解説・講評

UTPC2023 運営チーム

目次

オンサイトへようこそ

- UTPC2022 に引き続き、今年もオンサイトで開催できました

- **会場を貸してくださった　MC Digital 社の皆さん、**
  ジャッジシステムを貸してくださった　AtCoder の皆さん、
  ありがとうございます！

去年までとの変更点

- 運営チームの増員
  - 継続 7 人
  - 新規 9 人

- 問題数の増加
  - 13 (2020) → 14 (2021) → 15 (2022) → 17 (2023)

全体講評（想定難易度 vs 実際の解かれ具合）

想定難易度

- L, N, J, E, Q, B, A, C, D, H, O, G, M, P, K, F, I

実際の解かれ具合

- L, E, N, J, A, B, Q, H, D, C, K, G = O, M, F = P, I (17:55)

A - Again Make UTPC

原案: AngrySadEight
準備: AngrySadEight
tester: kumjin3141, zkou

問題概要

- 長さ N の文字列 S が与えられる
- 操作「区間を一つ選んでソートする」を好きな回数行える
- S に連続部分列 UTPC を作るのに操作は何回必要か？

制約

- $1 \leq N \leq 2 \times 10^5$

考察

- 「UTPC」は辞書順とは逆順になっている
- よって以下の性質が成り立つ
  - 最終的に「UTPC」ができるとき、これを構成する文字の相対的な位置は変化しない
- 「UTP」や「PC」などの**塊**を、どんどん大きくしていくことを考える
- 辞書順の性質より、以下も成り立つ
  - 1 回の操作で塊の大きさは最大でも1 しか増やせない

C C C U C C T P U U C U

↓

C C C C C U T P U U C U

↓

C C C C C U T P C U U U

塊の全探索＋場合分け

- 「どの塊から開始するか」を全探索
- 他の文字の相対的な位置に応じて場合分け
  a. [*]UTPC[*] → 0 回
  b. [*]UTP[*]C[*] → 1 回
  c. [*]U[*]TPC[*] → 1 回
  d. [*]UT[!C]P[*]C[*] → 2 回
  e. [*]U[*]TP[*]C[*] → 2 回
  f. [*]U[*]T[!U]PC[*] → 2 回
  g. [*]U[*]T[!C]P[*]C[*] → 3 回
  h. [*]U[*]T[!U]P[*]C[*] → 3 回
  i. [*]U[*]T[C&U]P[*]C[*] → 4 回
  j. [*]U[*]T[C&U]P[*]C[*] → 4 回

凡例

赤文字:開始する塊
[*] :任意の文字列
[!C] :C を含まない文字列
[!U] :U を含まない文字列
[C&U] :C と U を両方含む文字列

詳細は AtCoder 上の公式解説へ

統計情報

- AC / Trying teams
  - 36 / 96 (38%)

- First Acceptance
  - オンサイト参加: Sayaka_Manipulators (01:27:03)
  - 全体: Can Be UT (00:31:54)

目次　全体講評

B - Black or White 2

原案: Mitsubachi
準備: Mitsubachi
tester: AngrySadEight, Mitarushi

問題概要

- N × M のマス目のうち K マス黒で塗る
- 2 × 2 のマス目の部分で白黒 2 つずつのようなものを少なくしたい
- 最適例を構築してください

制約

- $1 \leq N, M \leq 1500$
- $0 \leq K \leq NM$

考察

- マス目を斜め線にする
- 連続する 2 本以上の斜め線に塗るのは良さそう
- 間を開けて塗るのも良さそう

- $N \leq M, 0 \leq K \leq NM/2$ として一般性を失わない
- 左下部分を斜め線部分にして右上部分を間を開けて塗って微調整

考察

- N, M が大きいほど微調整がたくさんできるので小さい方だけ考えれば良い
- 全探索を回すと (N, M, K) = (3, 3,4), (2, 2, 2) 以外は OK

- (N, M, K) = (3, 3,4) は 4 つの角に塗れば OK
- (N, M, K) = (2, 2, 2) はどうやっても同じ

統計情報

- AC / Trying teams
  - 32 / 49 (65%)

- First Acceptance
  - オンサイト参加: UPCT (01:24:29)
  - 全体: tomerun (00:43:16)

目次　全体講評

C - Contour Multiplication

原案: SSRS
準備: nok0
tester: sapphire15, Kite_kuma

問題概要

- 「popcount(j ⊕ C_i) = D_i なる各 j について、A_j ← A_j × X_i mod M」という操作を K 回行った後の結果を求めよ

制約

- $1 \leq N \leq 18$
- $1 \leq K \leq 5 \times 10^5$

解法

- 高速ゼータ変換のような DP に、異なる桁の個数を持たせる
- dp[i][j][k] = (下から i 桁が j と一致し、それより上の桁が j と k 箇所で異なるような箇所に掛ける値)
- 初期値: 最初 dp[0][j][k]=1 とし、各操作について dp[0][C_i][D_i] に X_i を掛ける
- 遷移: dp[i+1][j][k] *= dp[i][j][k], dp[i+1][j ⊕ 2^k][k-1] *= dp[i][j][k]
- dp[N][0][i] が答え
- 計算量は O(2^N N^2)

統計情報

- AC / Trying teams
  - 14 / 15 (93%)

- First Acceptance
  - オンサイト参加: Traffic Light (02:02:40)
  - 全体: yosupo (00:56:43)

