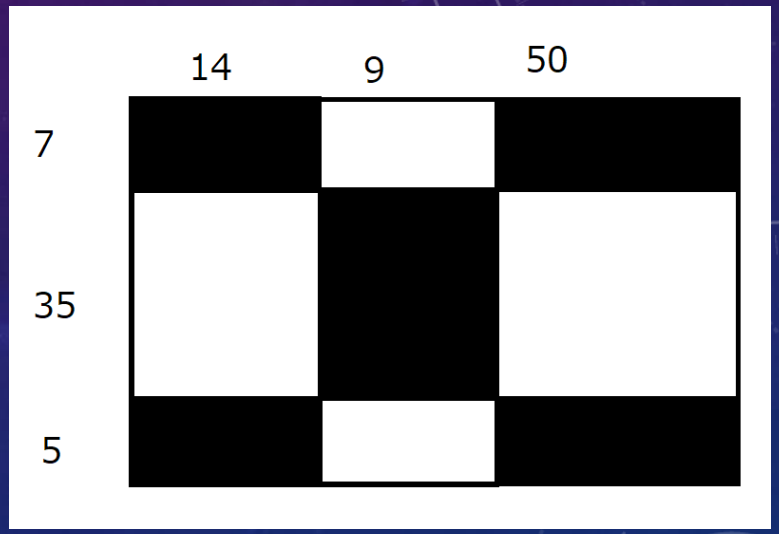
Ryo2016(@Zen_Re_Kkyo)

問題概要

- マス目があり、上から i 番目の縦の長さはH_i ,
左から j 番目の横の長さはW_j のグリッドである。

- マス目は市松模様に塗られており、左上は黒である。

- クエリがたくさん与えられるので、その長方形の中に
存在する(黒マスの面積の総和) – (白マスの面積の総和) を
クエリごとに求めよ



部分点解法[1] ―30点―

部分的解法[1] ―30点―

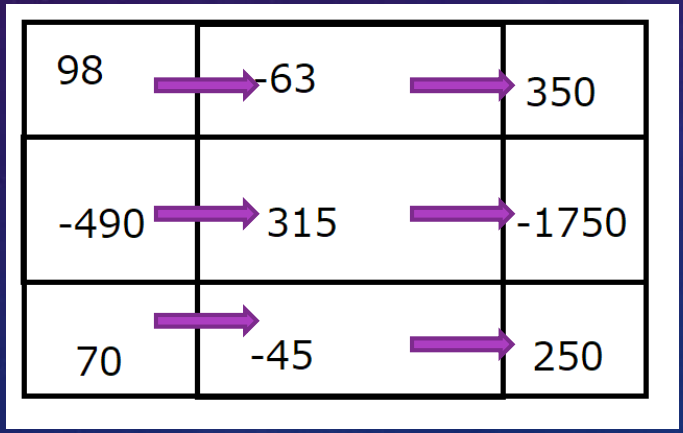
- H,W≦1000 → 配列にマスの面積を全て保存できそう!
- 白マスの部分は面積をマイナスにすると実装がやりやすい

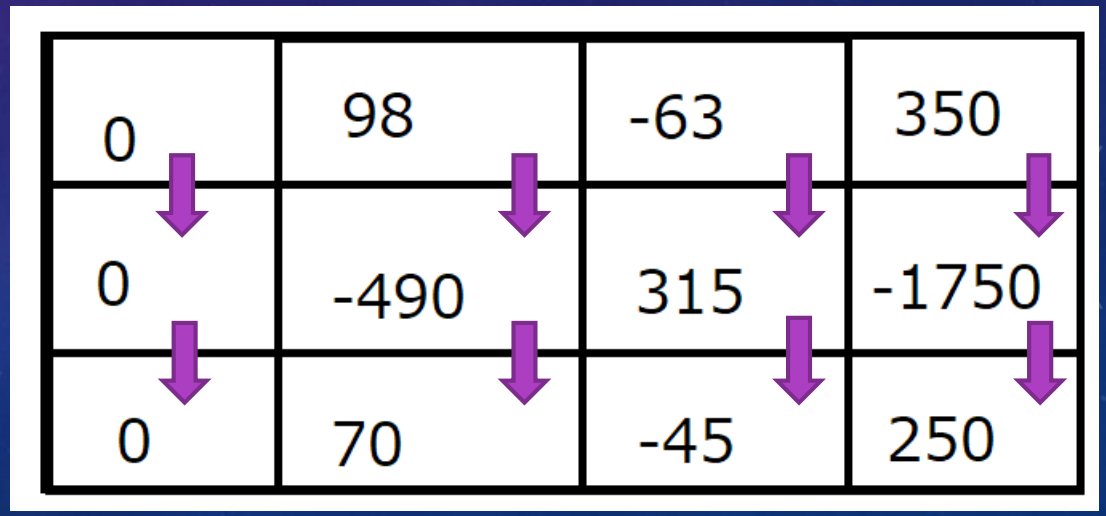
- Q≦5 ということは、区間の面積をその都度全て足しても間に合う
- 計算量はO(HWQ)で、Q≦5なので十分間に合う

部分点解法[2] ―150点―

部分点解法[2] ―150点―

- Q≦100,000 →O(HWQ)ではダメ
- 考察… 部分点[1]では区間の面積をいちいち求めていた
  - 累積和を使えば計算量を落とせそう！！！





部分点解法[2] ―150点―

- このように加算を行うと、次のような累積和配列ができる
- (px,py)~(qx,qy)までの和を求めたいときは

包除の原理を使えばいいので、　　…(*1)

1クエリごとにO(1)で求められる

→O(Q)で解けた！

- (*1) 例えば緑枠を求めたいときは、

(桃色)-(黄色)-(灰色)+(青)で求められる

ABC086-Dなどを参照

0	0	0	0
0	98	35	385
0	-392	-140	-1540
0	-322	-115	-1265

満点解法[3] ―400点―

満点解法[3] ―400点―

- 制約がH,W≦100,000になった

- 部分点[2]でO(Q)にできたので変わらず… **というわけにはいかない！**

理由:面積の累積和を求める時点でH*Wの計算量を使っているから

- ここで考察…縦*横を縦と横に分解できないか?

理由:長方形の面積は縦*横なので縦の総和と横の総和さえ分かればいい

→面積が負の時がすごくやっかい…

満点解法[3] ―400点―

- ある行を取り出して考えてみる　(高さは正と仮定)

2	7	18	28	1	8	2

- このうち、黒マス==面積が正なのは?→黄色の部分

- 偶数行では、黄色の部分は白マス==面積は負

満点解法[3] ―400点―

2	-7	18	-28	1	-8	2

- このように横の長さの符号を偶数番目だけ反転してみた

- 奇数行… 黒白黒白…と並びそう
- 偶数行… 黒白黒白…と並ぶ→ここをさらに反転したい！

- 解決には… 「偶数行目の高さの符号を反転する!」

満点解法[3] ―400点―

- 結局、「偶数列と偶数行の符号を反転すればよい」ことがわかる(下の表)

縦/横	+	-	+
+	+	-	+
-	-	+	-
+	+	-	+

- あとは符号を反転した縦と横の数列で

累積和を取ってやればいい

計算量は前処理にO(H+W),クエリにO(Q)なので全体でO(Q),よって間に合う。

おわりに

- Atcoder歴約20カ月にして初めての作問でした。いきなり400点って…
- 個人的にはこの問題は400適正かやや易だと思っています。
  部分点[2]は二次元累積和の練習に使えると思います。

参加ありがとうございました！

このまま500点も解説するんですけどね…

デフレゲーム　解説

技術室奥プログラミングコンテスト #3

Ryo2016(@Zen_Re_Kkyo)

問題概要

- 全ての面が等確率に出る1~nが書かれたサイコロがある
- 今まで出たことのある面が2度目に出るまでサイコロを投げ、その目の和がもらえる金額である
- もらえる金額の期待値を求めよ

- 制約: [1]n≦8
  [2]n≦18
  [3]n≦500,000

小課題[1],[2] 60点,180点

小課題[1],[2] 60点,180点

- [1]は目の出方を全探索でO(N^N)

- [2]はbitDPでO(2^N)
- 面の集合Aが全て出たときの確率は求まるのでその場合の数を確率で割ればよい

- 部分点は考察上にてさして重要ではないのでこのくらいで。

満点解法 —500点—

満点解法 ―500点―

- 「k面が出た状態でゲームが終了する」ということを考える
- まず、n面中k面が出た状態での出た目の和の期待値は?
  - $k\frac{n+1}{2}$ になる (理由:1面の期待値は$\frac{n+1}{2}$なので、期待値の線形性から)
- これで**出た面の種類について考えなくて良くなった**

満点解法 ―500点―

- 次に、k面でゲームが終了する確率を求める

- k面すべてが違う面で、最後の1面だけ今までと同じ面が出ているので、

求める確率は $\left(\frac{n}{n} \times \frac{n-1}{n} \times \cdots \frac{n-k+1}{n} \times \frac{k}{n}\right)$ となる

- よって、各kに対して期待値と確率を掛け合わせてやればいい

- 確率の計算をメモ化すればO(n)で各kに対する答えがすべて求まる

(1回1回求めていると$O(n^2)$になってしまう)

- よって、O(n)でこの問題が解けた

おわりに

- 個人差問題かな…と思って500を付けました。

ご参加いただきありがとうございました！

引き続きプロの作問した問題をお楽しみください!