目次　全体講評

D - DRD String

原案: confeito
準備: confeito
tester: chinerist, nok0

問題概要

- 空でない文字列 D,R を用いて S = D+R+D と表せるような文字列（**DRD文字列**）の個数を数え上げる
- 長さは N 、使える文字の種類数は M

制約

- $3 \leq N \leq 10^6$
- $1 \leq M \leq 10^6$

考察

- S の代わりに D,R を数え上げたい
- しかし S に対して D,R の取り方は一意ではない

  “abacaba” = “a” + “bacab” + “a” = “aba” + “c” + “aba”

- 逆にどのようなときに一意ではなくなるのか？

- 上の例を見ると、”aba” はDRD文字列
  → D に当たる部分がDRD文字列であるときには、もっと短くできる？

「 D の部分がDRD文字列か」に注目

- S = D + R + D と表せる場合、D が DRD 文字列なら、D としてもっと短いものを取れる
- 逆は？
  → D がDRD文字列でなくても、D = D’ + D’ と表せる場合は S = D’ + (D’ + R + D’) + D’ と表せる

- ということは…… R が空でも良いとした方が考えやすそう
- 以下、Rが空でも良い「広義DRD文字列」を考える

広義DRD文字列の性質

- 広義DRD文字列 S が D + R + D と表せる場合、D が広義DRD文字列なら、D としてもっと短いものを取れる
- 逆に、D が広義DRD文字列でないなら、それよりも短い D の取り方は存在しない（証明できる）

- つまり、広義DRD文字列を数え上げるには、「D が広義DRD文字列ではない」として数え上げればOK
  → これは累積和を取りながら DP すれば、O(N)

元の問題を解く

- 「広義DRD文字列だが、元の意味でのDRD文字列でないもの」を除外
- R が空であるような表し方しかないもの
  → S = D + D と表せて、かつ D が広義DRD文字列ではないもの

  （D が広義DRD文字列なら S は元の意味でのDRD文字列であることは明らか。逆は解説記事へ）

- 広義DRD文字列でないものの通り数は前述のDPで求めているので、容易に除外できる
- この処理まで正しく行えればAC

統計情報

- AC / Trying teams
  - 18 / 19 (95%)

- First Acceptance
  - オンサイト参加: UPCT (01:39:06)
  - 全体: HoMaMaOvO (01:31:18)

目次　全体講評

E - Equally Dividing

原案: Mitsubachi
準備: Mitsubachi
tester: miscalc, kumjin3141

問題概要

- N × M のマス目に 1 から NM を 1 つずつ割り当てる
- 各行に割り当てた M 個の整数をどの行も等しくできますか

制約

- N × M ≤ 3 × 10^5

考察

- M の偶奇で分ける
- M が偶数のときは、左半分の縦列を 1 ずつ減らして、右半分の縦列を 1 ずつ増やしてけば OK

- M が奇数を考えよう
- N が偶数のときは、総和が N の倍数でないので無理
- M = 1 は、N = 1 のときだけ OK (コーナーなので要注意)

考察

- M = 3 を作れるか判定したい
- mod 3 で 0, 1, 2 のを 1 つずつ割り当てると商の総和を等しくしたい
- これは、できる (全探索で実験すると良い)

- M = 3 と偶数でできるので 5 以上の奇数でも OK

- (N, M) = (1, 1) はサンプル以外の全テストケースに入れてます！！

統計情報

- AC / Trying teams
  - 71 / 95 (75%)

- First Acceptance
  - オンサイト参加: Sayaka_Manipulators (00:38:01)
  - 全体: hitoare (00:19:46)

目次　全体講評

F - Flip or Not

原案: Mitarushi
準備: Mitarushi
tester: torisasami, chinerist

問題概要

- N 枚のカードがある
- 以下の操作を最大 10^6 回繰り返す
  - 一番右のカードを一番左に移動する
  - その後、移動したカードが表なら左から A_1, A_2, …, A_P 枚目の表裏を入れ替える
  - 最後に、左から B_1, B_2, …, B_Q 枚目の表裏を全て入れ替えるか何もしないか選択する
- N 枚のカードの表裏の初期状態と終了状態が与えるので、条件を満たすよう操作列を構築

制約

- $1 \leq N \leq 5000$

考察

- 操作を多項式 f ∈ GF(2)[x] の演算として解釈できる
- $[x^n]f = 1$ ⇔ 左から n+1 枚目のカードが表 とする

- 一番右のカードを一番左に移動し、移動したカードが表なら左から A_1, A_2, …, A_P 枚目の表裏を入れ替える
  - f ← xf mod f_a (f_a := 1 + x^n + Σ_i x^{A_i-1})
- 左から B_1, B_2, …, B_Q 枚目の表裏を全て入れ替える
  - f ← f + f_b (f_b := Σ_i x^{B_i-1})

問題の言い換え

- 操作列に対応する f_u を考える
- [x^n]f_u = 1 ⇔ 最後から n+1 回目の操作で B_1, …, B_Q 枚目を入れ替えた
- f_t ≡ x^M*f_s + f_b*f_u (mod f_a) が成り立つ
  - ただし M: 操作回数、f_s, f_t: S と T に対応する多項式

- よって以下の (M, f_u) を見つける問題に帰着
- f_t + x^M*f_s ≡ f_b*f_u (mod f_a) かつ deg(f_u)<M

操作列の存在判定

- 操作回数を決めたとき f_u が存在するか判定する方法を考える

- g ∤ f_t + x^M*f_s ならば f_u は存在しない (g := gcd(f_a, f_b))
- g | f_t + x^M*f_s ならば f_t + x^M*f_s ≡ f_b*f_u なる f_u が存在
- x^M*f_s (mod g) から x^{M+1}*f_s (mod g) は O(N/w) で計算できる
- よって g | f_t + x^M*f_s の条件は全体で O(NM/w) で判定可能

操作列の存在判定

- 操作回数を決めたとき f_u を求める方法を考える
- g | f_t + x^M*f_s ならば次数が最小の f_u は次のように求められる

- p*f_b ≡ g (mod f_a) なる p を拡張 Euclid の互除法 (or binary gcd) で求める
- (f_t + x^M*f_s)/g*p*f_b ≡ f_t + x^M*f_s なので、f_u' = (f_t + x^M*f_s)*p/g は f_t + x^M*f_s ≡ f_b*f_u の解の一つ
- f_u = f_u' mod (f_a/g) が次数最小なので、deg((f_t + x^M*f_s)*p/g mod (f_a/g)) < M か判定したい
- deg((f_t + x^M*f_s)*p mod f_a) < M+deg(g) と変形すればこれも O(NM/w)

まとめ

- 以下のようにこの問題は解ける
- 計算量は O(N(N+M)/w) とか

- g = gcd(f_a, f_b) と p*f_b ≡ g (mod f_a) なる p を計算する
- M = 1,...,10^6 に対して逐次以下の条件を判定する
- 0 ≡ f_t+x^M*f_s (mod g) かつ deg((f_t + x^M*f_s)*p mod f_a) < M+deg(g)
- もし条件を満たせば (f_t + x^M*f_s)*p mod f_a) / g が求める操作列

統計情報

- AC / Trying teams
  - 1 / 2 (50%)

- First Acceptance
  - オンサイト参加: ? (??:??:??)
  - 全体: HoMaMaOvO (04:46:48)

目次　全体講評

G - Graph Weighting

原案: tokusakurai
準備: tokusakurai
tester: Kite_kuma, sapphire15

問題概要

- **連結無向グラフ** が与えられる
- 各辺に 0 以上 L 以下の**整数重み**を割り当てる
- W = 0,…,K について、任意の全域木の重みを W にできるか判定し、可能なら重みの 2 乗和の最小値を求めよ

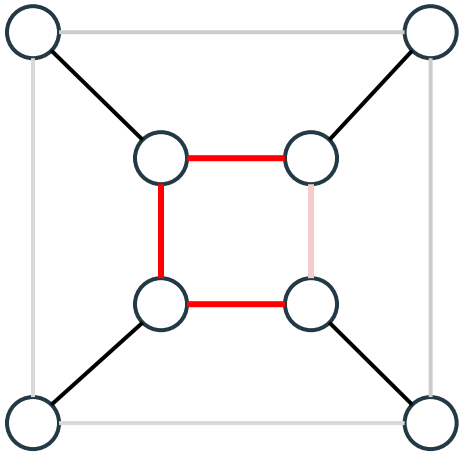
制約

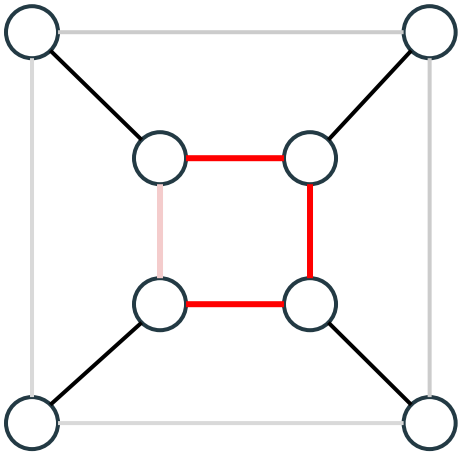
- $2 \leq N \leq 10^5$
- $1 \leq M \leq 2 \times 10^5$
- $1 \leq L \leq K \leq 10^5$

考察

任意の全域木 の重みが同じになるには？

必要条件 : サイクルに含まれる辺の重みは**全て等しい**





2-連結成分分解

任意の全域木 の重みが同じになるには？

必要十分条件 : 2-連結成分に含まれる辺の重みは**全て等しい**

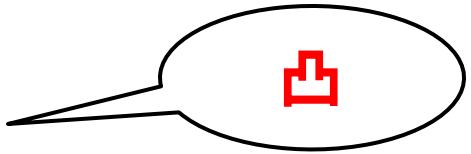
- **2-連結成分** G_i = (V_i, E_i)
- a_i := |V_i| - 1, b_i := |E_i|
- 任意の全域木 T について、|E(T) ∩ E_i| = a_i

※ **2-連結成分** : どの 1 頂点を除いても連結な極大部分グラフ

問題の言い換え

W = 0,…,K について、以下の最適化問題を解けばよい：

$$\begin{aligned} \text{min.} \quad & \sum b\_i \, x\_i^2 \\ \text{s.t.} \quad & \sum a\_i \, x\_i = W \\ & x\_i \in \{0,\ldots,L\} \end{aligned}$$

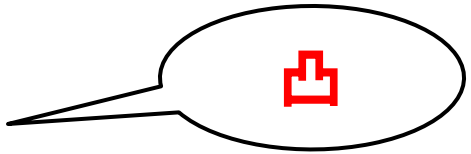


x_i : i 番目の 2-連結成分の辺重み（全て同じ）

a_i ごとに分類

a_i ごとに i を分類すると、**貪欲法**で最適化問題を解ける

$$\text{min.} \quad \sum b_i x_i^2$$
$$\text{s.t.} \quad \sum a_i x_i = W$$
$$x_i \in \{0,\ldots,L\}$$