F 問題: 天使とふすま

技術室奥プログラミングコンテスト #3 解説

WRITER: YNYMXIAOLONGBAO

問題概要

- 実は、問題文に出てくる天使のTさんは問題とは まったく関係がありません。Writerの趣味です。

- 天使の T さんが出てくると集中できないので、問題を式で言い換えます。
- 数列 a, b と $1 \sim n$ の順列 p があって、p を適切に並び替えることによって次の式を最小化せよ。

- $\sum_{i=1}^{n}(b_{p_i} * \sum_{j=1}^{i-1} a_{p_j})$

部分点解法 (200 点)

- $N <= 10$ と小さいので、順列 p を全列挙できます。
- 式を計算しながら深さ優先探索をすることで、$O(N!)$ の計算で解くことができます。
- またnext_permutationなどで順列を挙げて後から式を計算すると $O(N! * N)$ になりますが、この方法でも間に合います。

考察

- 式をみると、p のいくつかの連続する項を並び替えたとしてもそれ以前、それ以降の計算には関係無いことが分かります。
- そこで、p の隣り合う項を入れ替えることを考えます。
- 計算すると、並び替える前と並び替えた後の値は次のようになります。

- 並び替える前　$b_{p_{k+1}} * a_{p_k} + t$
- 並び替えた後　$b_{p_k} * a_{p_{k+1}} + t$
- 詳しい計算は省略しますが、t は一意の整数値です。当然、この値が小さい方が良いです。

考察

- 式を変形すると、隣り合う二項間では $\frac{b_i}{a_i}$ が小さい方が後になれば良いことが分かります。
- それなら、全体は $\frac{b_i}{a_i}$ の降順にソートされるはずです。このことは、背理法で簡単に示すことができます。

満点解法

- $\frac{b_i}{a_i}$ の降順（$= \frac{a_i}{b_i}$ の昇順）にソートして、その順にふすまを並べれば良いです。
- 計算量は $O(NlogN)$ になります。

- たぴたぴ～

G問題 パソコンの買い替え

技術室奥プログラミングコンテスト#3 解説

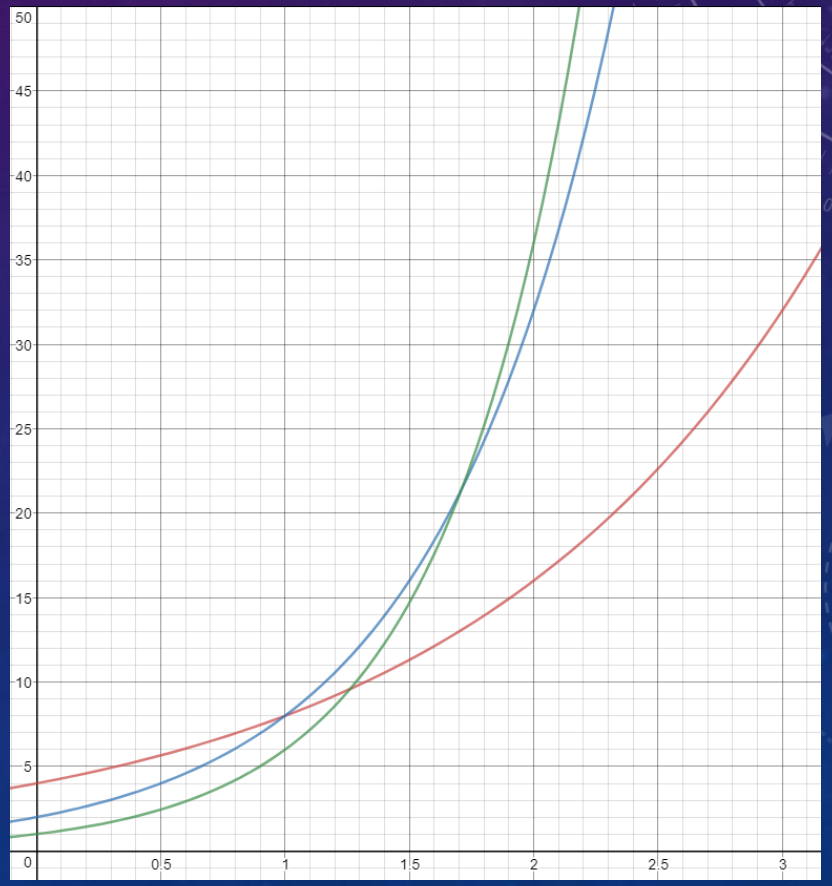
writer: autumn_eel

問題概要

- $y = a * b^{x-r}$の形をしたN個の指数関数の式が与えられる
- K回それぞれにおいて, $x = y_i$ の時に最大値をとる式のうち1個を選ぶ
- 合計で最大何種類の式を選ぶことができるか?

問題概要

- 例: サンプル1



部分点解法(200点)

- y_i年後において, それぞれの式の値の大小を比較できれば良い
- 愚直に計算すると巨大な数になってしまう

→logを取ろう!!

部分点解法(200点)

- 適当な値 d $(d > 1)$ でlogを取ると, $a \geq 1, b \geq 2$ より $y = a * b^{x-r}$の式を $y = log_d(b)\,(x - r) + log_d(a)$ に変換できる

- logを取っても大小関係は変化しないので,これで比較をすれば良い

部分点解法(200点)

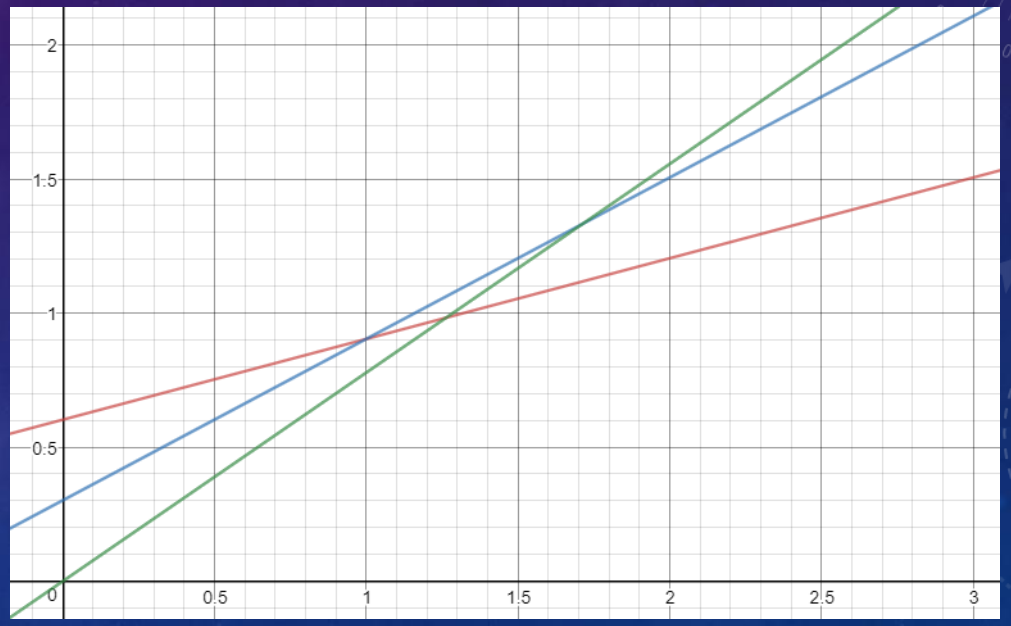
- ある y_i において,最大値をとる式が複数ある場合はどうすれば良いか?
- 今まで使ったことのない式を優先して使い,それでも複数ある場合は b が最も小さいものを使う
- 計算量は$O(NK)$

→200点

満点解法(700点)

- logを取った後の式のグラフを観察してみる
  →直線がたくさん

- 一次関数の式になっているので当たり前



満点解法(700点)

- たくさんの直線から最大値を効率的に求めるにはどうすれば良いか?

→Convex Hull Trick

満点解法(700点)

- 解説記事があるので詳細は調べてください
- これを使うことによってK回合わせて$O(N)$で最大値を求めることができる
- 誤差には注意
- 傾きが同じ直線は1種類しか使わないのでまとめておくと実装が楽
- 計算量は$O(N)$くらい

→700点

H: 新入生歓迎数列 - HARD

技術室奥プログラミングコンテスト #3 解説

WRITER: E869120

問題概要

- 長さ N の数列 P がある。最初全ての要素は 0 である。
- あなたは以下の操作を合計で 550000 回以内することができる。
  - P_iを 1 に変更
  - P_iに P_jの値を足す
  - の 2 つのうちどちらかをやる、i, j は適当に選んでよい
- 数列 P を $A = \{A_1, A_2, A_3, \ldots, A_N\}$ と一致させてください。

問題概要

- 例えば、入力例 1 の場合 （$N = 2, A = \{1, 4\}$）

- 4 回の操作でできる
  - [0, 0]
  - [1, 0] → A[1] を 1 に変更
  - [1, 1] → A[2] を 1 に変更
  - [1, 2] → A[2] に A[2] の値を足す
  - [1, 4] → A[2] に A[2] の値を足す

全体の展望

- 小課題 1 [60 点] → $N = 2$
- 小課題 2 [140 点] → $N \leq 8\,000$（目安は $2\ logA_i * N$）
- 小課題 3 [80 点] → $N \leq 16\,000$（目安は $log\ A_i * N$）
- 小課題 4 [230 点] → $N \leq 25\,000$（目安は $\frac{1}{2} log\ A_i * N$）
- 小課題 5 [270 点] → $N \leq 160\,000$ （目安は $3N$ + 定数）
- 小課題 6 [220 点] → $N \leq 200\,000$ （目安は $2.5N$ + 定数）