- **差分**を優先度付きキューで管理（凸性により差分が単調増加）
- 計算量 O(N log N + K log K log N)

min-plus convolution

- **f(W)** := 最適値
- **f_m(W)** := a_i = m となる i のみを考えたときの最適値

$$f = f_1 \oplus f_2 \oplus \cdots \oplus f_K$$

アルゴリズム

1. g = f_1 と初期化
2. for m = 2,...,K do **g = g ⊕ f_m**

凸性を用いた高速化

g ⊕ f_m を **O(K) 時間**で計算する方法:

- f_m は m の倍数以外では +∞
- m の倍数のところだけ取ってくると凸 (**h_m** とする)
- g を添字 mod m で分類して m 個の数列 g_0, g_1,....,g_{m-1} に分解
- 各 g_i と h_m の **min-plus convolution**

h_m の**凸性**により、最後の畳み込みが線形時間で可能 (**SMAWK algorithm**)

まとめ

- ∑ a_i = N - 1 より、a_i の種類は **O(√N)** 個
- min-plus convolution の計算量は **O(K√N)**
- 全体の計算量 : O(M + N log N + K log K log N + K√N)
- monotone minima を使うと最終項が K√N log K になるが、どちらでも間に合う

統計情報

- AC / Trying teams
  - 6 / 7 (83%)

- First Acceptance
  - オンサイト参加: UPCT (03:55:13)
  - 全体: Pianist (01:18:51)

目次　全体講評

H - Huge Segment Tree

原案: SSRS
準備: SSRS
tester: zkou, sapphire15

問題概要

- 長さ 2^K のセグメント木を作った。すべての区間にクエリを投げたとき、値を取ってくる区間の個数の分布を求めよ

制約

- $2 \leq K \leq 5 \times 10^5$

考察

- 全体を、左半分 2^(K-1)・右半分 2^(K-1)に分ける
- 片方に完全に含まれる場合: K-1 の場合と同じ
- そうでないときは、中央をまたいでいるので、(全体の区間数)=(左側の区間数)+(右側の区間数) と分けられる
- l=0 と固定されたときの分布が必要になる

考察

- l=0 という条件が加わったときの分布を考える
- このとき、BIT と全く同じ構造になり、必要な区間数は popcount(r) になる
- l=0 のときの区間数の分布の母関数を f_K(x) とすると、f_K(x)=(1+x)^K+x-1
- これが左端・右端ともに考えられる
- l=0 と限らないときの区間数の分布の母関数を g_K(x) とすると、
  g_K(x)=2g_{K-1}(x)+f_{K-1}(x)^2-x^2+x
  - 2g_{K-1}(x): 中央を通らない場合、f_{K-1}(x)^2: 中央を通る場合、-x^2+x: 全体の場合の補正

考察

- 代入すると g_K(x)=2g_{K-1}(x)+((1+x)^(K-1)+x-1)^2-x^2+x
- 整理すると g_K(x)=2g_{K-1}(x)+((1+x)^2)^(K-1)+2(x-1)(1+x)^K-x+1
- 初期値は g_0(x)=x
- よって g_K(x)=Σ[i=0,K-1]2^(K-1-i)(1+x)^(2i)+2(x-1)Σ[i=0,K-1]2^(K-1-i)(1+x)^i+x+2^N-1
- Σ[i=0,K-1]2^(K-1-i)(1+x)^(2i) は、等比数列の和の公式を使って整理すると (2^K-(1+x)^(2K))/(1-2x-x^2) となる → 二項係数+疎な除算で O(K)
- Σ[i=0,K-1]2^(K-1-i)(1+x)^i は同様に (2^K-(1+x)^K)/(1-x) → O(K)
- 全体で O(K) で解けた

統計情報

- AC / Trying teams
  - 22 / 25 (88%)

- First Acceptance
  - オンサイト参加: UPCT (00:27:27)
  - 全体: UPCT (00:27:27)

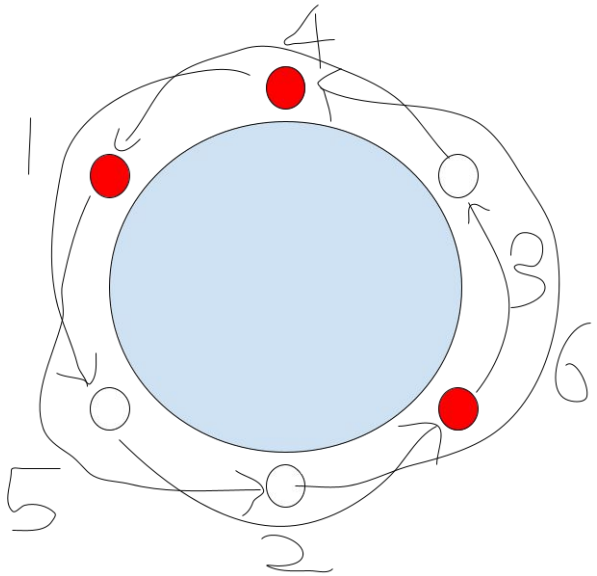
目次　全体講評

I - I Love Marathon Contest

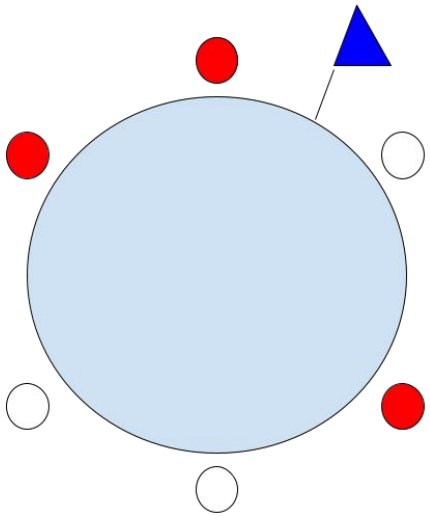
原案: noimi
準備: noimi
tester: tokusakurai, torisasami