- 全体の操作回数の上限: 550 000 回, $1 \leq A_i < 2^{30}$

小課題 1 （累積 60 点）

- まず、P_1 の値をどんどん 2 倍していくことを考える （2 倍することは、P_1に P_1 の値を足すことによって操作できる）
  - 1 → 2 → 4 → 8 → 16 → …
- 2 倍していく中で、適切なタイミングで「P_2 に P_1 の値を足す」操作を行う。A_2 は 2 のべき乗 $log\ A_i$ 個以内の和で表せるので、操作回数的には全く問題がない。（例：$A_2 = 14$ のとき、$2 + 4 + 8$ という形のべき乗の和で表せる）
- 制約より $P_1 = 1$ なので、最後に「P_1 を 1 にする」という操作を忘れずに。

小課題 1（累積 60 点）

P_1	P_2	備考
0	0	
1	0	P_1 を 1 にした。
2	0	P_1 に P_1 を足した。（2 倍した）
2	2	P_2 に P_1 (2) を足した。
4	2	P_1 に P_1 を足した。（2 倍した）
4	6	P_2 に P_1(4) を足した。
8	6	P_1 に P_1 を足した。（2 倍した）
8	14	P_2 に P_1 (8) を足した。
1	14	P_1 を 1 にした。

60 点

小課題 2（累積 200 点）

- $N \leq 8\,000$
- 実はこの小課題は $N = 2$ の場合に帰着できる！
  - まず、$P_1 = 1$ にした後 P_2 に対して $N = 2, A = \{A_1, A_2\}$ のつもりで操作をする
  - 次に、$P_1 = 1$ にした後 P_3 に対して $N = 2, A = \{A_1, A_3\}$ のつもりで操作をする
  - 次に、$P_1 = 1$ にした後 P_4 に対して $N = 2, A = \{A_1, A_4\}$ のつもりで操作をする
  - （続く）

- 小課題 1 の操作回数が最大で $1 + 30 + 30 + 1 = 62$ 回なので、多くても $62 * 8\,000 = 496\,000$ 回の操作が必要 → 間に合う

200 点

考察 1

- $N \leq 16000$
- なので、操作の回数を 2 分の 1 にしなければならない

- 小課題 1 の操作には、「P_1 を倍にする (30 回)」と、「P_2 に P_1 の値を足す (30 回)」 という操作の 2 種類が主にある
- 前者を削れないか？

考察 1

- とりあえず、数十回以内の操作で $P_1 = 1, P_2 = 2, P_3 = 4, \dots, P_{30} = 2^{29}$ になったとする （Step 1）
- そうすると、P_{31} 以降は $P_1, P_2, \dots, P_{30}$ のうち 30 個以内の和で表すことができるので、P_{31} から P_N までは $30N$ 回の操作でできる！（Step 2）

考察 1

- じゃあ $P_1 \sim P_{30}$ の値はどうやって元に戻し、最終的に $A_1 \ldots A_{30}$ にしていくか？
  - まず、$P_1 \sim P_{30}$ の値を全て 1 に変更する
  - その後、小課題 2 と同じ操作を $P_1 \sim P_{30}$ に対して行う。$P_i = A_i$ になるためには、P_i に合計でちょうど $A_i - 1$ を足す必要があることに注意せよ。（最初の時点での $P_i = 1$ だから）
  - 一連の操作を Step 3 とする

$P_1 \sim P_{30}$ 2 進数がポイント 1, 2, 4, 8, 16, 32, 64, 128, …	$P_{31} \sim P_N$ $P_1 \sim P_{30}$ の値を使って 30 回で A_i を作る

小課題 3 (累積 280 点)

- Step 1 は,「A_i に A_{i-1} の値を足す」「A_i に A_i の値を足す」の 2 回の操作の繰り返しでできるので、高々 60 回
- Step 2 は、$N = 16000$ のとき高々 $30 * (16000 - 30) = 479100$ 回
- Step 3 は、1 に変更する操作 30 回と、小課題 2 と同じ操作 $30 * 60 = 1800$ 回が必要なので、合計で高々 1830 回
- 全てを合計すると、$60 + 479100 + 1830 = 480990$ 回 → 間に合う

280 点

考察 2

- $N \leq 25000$ → 基本的に、各 P_i を A_i にするのに 15 回くらいしか使えないと考える
- 今までは、$1 \leq \mathrm{A_i} < 2^{30}$ を満たす全ての A_i が $P_1, P_2, P_3, \ldots, P_{30}$ までの 30 個以内の部分和で表されていたので、基本的に各 P_i を A_i にするのに高々 30 回くらいかかった
- → だから、15 個以内の部分和で表したい！
- 従来 2 進法だったのを、4 進法にすることを考える

2 進法 (高々 30 個の和)

↓

4 進法 (高々 15 個の和)

考察 2

- 4 進法にするということは、$P_1 \dots P_{60}$ が以下のようになるように操作を行い (step 1), ここから P_{61} 以降に対して操作を行う (step 2)

$P_1 = 0, P_2 = 1, P_3 = 2, P_4 = 3$

$P_5 = 0, P_6 = 4, P_7 = 8, P_8 = 12$

$P_9 = 0, P_{10} = 16, P_{11} = 32, P_{12} = 48$

$P_{13} = 0, P_{14} = 64, P_{15} = 128, P_{16} = 192$

- そうすると、A_i を $P_1, P_2, \dots, P_{60}$ のうちの 15 個以内の部分和で表すことができるので、（上図でいう各色から一個ずつ取っていく感じ）各 P_i につき 15 回以内の操作で $P_i = A_i$ にできる

小課題 4 (累積 510 点)

- 前ページに述べた step 1 については、以下のような操作の手順で $P_1, \dots P_{60}$ を目標の値にすることができる。
  - まず、各 a について操作を行う。ただし a を 4 で割って余り 2 になるものとする。$P_a = 4P_{a-4}$ なので、「P_a に P_{a-4} を足す」「P_a に P_a を足す」「P_a に P_a を足す」の 3 回の操作で P_a の値を目標通りにすることができる。
  - 次に、$a + 1 \leq x \leq a + 2$ を満たす各 x に対して操作を行う。$P_x = P_{x-1} + P_a$ なので、既に作った 2 つの整数の和で表されるということになる。よって、2 回の操作で P_x の値を目標通りにすることができる。

- よって、回数としては $15 * 3 + 30 * 2 = 105$ 回の操作で $P = \{0, 1, 2, 3, 0, 4, 8, 12, 0, 16, 32, 48, \dots\}$ とすることができる。

小課題 4 (累積 510 点)

- Step 2 に関しては、高々 $15 * (25000 - 60) = 374100$ 回
- Step 3（一度 A_i ではない値に変えられた $P_1, P_2, \ldots, P_{60}$ を A_i に直す操作）は、小課題 3 の方針を使うと 2790 回

- よって、合計操作回数は $105 + 374100 + 2790 = 376995$ 回
  - → 間に合う

$P_1 \sim P_{60}$ 4 進数がポイント 0, 1, 2, 3, 0, 4, 8, 12, 0, 16,…	$P_{61} \sim P_N$ $P_1 \sim P_{60}$ の値を使って 15 回で A_i を作る

510 点

小課題 5 (累積 780 点)

- 4 進数ではなく 1024 進数を使う事を考える
  - つまり、$P_1 = 0, P_2 = 1, \ldots, P_{1024} = 1023, P_{1025} = 0, P_{1026} = 1024, P_{1027} = 2048 \ldots$
  - それを作るのにも、「$P_a = 1024P_{a-1024}$」「$P_x = P_{x-1} + P_a$」（a は 1024 で割って 2 余る、$a + 1 \leq x \leq a + 1022$）の性質を使うと高々 $3 * 3 + 2 * 3066 = 6141$ 回

- 各 A_i は 3 個以内の $P_1, \ldots P_{3072}$ の部分和で表すことができるので、$P_{3073} \sim P_N$ の範囲はそれぞれ高々 3 回の操作でできる

$P_1 \sim P_{3072}$ 1024 進数がポイント 0, 1, …, 1023, 0, 1024, 2048, …	$P_{3073} \sim P_N$ $P_1 \sim P_{3072}$ の値を使って 3 回で A_i を作る

小課題 5 (累積 780 点)

- $P_1, P_2, P_3, \ldots, P_{3072}$ を $A_1, A_2, A_3, \ldots, A_{3072}$ にするためには、まず全部P_i を 1 にしてから、小課題4 の方針を使うと 48075 回でできる
  - → 全部合わせて $6141 + 470784 + (3072 + 48075) = 528072$ 回の操作 → 間に合う
- もっと良い方針があって、小課題 4 の方針だと 4 進数だが、これを 32 進数でやると、最後の部分は $3072 + 384 + 6 * (3072 - 192) + 15 * (192 - 30) + 30 * 30 = 24066$ 回でできる
  - （この方針は 1000 点解法で使います。）

780 点

考察 3

- $N \leq 200000$ → $3N$ だと無理で、多くの A_i を 2 回で作らなければならないことになる。
- とりあえず、数列の最初の部分を以下のようにすることを仮定する

$P_1 \sim P_{3072}$ 780 点解法と同じ 0, 1, …, 1023, 0, 1024, …	$P_{3073} \sim P_{19456}$ 順に 0, 1, 2, 3, …, 16383	$P_{19457} \sim P_N$ これまでに作られた値を基にして、その和で A_i を作っていく

考察 3

$P_1 \sim P_{3072}$ 780 点解法と同じ 0, 1, …, 1023, 0, 1024, …	$P_{3073} \sim P_{19456}$ 順に 0, 1, 2, 3, …, 16383	$P_{19457} \sim P_N$ これまでに作られた値を基にして、その和で A_i を作っていく