問題概要

- 円周上に赤い人が N 人、白い人が N 人が位置 1, …, 2N にいる。位置 1 の人から始めて、別の色の人に到達するまで時計周りに回り、離脱する。次の人も同様のルールで周ることを 2N 人がいなくなるまで続行し、最後に位置 1 に戻る。
- すべての並べ方に対して、周回数の合計数を求めてください。



解説



周回数を数える代わりに、位置 2N と 1 の間に旗を立て、旗を横切った回数を求める。

同じ色の人同士の区別をなくし、位置 1 を赤とする。

旗を横切った回数を、赤が横切った回数と白が横切った回数で別々で数える。

解説

赤が横切る回数は簡単に求まる。

赤に -1、白に +1 を割り当て、累積和を取る。累積和の最大値が求める値。

赤⇒白にロープが渡されるのをイメージすると、上で求めた値は不足しているロープの本数

= スタート地点より前から持ってくる必要のあるロープの本数

= 赤が旗を横切る回数

最大値ごとに、カタラン数の拡張で求まる。

解説

白が横切る回数もほぼ同様だが、位置 1 の特殊性のためにズレる(+1 される)時がある。

位置 1 の赤と最後に走ることになる白を取り除き、同様の累積和を求める。

その値 +1 が白が横切る回数になる。

解説

累積和を赤(+1)、白(-1) で取り直す。

動きを観察すると、最後に走ることになる白は、

累積和最小の位置で区切っていくつかの区間に分けたとき、累積和最大を含む区間のうち最右端の区間の -1 に対応する白。

実は、最右端を取り除いたとしても周回数は正しく求まることが示せる。

取り除いた後の列を考えると、各列には元の列としての候補が
ほとんどの場合は 2 通り、一部のケースのみ 1 通り存在する。

解説

1 通りしか存在しないのは、累積和の最小値が 0 のとき。

よって、(累積和 >= 0) の条件を加えて、累積和の最大値を足し合わせたい。

これは、鏡像法と包除原理によって解くことができる。(実はこのパートは OEIS にもあります。)

統計情報

- AC / Trying teams
  - 0 / 0

- First Acceptance
  - オンサイト参加: ? (??:??:??)
  - 全体: ? (??:??:??)

目次　全体講評

J - Japanese Gift Money

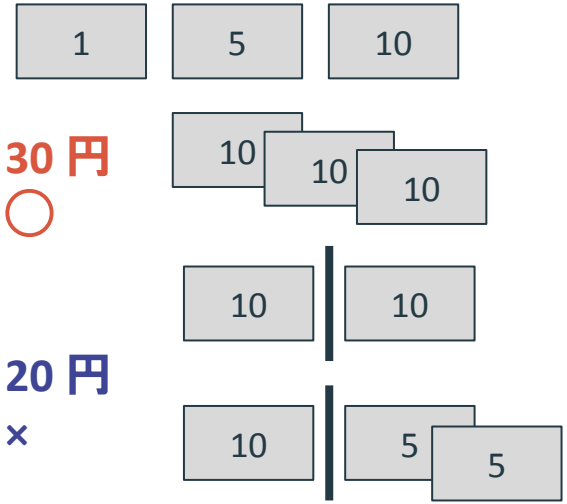
原案: miscalc
準備: miscalc
tester: confeito, AngrySadEight

問題概要

- N 種類の紙幣 A_1, …, A_N 円札がそれぞれたくさんある
  - **A_{i+1} は A_i の倍数**
- ここから選んで袋に入れる
- 「x 円の良い入れ方」とは次を満たす入れ方
  - 合計金額が x 円
  - 入れた紙幣を x/2 円と x/2 円に分割できない
- x 円が「良い金額」⇔「x 円の良い入れ方」が存在
- L 円以上 R 円以下の「良い金額」を数えよ

- N <= 60
- L, R, A_i <= 10^18

例



前提

- 「倍数」の制約が超重要
- 「各紙幣を何枚使うか」を並べると、位取り記数法のようになる
  （枚数制限のもとで非負整数を一意に表せる）
  - これは上から貪欲に入れた結果と一致するので、「貪欲な入れ方」と呼ぶことにします

	1	5	10	100
	0-4 枚	0-1 枚	0-9 枚	0- 枚
248	3	1	4	2

[1] 条件

- 「良い金額」の必要十分条件を考えたい

[1] 条件

- 「良い金額」の必要十分条件を考えたい

- 実は

x 円が良い金額　⇔　x 円の貪欲な入れ方で奇数枚の紙幣が存在

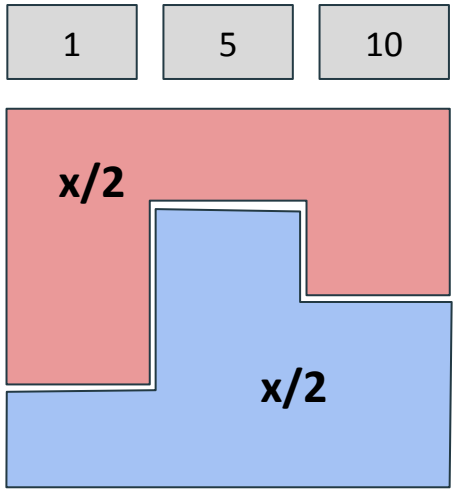
[1] 条件

- 最も本質的なところだけ説明します
- 本質：**貪欲な入れ方が良い入れ方にならないのは全部偶数枚のときだけ**

[1] 条件

- 最も本質的なところだけ説明します
- 本質：**貪欲な入れ方が良い入れ方にならないのは全部偶数枚のときだけ**

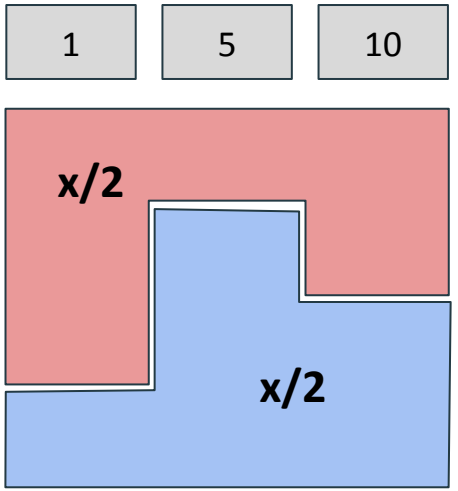
- 貪欲が 2 分割できるとする



[1] 条件

- 最も本質的なところだけ説明します
- 本質：**貪欲な入れ方が良い入れ方にならないのは全部偶数枚のときだけ**

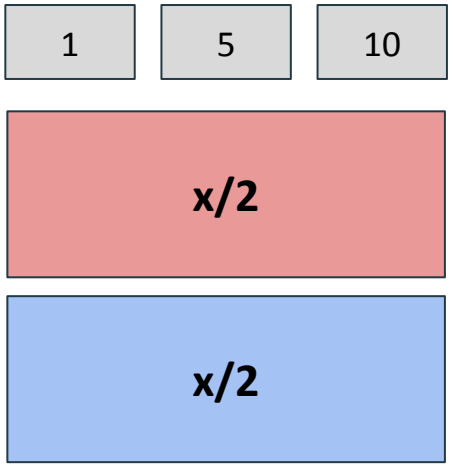
- 貪欲が 2 分割できるとする
- 分割した先もまた貪欲



[1] 条件

- 最も本質的なところだけ説明します
- 本質：**貪欲な入れ方が良い入れ方にならないのは全部偶数枚のときだけ**

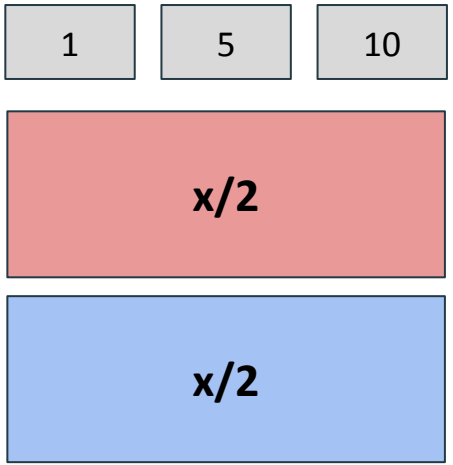
- 貪欲が 2 分割できるとする
- 分割した先もまた貪欲
- ところで、貪欲は一意 → 両方同じ



[1] 条件

- 最も本質的なところだけ説明します
- 本質：**貪欲な入れ方が良い入れ方にならないのは全部偶数枚のときだけ**

- 貪欲が 2 分割できるとする
- 分割した先もまた貪欲
- ところで、貪欲は一意 → 両方同じ
- → もとの入れ方は全部偶数枚



[2] 数え上げ

- 「M 円以下で、貪欲に入れたときに全部偶数枚」を数えればよくなった
- 桁っぽい考え方でできる（桁 DP とか）
- 「初めて下回る桁」を固定すると、簡単な式で求まる（これが一番楽？）

統計情報

- AC / Trying teams
  - 56 / 57 (98%)

- First Acceptance
  - オンサイト参加: UT a.k.a. Is (00:30:07)
  - 全体: ikejirou (00:25:56)

目次　全体講評

K - Kth Sum

原案: SSRS
準備: SSRS
tester: noimi

問題概要

- 長さ N の数列 A,B,C が与えられる
- A_i+B_j+C_k N^3 個のうち、K 番目に小さい値は？

制約

- 1 ≤ N ≤ 50000
- 1 ≤ K ≤ 10^9

考察

- 以下、A_i,B_j,C_k の値の最大値を M とおく
- 答えを決め打ち、二分探索する
- 「A_i+B_j+C_k <= x となるような (i,j,k) は K 個以下か？」という問題を O(log M) 回解くことに帰着
- ここで、「何個あるか？」ではなく「K 個以下か？」さえわかればよいことに着目

考察

- A, B, C は昇順に並んでいるとする
- f(i) = (A_i+B_j+C_k <= x なる (j,k) の個数) とすると、Σ[i=1,N] f(i) <= K かどうかわかればよい
- f(i) は広義単調減少であることに注意
- 各 f(i) の値は B, C の単調性を利用して O(N) で求められる