- 2 回で $P_i = A_i$ $(19457 \leq i \leq N)$ にできる条件とは？
  - $P_{3073} \sim P_{19456}$ の緑の領域を使えること
  - つまり、P_x $(19457 \leq x \leq N)$ の中で $P_i - 16383$ 以上 P_i 以下の数が 1 個でもあればよい

- よって、$P_{19457} \sim P_N$ の領域は A_i の小さいインデックスから埋めればよい！

考察 3

- A_i の小さいインデックスから埋めるので…?
  - 現在 P_x を埋めているとき、A_{19457} ～ A_N の中で、A_x より小さいものの中で最大の数を B とすると $A_x - B \leq 16383$ のときに 2 回の操作で $P_x = A_x$ にできる
  - 逆にそうではない場合 3 回かかるが、このような x の個数は高々 $2^{30} \div 16384 = 65536$ 個, つまり $N = 200000$ とすると $200000 - 19456 - 65536 = 115008$ 個の x は 2 回の操作でできる

- 言っていることは、$1 \leq Q_1 \leq Q_2 \leq \cdots \leq Q_{N-1} \leq Q_N < 2^{30}$ を仮定したときに、$Q_{i+1} - Q_i \geq 16384$ となる i の個数が 65536 個以下、ということだけ

満点解法

- まず、$P_1 \sim P_{19456}$ を適切に作るためには, 1 個のインデックスにつきほとんど 2 回の操作が必要なので $19456 * 2 = 38912$ 回
  - P_{3072} までは「$P_a = 1024P_{a-1024}$」「$P_x = P_{x-1} + P_a$」（a は 1024 で割って 2 余る、$a + 1 \leq x \leq a + 1022$）の性質を使える、$P_{3073}$ 以降もおなじような性質を使うことができる

$P_1 \sim P_{3072}$ 780 点解法と同じ 0, 1, …, 1023, 0, 1024, …	$P_{3073} \sim P_{19456}$ 順に 0, 1, 2, 3, …, 16383	$P_{19457} \sim P_N$ 全部 0

満点解法

- 次に、$P_{19457} \sim P_N$ について、考察 3 で求めたように N = 200000 のとき、2 回操作が必要なのが 115008 個、3 回操作が必要なのが 65536 個なので $2 * 115008 + 3 * 65536 = 426624$ 回

- 残りは $P_1 \sim P_{19456}$ を頑張って求めるだけ

$P_1 \sim P_{3072}$ 780 点解法と同じ 0, 1, …, 1023, 0, 1024, …	$P_{3073} \sim P_{19456}$ 順に 0, 1, 2, 3, …, 16383	$P_{19457} \sim P_N$ 全部 A_i（正しい値）

満点解法

- $P_1 \sim P_{19456}$ は、小課題 5 の方針を使えば $P_i = A_i$ にできる
  - 必要な回数は $24066 + (19456 - 3072) * 3 = 73218$ 回

$P_1 \sim P_{3072}$ 全部 A_i （正しい値）	$P_{3073} \sim P_{19456}$ 全部 A_i （正しい値）	$P_{19457} \sim P_N$ 全部 A_i（正しい値）

- 合計で $38912 + 426624 + 73218 = 538754$ 回 → 間に合う
  - どのようなケースでも 538754 回以下

1000 点

補足

- 小課題 6 は $N \geq 19457$ でしか通用しない
- 小課題 5 は $N \geq 3073$ でしか通用しない　ので、

- N によって解法を場合分けする必要があることに注意してほしい。
  - $N \leq 3072$ のとき、小課題 2 の解法
  - $N \leq 19456$ のとき、小課題 5 の解法
  - $N > 19456$ のとき、小課題 6 の解法

- 実装や定数倍を多少さぼると、1000 点ではなく 780 点になったりすることもあり得ます。

サンプルコード

- 以下、C++ のサンプルコードです。使う領域の境界値などが若干解説と違うのでご注意ください。

- 満点解法 : こちら … 実行時間 1039 ミリ秒, 実装量 5042 バイト
- 780 点解法 : こちら … 実行時間 1344 ミリ秒、実装量 2088 バイト
- 280 点解法 : こちら
- 200 点解法 : こちら
- 60 点解法 : こちら

I 問題：王国と M 種類の店

技術室奥プログラミングコンテスト #3 解説

WRITER: E869120

問題概要

- N 頂点の木がある。頂点 i $(2 \leq i \leq N)$ は P_i と長さ L_i で結ばれている。
- 各頂点にはちょうど 1 個店がある。頂点 i にある店の種類は R_i $(1 \leq R_i \leq M)$ である。
- 頂点 i の不便さは、頂点 i からの (最寄りの種類 1 の店までの最短距離) + (最寄りの種類 2 の店までの最短距離) + … である。
- その時、全ての頂点の不便さを求めよ。

- ただし、入出力サイズが大きいので cin/cout ではなく scanf/printf が推奨される。

問題概要

- 小課題 1 [55 点] → $N \leq 1500, M \leq 750$
- 小課題 2 [110 点] → $N, M \leq 70000, N = M$
- 小課題 3 [275 点] → $N, M \leq 70000$, 同じ種類の店は高々 8 個しか存在しない
- 小課題 4 [330 点] → $N, M \leq 70000$
- 小課題 5 [330 点] → $N, M \leq 400000$

小課題 1 (55 点)

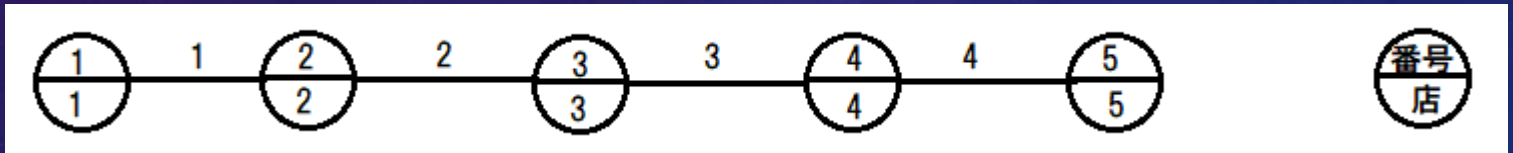
- まず、全頂点に対して、「ある特定の種類の店に行くための最短距離」を求めることを考えます。
- 頂点 i からの最短距離を $dist_i$ とするとき、「ある特定の種類の店」がある頂点に対しては $dist_i = 0$ とし、ダイクストラ法をすると解くことができます。
- ダイクストラ法の計算量は $O(N \log N)$ であり、これを M 回やる必要があるので、計算量は $O(NM \log N)$ となります。$N \leq 1500, M \leq 750$ なので、余裕で間に合います。

- ちなみに、ダイクストラ法について知らない人は こちら をご覧ください。

考察 1

- 小課題 2 は、N = M ですので、全ての種類の店が 1 個ずつ存在します。

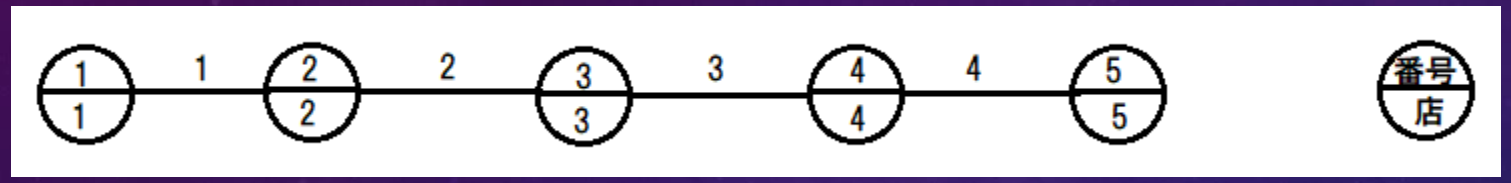
- この特徴を活かして、全ての辺 (u, v) に対して、$ans_u - ans_v$ を求めてみましょう。ただし、ans_i を頂点 i の不便さとします。
- 例えば、木が右図のような場合、$ans_u - ans_v$ の値は、以下のようになります。



- $ans = \{20, 17, 15, 18, 30\}$ より、
- 差分は $(1, 2) \to 3, (2, 3) \to 2, (3, 4) \to -3, (4, 5) \to -12$
- **長さ 1 当たりの差分は $(1, 2) \to 3, (2, 3) \to 1, (3, 4) \to -1, (4, 5) \to -3$**

↑なぜこのようになるのでしょうか？

小課題 2 (110 点)



- 頂点 1 から頂点 2 に向かうと、距離はどうなるか？
  - 頂点 1 からの距離は遠くなる
  - 頂点 2, 3, 4, 5 からの距離は近くなる → 近くなる: 4, 遠くなる: 1 なので、$(ans_1 - ans_2)$ ÷辺の長さ = (4 – 1) = 3

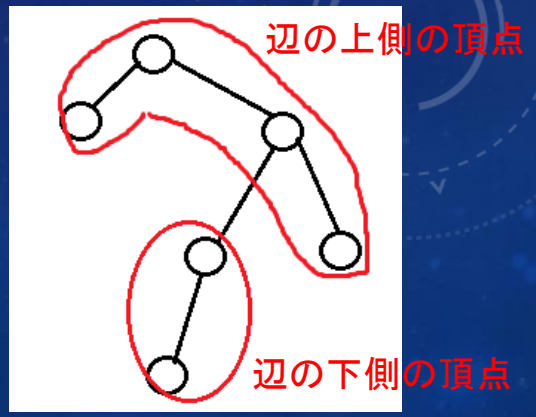
↑つまり、辺の中を 1 動くと不便さは 3 変わる

- 頂点 2 から頂点 3 に向かうと、距離はどうなるか？
  - 頂点 1, 2 からの距離は遠くなる
  - 頂点 3, 4, 5 からの距離は近くなる→ 近くなる: 3, 遠くなる: 2 なので、$(ans_1 - ans_2)$ ÷辺の長さ = (3 – 2) = 1