解法

- まず f(√(K/N)) を求め、これが √(NK) より大きかったら、f(1)+f(2)+...f(√(K/N)) だけで K を超えてしまう → 判定問題の答えは NO
- √(NK) 以下であるとする。
- f(1), f(2) …, f(√(K/N)-1) はそれぞれ O(N) で求められる → O(√(NK))
- f(√(K/N)+1) 以降は必ず √(NK) 以下であるため、B_j+C_k の下位 √(NK) 個を前計算しておくと、二分探索で求まる
- 前計算は優先度付きキューを使い O(√(NK)logN) でできる
- 全体で O(√(NK) (logN+logM) + NlogNlogM)

統計情報

- AC / Trying teams
  - 11 / 16 (68%)

- First Acceptance
  - オンサイト参加: ? (??:??:??)
  - 全体: yosupo (02:47:34)

目次　全体講評

L - Largest Triangle

原案: confeito
準備: confeito
tester: AngrySadEight, zkou

問題概要

- N 本の棒から 3 本を選び、それらからなる三角形を作る
- 三角形は作れるか？
- 作れるなら、面積の最大は？

制約

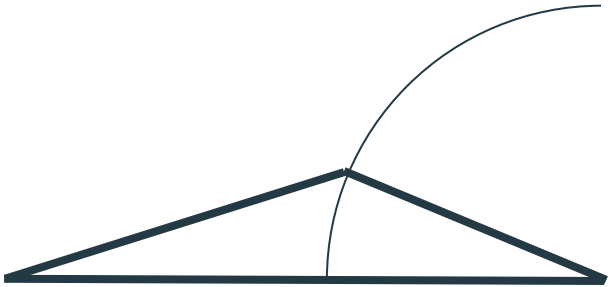
- $1 \leq N$ の総和 $\leq 2 \times 10^5$
- $2 \leq L\_i \leq 20000$　偶数（答えを整数かつlong longに収めるため）

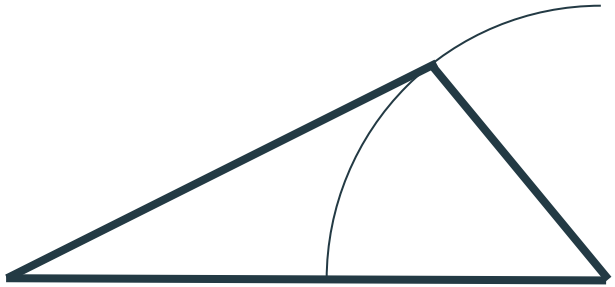
考察

- とりあえず何かを固定して考える

  → 最長の辺を固定する

  → 最長でない辺の片方を伸ばすと、高さは必ず高くなる





考察

- つまり、最長の辺を固定すれば、残りの2本の辺は

  「**最長の辺の長さを超えない範囲で長い　2本**」

  を選べばOK（これによって三角形にならないなら作成不可能）
- よって、棒を長さでソートして、連続する 3 本を全通り試せば良い

- 面積の 2 乗 → ヘロンの公式

統計情報

- AC / Trying teams
  - 113 / 128 (88%)

- First Acceptance
  - オンサイト参加: Rubikun (00:14:32)
  - 全体: パイナップル酢豚 (00:11:11)

目次　全体講評

M - Majority and Permutation

原案: chinerist
準備: chinerist
tester: tokusakurai, torisasami

問題概要

- （A=(1,3,...,2N-1)の場合）
- 順列Pのうち、元から正しい括弧列である+Pに従って並べ替えても正しい括弧列であるような文字列Sが存在するものの数mod998244353を求めよ

P=(6,3,8,7,1,4,5,2)の場合

考察

元の文字列のi文字目と並び替えた後のQ_i文字目が対応するとする

- A_i+A_j = 2N
- (Q_1,Q_2,...,Q_{A_i}) は (2N-A_i+1,2N-A_i+2,...,2N)

をみたすijがあると…？

並び替えた後、先頭A_j=2N-Ai文字の過半数が0+並び替えた後、末尾A_i文字の過半数が0

→ 0がN+1文字以上必要になってしまうのでダメ

解法

~~ゴミサンプルから察せられるように~~実は十分

すると

- A_i+A_j=2Nなるjが存在しないA_iはAから除く
- 2番目の条件に関して包除原理を適用→愚直にやるとO(N^2)のdp→分割統治 fftで高速化できてO(Nlog^2N)で解ける

十分性の証明

- 貪欲で並び替えた後のA_i文字目までの0の数を同時に最大化できる
- 条件どっちか満たさないと過半数にできることが示せる

統計情報

- AC / Trying teams
  - 2 / 2 (100 %)

- First Acceptance
  - オンサイト参加: ? (??:??:??)
  - 全体: HoMaMaOvO (04:05:30)

目次　全体講評

N - Number of Abbreviations

原案: SSRS
準備: Kite_kuma
tester: nok0, miscalc

問題概要

- 文字列 S が与えられる
- S から長さ 1 以上の部分文字列を取り去ってできる文字列の種類数は？
- 例
  - S = “aabc” から “ab” (2-3 文字目) を取り去ると “ac” になる
  - 他にも “a”, “”, “bc”, “abc” などができる