↑つまり、辺の中を 1 動くと不便さは 1 変わる

小課題 2 (110 点)

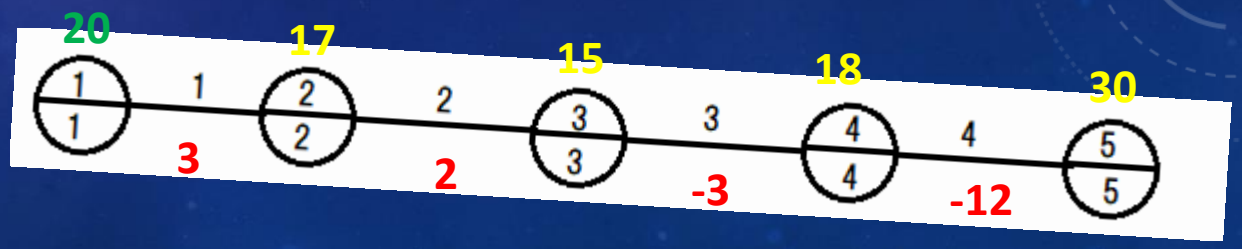
- よって、各辺に対して、長さ 1 当たりの不便さの差分（辺内を長さ 1 動くことによって下がる不便さの値）は（近くなる頂点の数）-（遠くなる頂点の数）
- これを言い換えると、(辺の左側の頂点の数) – (辺の右側の頂点の数) となります。
- 頂点 1 を根とする根付き木にすると、頂点 i $(1 \leq i \leq N)$ を根とする部分木に含まれる頂点数が DP によって求められるので、(辺の下側の頂点の数) が分かります。(辺の上側の頂点の数) = N – (辺の下側の頂点の数) なので、計算により「この辺における、長さ 1 当たりの不便さの差分」が求まります。
- 長さ 1 当たりの不便さの差分が求まったら、各辺 (u, v) に対して、$ans_u - ans_v$ が求まるはずです。（辺の長さ L を掛けるだけです）



小課題 2 (110 点)

- 各辺 (u, v) に対して $ans_u - ans_v$ が求まれば、あとは頂点 1 の不便さが求まるだけで全ての頂点の不便さが求まります。(上から累積的に計算すれば)
- 頂点 1 の不便さは $dst(1,1) + dst(1,2) + dst(1,3) + \ldots + dst(1,N)$, ただし $dst(u,v)$ は頂点 u から v までの最短経路の長さです。そのため、頂点 1 の不便さは DFS などにより $O(N)$ で求めることができます。
- 例えば、右の木における頂点 1 の不便さは 20 なので、木の根から累積的に計算すると、各頂点の不便さは {20, 17, 15, 18, 20} となります。

- 計算量は $O(N)$ です。
- 小課題 1 と一緒に取ると 165 点がもらえます。



小課題 3 以降の全体の方針

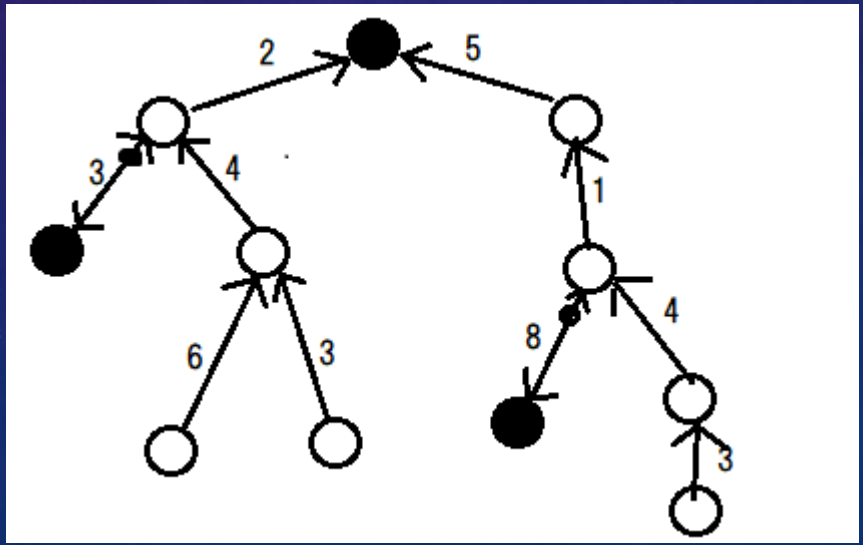
- 今度は、N が大きく、M などに特殊な制約はありません。ここで全体の方針から先に説明します。

- 【全体の流れ】
- 基本的に、各店の種類に対して**「どの店が一番近いか、が変わる点」**）」を列挙し、$ans_u - ans_v$ の差分を更新していく感じにする。仮にこの種類の店が K 個あるのであれば、$O(K \log K)$ で済ませる。全体の計算量は $O(K \log K)$ である。
- 各辺 (u, v) [u が v より頂点 1 に近い] に対して、$ans_u - ans_v$ の値が求まれば、あとは頂点 1 の不便さを求めるだけで、全部の辺の不便さが求まる。（小課題 2 で既に言及した通り）

小課題 3 以降

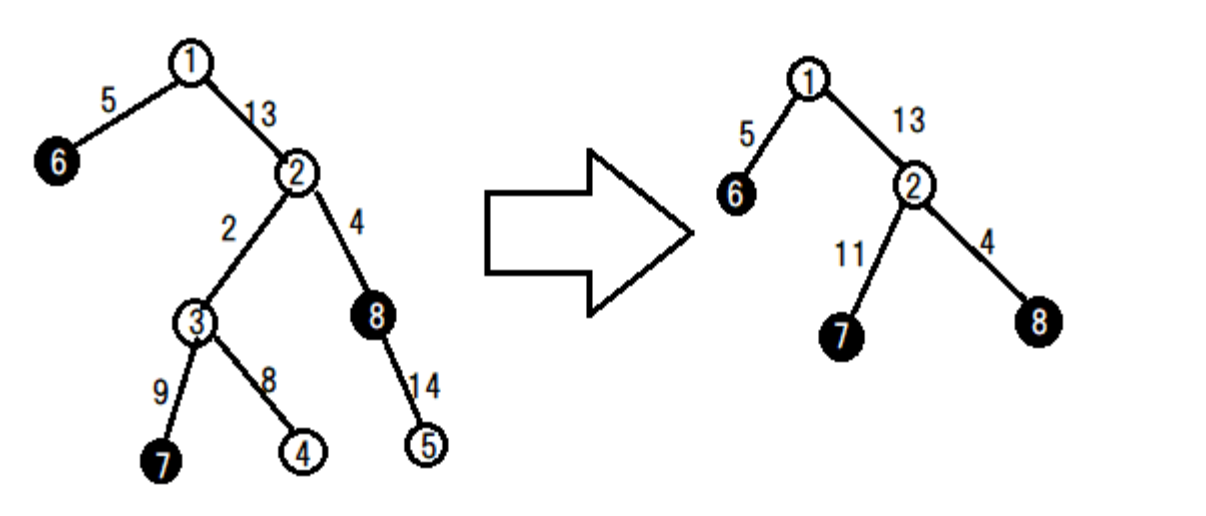
- とりあえず、種類 1 の店について考える（種類 2 以降も同様にできるため）

- 種類 1 の店を全て黒く塗りつぶした場合、例えば状態は以下の図のようになる。（矢印はどちらの方向の店に近いかを表す） 頂点 1 を根とした時の辺 (u, v) の矢印が下向きの場合、当然最寄りの種類 1 の店までの最短距離は v の方が小さい。逆も言える。つまり、結局は矢印の向きによって $dist_u - dist_v$ の値が決まってくる。



小課題 3 以降

- 結局は、前述の「矢印の向き」をどう求めるか、また矢印の向きが変わるところはどこなのかを求めなければならないのです。
- この手法の一つとして、「木を圧縮させる」 という方針が挙げられます。（以下の図のように） ここでいう「圧縮」というのは、黒い頂点がどのように結ばれているかの情報を保持したまま、頂点数が最小になるようにグラフを持つことを指します。（黒い頂点の数）≧（白い頂点の数）になります。
- 果たしてどうやって圧縮させるのでしょうか？

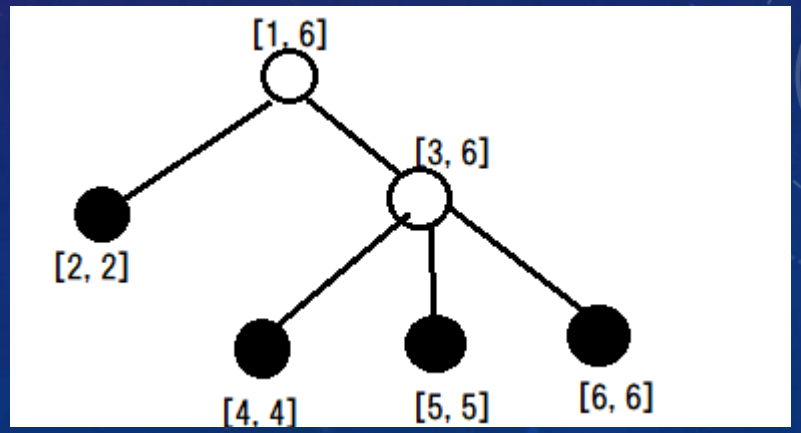


木の圧縮 @PHASE 1

- まず、圧縮された木に含まれる頂点を全部列挙します。
  - 当然、黒い頂点は含まれます。
  - 黒い頂点を、オイラーツアーの早い順に $v_1, v_2, v_3, \ldots, v_I$ とするとき、$LCA(v_1, v_2), LCA(v_2, v_3), LCA(v_3, v_4), \ldots, LCA(v_{I-1}, v_I)$ は含まれます。しかし、オイラーツアーの性質上、$LCA(v_A, v_B)$ は $LCA(v_A, v_{A+1}), +LCA(v_{A+1}, v_{A+2}), \ldots, LCA(v_{B-1}, v_B)$ のどれかなので、これ以外の LCA を調べる必要はありません。
  - あと、矢印の方向を求める都合上、頂点 1 は含まれた方がいいです。

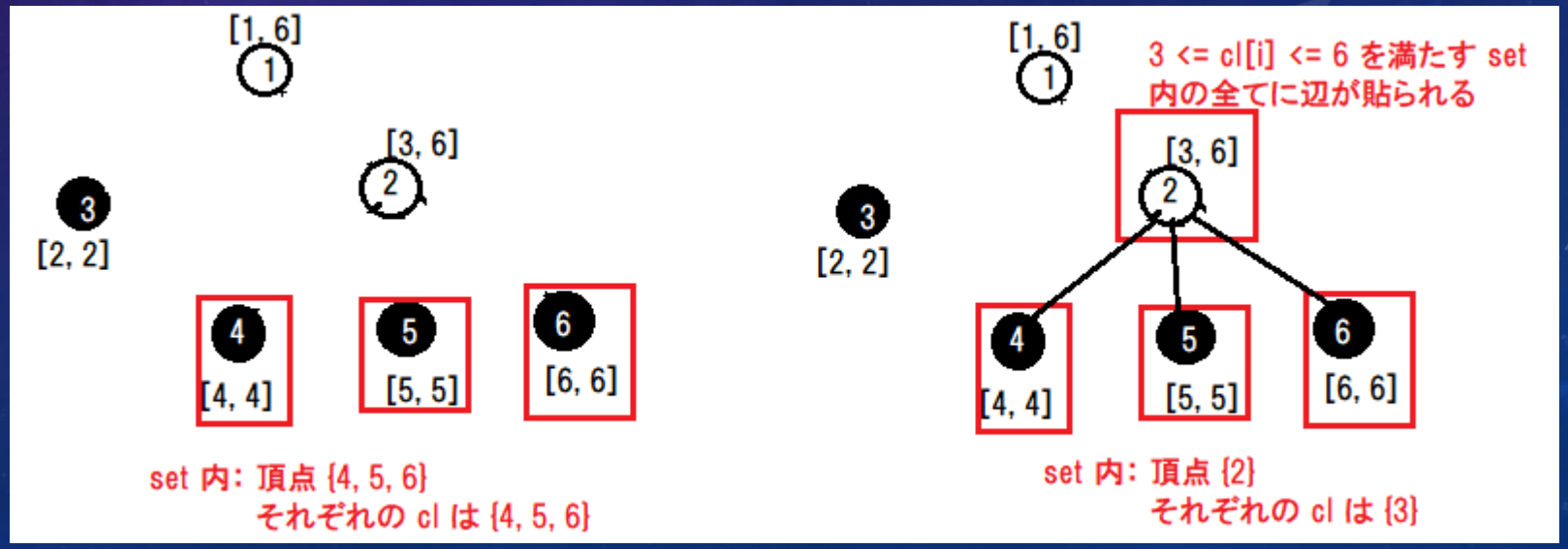
木の圧縮 @PHASE 2

- 次に、圧縮された木に含まれる頂点の情報から、辺の情報を求めます。
- まず、オイラーツアーにより、各頂点に対して [順番の下限, 順番の上限] を求めます。頂点 i に対する順番の下限を cl_i, 順番の上限を cr_i とします。
- 次に、オイラーツアーの順番のある性質を使います。
  - j は i の部分木に含まれる ⇔ $cl_i \leq cl_j \leq cr_j \leq cl_i$



木の圧縮 @PHASE 2

- 前述の性質を使い、根から遠い頂点から順番に見ていき、辺を貼っていきます。これには、set を使います。set には、「既に見た頂点であるが、上方向に辺が繋がれていないもの」が入ります。
- set には、cl_i の値も記録します。そして、頂点 x から辺を貼る際には、set の中に入っている頂点の中で $cl_x \leq cl_i \leq cr_x$ を満たしている全ての i に対して $x \rightarrow i$ の辺を貼ります。（set の中での i の開始位置は、cl_i で二分探索をすることによって求められます。）辺を貼られたら、i は条件を満たさなくなるので、set からは当然削除されます。
- 以下の図は、辺を貼る例です。



木の圧縮が終わったら

- 木の圧縮が終わったら、後はダイクストラ法によって、各圧縮済み木に含まれる頂点に対して「最寄りの種類 1 を持つ頂点 = 黒い頂点」までの最短距離を求めます。
- 最短距離が求まれば、後は5頁前のスライドで述べた矢印の向きを決めるだけです。ここでは、頂点 i からの「最寄りの黒い頂点」までの距離を $dista_i$ とします。
  - 各圧縮済み木の辺 (u, v) [u の方が根の頂点 1 に近い] に対し、矢印の向きが以下のように決まる。ただし、$cost$ を辺の長さとする。
  - $dista_u = dista_v + cost$ → 矢印の向きは $v \rightarrow u$ … ①
  - $dista_v = dista_u + cost$ → 矢印の向きは $u \rightarrow v$ … ②
  - どちらでもない場合 → 頂点 v の $\frac{dista_u - dista_v + cost}{2}$ 上に矢印の境界がある。… ③

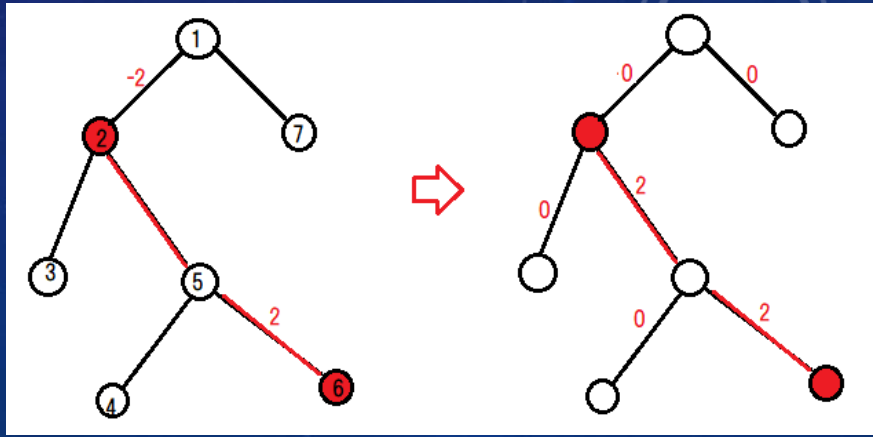
木の圧縮が終わったら

- まず前提として、圧縮済み木に含まれない辺における**根付き木上で長さ 1 下に行く事によって増える最寄り店までの最短距離は 1** です。これが通常の状態なので、不便さの長さ 1 あたりの差分が通常の状態からどの程度差がついているかというのを記録する方が楽になります。
- ① の場合、矢印が葉から根に向かっています。そのため、長さ 1 下に行く事によって増える最寄り店までの最短距離は 1 です。**通常の状態からは差がついていません。**
- ② の場合、矢印が根から葉に向かっています。そのため、長さ 1 下に行く事によって増える最寄り店までの最短距離は -1 です。$u-v$ パスに含まれる辺に対しての長さ 1 あたりの差分は、**通常の状態から 2 差がついていますので、$u-v$ パス上にある辺の不便さの差分に加算をする必要があります。**
- ③ の場合、$u-v$ パス内のどの辺上に矢印の向きの境目があるかを求める必要があります。これはダブリングによって $O(\log N)$ で求められます。あとは ③ の $u-v$ パスを ① と ②に分解するだけです。（ただし辺の途中に境界がある場合、該当する辺の差分に関してだけは特別に数える必要がある）

木の圧縮が終わったら

- ②において、$u-v$ パス上にある辺全てに対して「差分の値」を加算するためには、累積和を用いることが効果的です。（クエリはオフラインなので）
- 累積和と言っても、木の下部から累積的に和を更新していくものです。
- たとえば、下の図の例で $2-6$ パス上にある辺全てに対して 2 を加える場合、P_6-6 を結ぶ辺に 2 を加算し、P_2-2 間を結ぶ辺に -2 を加え、木の下の部分（葉）から根に向かって累積和を取ると、差分の値がパス上に加算されているのが分かります。（ただし P_x を x の親とします）
- この累積和は、2 つ以上のパスに対して加算を行っても、問題はありません。

- という訳で、全ての辺 (u,v) に対して、$\frac{ans_u-ans_v}{cost}$ が求まりました。
  - $cost$ は辺の長さです



注意

- 通常の状態（全ての辺について、不便さの長さ 1 あたりの差分は M であり、根に近い上側の方が不便さは小さい） を忘れないこと
- $ans_v - ans_u$ を求めるためには、長さ 1 あたりの差分を $cost$ 倍しなければならないことに注意すること
- ③ で特別扱いする辺 (u, v) に対しての不便さの差分の加算を忘れないこと

各辺に対する差分が求め終わったら

- あとは頂点 1 に対する不便さを求めるだけです。
- 頂点 1 から各頂点までの最短距離は、DFS またはダイクストラ法によって求めることができます。最短距離が求まれば、あとは問題文の式に従って不便さを求めるだけです。

- 頂点 1 の不便さが求まったら、小課題 2 と同じように、木の根から葉に向かって累積和を取りましょう。そうすると、全ての頂点に対する不便さが求まります。

- サンプルコード: https://beta.atcoder.jp/contests/tkppc3/submissions/2831978
  - 実行時間 1131 ミリ秒、コード長 5891 バイト

J 問題: 円の重なり

技術室奥プログラミングコンテスト #3 解説

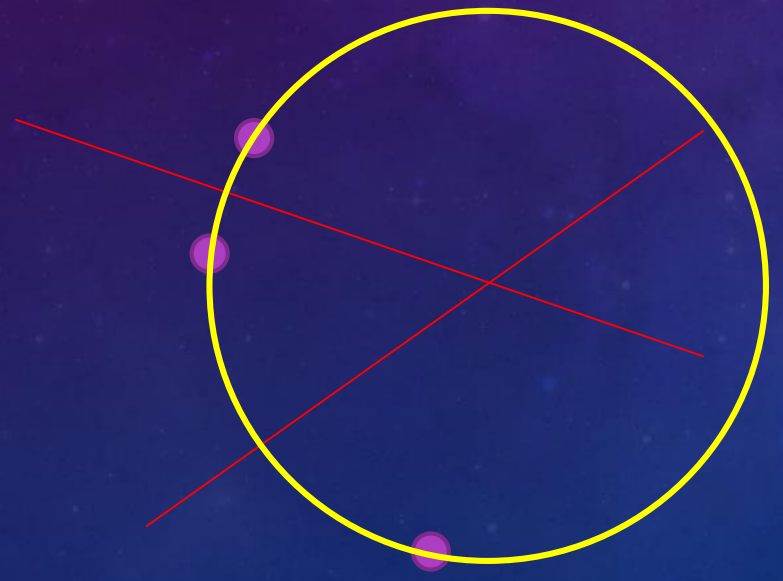
WRITER: E869120

問題概要

- 平面上に N 個の円があります。
- あなたは円の個数しか知らず、情報を全く知りません。あなたは以下のような質問を投げることによって、円の情報を得ようと思っています。
  - 点 (p, q) は何個の円に含まれるか。（ただし、p, q は小数でもよい）
- N 個の円全ての中心座標・半径を出力してください。
- N ≤ 20, 円の中心座標の絶対値 ≤ 1000, 200≤円の半径≤1000, 円の中心座標と半径は整数
- 入力は全てランダムである

重要な性質

- 3 点を通る円はただ 1 つしかない
- つまり、同一円周上にある 3 点を見出せば、円が分かる！



という訳で、円周上にある 3 点を見つけ出すという方針で得点を取りに行く

小課題 1（200 点）

- 円周を探し出せと言っても、前提として、円に含まれる点を見つけなければならない
- これは、**円の半径が 200 以上**であるという性質を用いる

- 円の半径が 200 以上なので、$(x, y) = (250p, 250q)$ を満たす少なくとも1個の点は円に含まれていることが分かる … ①

- 方針として、(x, y) は円に含まれており、$(x, y - 250)$ は円に含まれていない点 (x, y) を探せばよい → 線分 $(x, y - 250) - (x, y)$ は円周を通るから

- ① より、(x, y) を $(-2000, -2000)$ から$(1000, 1000)$ の範囲で 250 毎に探索すると必ず見つかる
  - **(-2000,-2000) → (-2000,-1750) → (-2000,-1500) → … → (-2000, 1000) → (-1750, -2000) → … という順番で**

小課題 1 (200 点)

- 線分 $(x, y-250)-(x, y)$ は円周を通るので、(x, r) が円に含まれるか、$y-250 \leq r \leq y$ の範囲で二分探索をすれば円周上の一点が求まる
- 線分 $(x, y-250)-(x, y)$ が円周を通る場合、線分 $(x-0.01, y-250)-(x-0.01, y)$ や線分 $(x+0.01, y-250)-(x+0.01, y-250)$ も円周を通る可能性は極めて高い（入力はランダムなので言えること）
- よって、前述の 2 つの線分でも二分探索をし、円周上の点を求めることができる。これで**円周上の 3 点が求まった。**

小課題 1 (200 点)

- 円周上の 3 点を A, B, C として、中心を D = (p, q) とすると、DA=DB なので、円の方程式より p, q の一次式 … ① ができる。また、DA=DC なので、これに関してもまた別の p, q の一次式 … ② ができる。

- (p, q) は ①･②の連立方程式の解である。連立方程式を解けば 3 点を通る円の中心座標が求まる。（小数誤差に注意。）

- 例えば, $A = (0, -3), B = (3, 6), C = (-1, -2)$ のとき、
  - $DA^2 = p^2 + (q+3)^2, DB^2 = (p-3)^2 + (q-6)^2, DC^2 = (p+1)^2 + (q+2)^2$
  - ① = $DA^2 - DB^2 = p + 3q - 6 = 0$, ② = $DA^2 - DC^2 = p - q - 2 = 0$ となる。
  - ①･② の連立方程式より、$(p, q) = (3, 1)$ が求まる。

- 詳しくは こちら の 2 つ目の解法を参照。

200 点

550 点解法

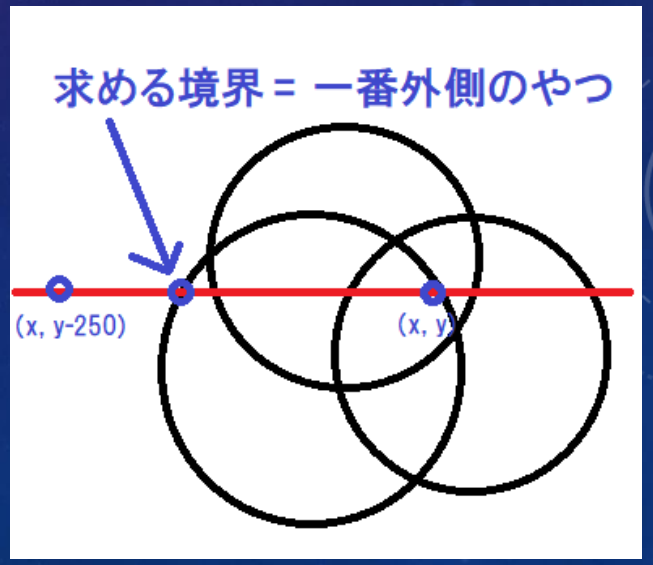
- 小課題 1 では円は 1 個しかなかったが、今度は円が 20 個もあるので、従来の方法だと何個の円が重なってしまう場合に対処できない。

- 対策として、**既に見つかった円はないものとして考える**
  - つまり、既に見つかった円に自分が質問した座標 (p, q) が含まれている場合、返答で入力された整数（何個の円に含まれているか）から既に見つかった円に含まれている数だけ**減らす**ということ。

- そうすると、1 つでも円を見つければ円の個数を 1 個減らせるので、1 つでもいいから円を見つけることを考える

550 点解法

- これは簡単
- 小課題 1 と同じように、点 (x, y) が**まだ見つかっていない** 1 個以上の円に含まれ、かつ点 $(x, y-250)$ が**まだ見つかっていない円**には全く含まれない、(x, y) を見つければよい（x, y は 250 の倍数）
- そうすると、$(x, y-250)-(x, y)$ 間には、ある円の円周、特に「全く円に含まれない」と「1 個の円に含まれる」の境界が存在するので、これは二分探索すればよい



- そうすれば円周上の一点が求まる
- また、小課題 1 と同じように $(x-0.01, p), (x+0.01, p)$ で二分探索
  - それをやっても、求める境界は一番外側のやつなので 3 点は同一円周上に

550 点解法

- 1 個の円を探すために何回質問をしなければならないか
  - まず、条件を満たす線分を探すために $13 * 13 = 169$ 回
    - 左下 (-2000,-2000) 右上 (1000,1000) の範囲を 250 ごとに探すため
  - 次に、二分探索 3 回に $3 * 45 = 135$ 回
    - ある程度精度は良い必要がある
- よって、全体での質問回数は $20 * (169 + 135) = 6080$ 回 → 200 + 350 = 550 点

550 点

650 点解法

- 実際、前の円を見つける時に座標 (x, y) まで探したら $(x, y-250)$ 以前を探索する必要はなく、ここ（探索済みの座標）にまだ見つかっていない円がある訳がない
  - 理由は簡単。$(x, y-250)$ 以前に関しては、(p, q) を含むまだ見つかっていない円があったら、これがなくなるまで「二分探索をして新たな円を見つけること」を繰り返しているため

- そのため、もし「まだ見つかっていない円に含まれる最初の座標」を探すときに、前の円を見つけた場所 (x, y) から探索を開始した場合、二分探索パート以外の質問回数は $169 + 20 = 189$ 回
- 二分探索パートが $20 * 3 * 45 = 2700$ 回なので、合計すると 2889 回 → 650 点

650 点

940 点解法

- さらに回数を少なくすることはできないのでしょうか？
  - 円に含まれる最初の場所 (x, y) を探索 → 合計 189 回
  - **二分探索 → 合計 2700 回 → こちらを短くするしかない！**

- 果たして本当に同一円周上の 3 点を取る必要はあるのか？　1 点だけでいいのではないか？
  - 全ての円の情報 (x, y, r) は整数の組である

- 整数なので、ある程度は絞られそう

940 点解法

- **実は, 無理数座標 (p, q) を通る, 整数座標を中心とする半径が整数の円は, 2 つ存在することはほぼあり得ない**
  - ただし、$p = 83.1218994137500812$ のような √ でもない適当な数の場合
- しかし、従来の方針だと、調べる x 座標は $-2000, -1750, -1500, -1250, \ldots, 500, 750, 1000$
  - それだと整数になってしまう
- 調べる x 座標を $-2000 + P, -1750 + P, -1500 + P, -1250 + P, \ldots$ (P は無理数) にしよう！

940 点解法

- じゃあ無理数座標 (p, q) を通る円の情報（中心座標・半径）はどのようにして得ればいいのか？
  - 制限時間が 5 秒もあるので、中心座標で全探索して、半径が整数となるものを求めるという手法も使える（sqrt は重いので、長さの 2 乗で計算した方がよいが）
- 以上の方法を使うと、今まで 3 個の同一円周上の点を取っていたのが 1 個しか必要なくなる
  - 二分探索の回数は45 回と見積もる
  - そのとき、$189 + 20 * 45 = 1089$ 回の質問 → 940 点くらい

940 点

1070 点解法

- 実は二分探索 50 回もいるか？
  - そのまま座標で二分探索したら精度の都合上 50 回しか必要ない
- しかし、実際は円は $2000 * 2000 * 800 = 3.2 * 10^9$ 通りしか存在しないので、二分探索の回数は理論上 $\log(3.2 * 10^9) = 32$ 回くらいしか必要がなさそう
- ただ通り数が多いので、全部を調べると計算回数の方で間に合わなさそう。
  - だから、20 回くらい二分探索した後にある程度半径が整数に近くなる中心座標を求め、（理論上 $3.2 * 10$^9 ÷ $2^{20} =$ 3000 個くらい） その後 3000 個に対して二分探索を行うとちょうど良い。
- $20 + \log(3000) \leq 32$ なので、二分探索の探索回数 32 回と見積もると…
  - $189 + 20 * 32 = 839$ 回 → 1070 点

1070 点

満点解法への道

- ここからは、「一気に 2 個の円について求める」などという特殊な手法を使わなければ厳しそう

- 理論上の最小値は、状態数が $C(2000 * 2000 * 800, 20) \simeq 2^{570}$ なので、仮に 1 回の質問につき状態数が平均して ½ に絞られるとすると 570 回の質問、1/3 に絞られるとすると 360 回の質問で全ての円の情報を知ることができそう

- 1 個以上の円に覆われている確率は $\frac{1}{2}$ よりは高いので、1 回の質問で状態数が 1/2 以下には絞られるとは予測される。だが、仮に 1/3 だとしても、円の領域は連続なので 360 回でいけるという上手い事にはならなさそう。

サンプルコード

- 小課題 1: 200 点解法
  - https://beta.atcoder.jp/contests/tkppc3/submissions/2829871
- 小課題 1 + 2: 650 点解法
  - https://beta.atcoder.jp/contests/tkppc3/submissions/2830152
- 小課題 1 + 2: 1070 点解法
  - https://beta.atcoder.jp/contests/tkppc3/submissions/2841090

おわりに

技術室奥プログラミングコンテスト #3 解説

おわりに

- 今回のコンテストは、どうでしたか？
- 皆さん参加していただきありがとうございました。楽しんでいただけたら幸いです。
- コンテストシステムを貸していただいた AtCoder 社様には感謝です。

- また、今回は筑駒中高パ研の主催になりました。皆さんも、中学・高校・大学などで一丸となってコンテストを開いてみてはいかがでしょうか？
- ちなみに、このコンテストはもしかしたら来年もやるかもしれません。ご期待ください。

結果発表

- 1 位: maroonrk (5200 点)
- 2 位: snuke (4300 点)
- 3 位: uwi (3955 点)
- 4 位: ats5515 (3900 点)
- 5 位: kirika_comp (3565 点)
- 6 位: WA_TLE (3345 点)
- 7 位: camypaper (3200 点)

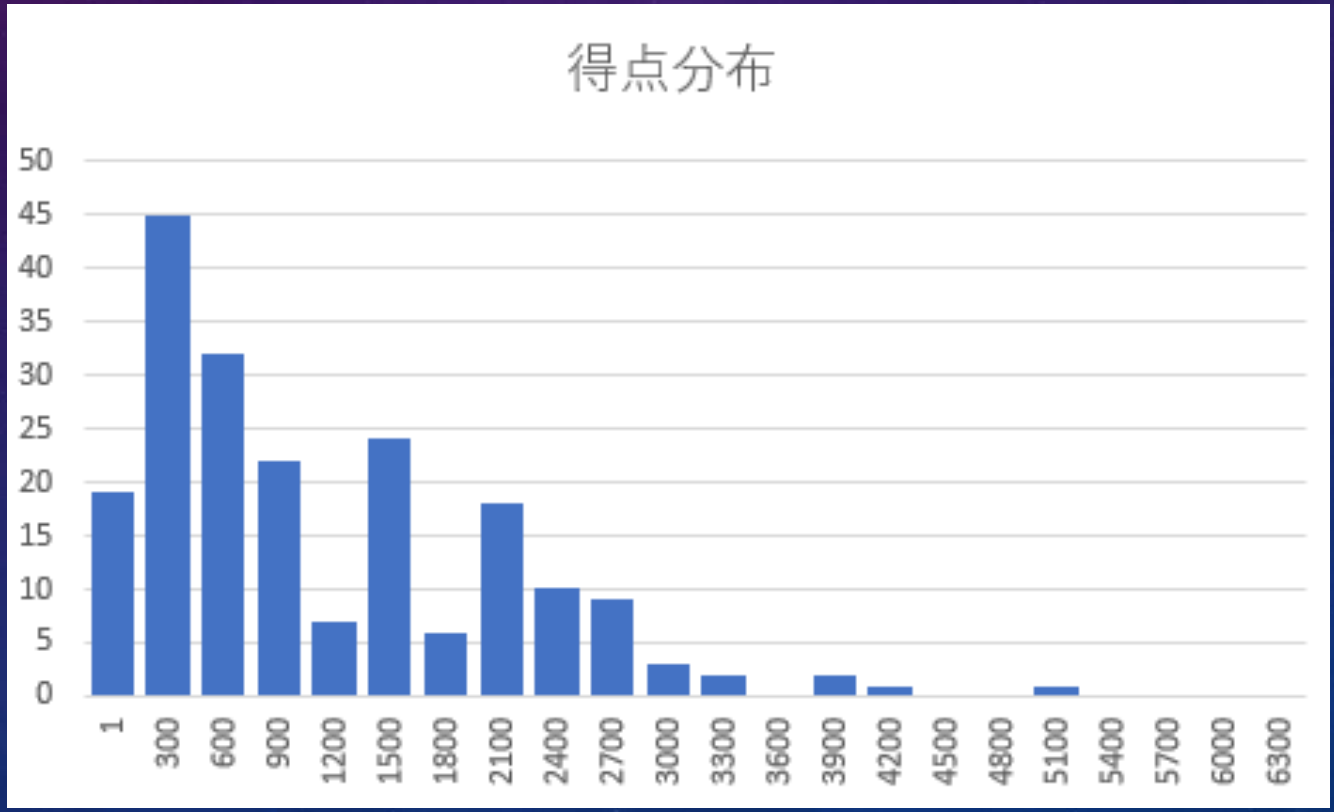
- 満点: 6300 点

カスタマイズ

順位	ユーザ	得点	A	B	C	D	E	F	G	H	I	J
1	maroonrk	5200 (1) 236:03	100 213:16	200 211:08	300 206:21	400 202:17	500 196:23	700 191:36	700 186:46	1000 139:28	1100 (1) 116:09	200 236:03
2	snuke	4300 190:33	100 190:33	200 185:12	300 183:18	400 179:50	500 5:52	700 10:31	(4)	1000 52:43	1100 162:50	-
3	uwi	3955 (2) 239:23	100 6:36	200 10:51	300 15:09	400 23:08	500 29:03	700 33:16	700 (1) 111:47	1000 142:12	55 (1) 239:23	-
4	ats5515	3900 (10) 239:22	100 3:26	200 7:40	300 14:11	400 38:30	500 46:56	700 55:26	700 (10) 239:22	1000 113:20	-	-
5	kirika_comp	3565 (12) 207:19	100 1:33	200 3:11	300 (4) 21:17	400 26:56	500 33:57	700 40:36	(16)	1000 (4) 149:21	165 (2) 184:11	200 (2) 207:19
6	WA_TLE	3345 (7) 156:42	100 1:55	200 (1) 11:29	300 9:32	400 21:42	500 25:24	700 30:44	-	780 (1) 62:22	165 (2) 123:53	200 (3) 156:42
7	camypaper	3200 (4) 95:07	100 66:45	200 (2) 74:06	300 73:26	400 83:18	500 95:07	700 87:04	-	1000 (2) 53:12	-	-
8	rickytheta	3165 (1) 234:38	100 4:21	200 6:32	300 9:48	400 61:57	500 69:20	700 137:27	200 181:33	510 167:17	55 196:14	200 (1) 234:38

得点分布

- 1 点以上得点をされた方の得点分布です。



おしまい

ありがとうございました！

2017. 7. 14
筑波大学附属駒場中学校・高等学校
中高パーソナルコンピューター研究部