数える対象の考察

- 取り去り方 N (N+1) / 2 通りから，結果が同じものを引く
- 取り去る文字列の長さが異なれば結果も異なる -> 取り去る長さごとに分類
- i < j について，「[i, i + k), [j, j + k) を取り去った結果が一致する」⇔「S の [i, j) 区間と S の [i + k, j + k) 区間の文字列が一致する」が成立
- このとき，[i, i + k), [i + 1, i + k + 1), …, [j - 1, j + k -1), [j, j + k) を取り去った結果は全て一致する
- よって，区間 [i, j) のうち，S から [i-1, j-1) を取り除いた文字列と，S から [i, j) を取り除いた文字列が一致するものの個数を引けば良い

解法

- 取り去り方 N (N+1) / 2 通りから，結果が同じものを引く
- 区間 [i, j) のうち，S から [i-1, j-1) を取り除いた文字列と，S から [i, j) を取り除いた文字列が一致するものの個数を引く
  - これは，文字 c の登場回数を f_c と置くと，\sum_c f_c(f_c-1)/2 に等しくなっている

- 以上より，答えは N (N+1) / 2 - \sum_c f_c(f_c-1)/2

統計情報

- AC / Trying teams
  - 65 / 79 (82%)

- First Acceptance
  - オンサイト参加: World Abyss (00:17:37)
  - 全体: World Abyss (00:17:37)

目次　全体講評

O - Optimal Train Operation

原案: confeito
準備: confeito
tester: tokusakurai, chinerist

問題概要

- N+1 個の駅がある
- いくつかの駅に「車両基地」を作り、車両基地の間に列車を走らせる
- 各駅間に走らせたい列車の本数が決まっている
- 車両基地を作るコストと列車を走らせるコストの最小は？

制約

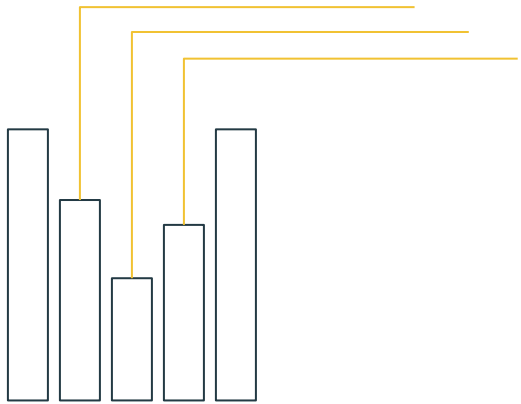
- $1 \leq N \leq 5 \times 10^5$
- $1 \leq C\_i, A\_i \leq 10^9$

$O(N^2)$解法

- 車両基地をまたぐ列車は走らせなくてOK
- 「その駅に車両基地を設置したときの最小コスト」は、直前の車両基地を決めれば求められる
- 各駅を求めるのに O(N) かかるので、$O(N^2)$

$$\mathrm{dp}(i) = \min\{\mathrm{dp}(j) + (i-j) \times \max\{C_j, \ldots, C_{i-1}\} \mid 0 \leq j < i\} + A_i$$

高速化



駅 i からの遷移のコストは、i から離れていくのに応じて
一次関数的に増加（最大値が変わらなければ）
→ 直線として管理すれば楽なのでは？

→ CHT を用いる

（ここを i についての一次関数として見る）

$$\mathrm{dp}(i) = \min\{\mathrm{dp}(j) + (i-j) \times \max\{C_j, \ldots, C_{i-1}\} \mid 0 \le j < i\} + A_i$$

高速化

- 途中で新たに大きい C_i が現れ、最大値が更新されると、一次関数の切片や傾きが変わる
  → そこまでで存在した一次関数を消して、新しく直線を追加
- 傾き……新たな最大値
- 切片……傾きが C_i である中で切片（下記橙）が最小になるものを選ぶ
  → -jx+dp(j) という直線を選んでCHTで最小を求める

$$\mathrm{dp}(i) = \min\{\mathrm{dp}(j) + (i-j) \times \max\{C_j, \ldots, C_{i-1}\} \mid 0 \leq j < i\} + A_i$$

まとめ

- 2段階のCHTが必要
- 遷移に使うCHT
  → 永続 Li Chao Tree
  → HLD木に載せる（Writer解、解説に補足あり）
- 切片の最小を求めるのに使うCHT
  → セグ木にCHTを載せる

- 分割統治で解く方法もあり、こちらの方が楽そう（Tester解）

統計情報

- AC / Trying teams
  - 7 / 12 (58%)

- First Acceptance
  - オンサイト参加: UPCT (01:09:31)
  - 全体: UPCT (01:09:31)

目次　全体講評

P - Priority Queue 3

原案: chinerist
準備: chinerist
tester: tokusakurai, torisasami

問題概要

- priority queueでpush,popをそれぞれN,M回固定の順序でやる
- pushするときの整数の順番はN!通り考えられる
- そのうちpopしたものの集合が指定したものになる順序の数 mod 998244353 を求めよ

考察

重要なのは

- 操作中 (Yに入ってないAの要素の最大値) > (Xに入っているAの要素でないものの最小値)が常に成り立つ
- （popの操作の際XにAの要素が1つ以上ある）

これさえ守られていれば操作後はかならずY=Aになり、popのとき最小値をpopする、という制限はあまり考えなくてよくなる

これらを状態量に持つdp？→popしたときの（Yの入ってないAの要素の最大値）がどう変化するかわからなくない？？？

解法

いわゆる予定調和dp

dp[i][j][k][f]: i文字目まで処理して(Yに入ってないAの要素の最大値)=A_j、Xに入っているAの要素の個数がk、XにA_jが入っているかがf(bool)

- pushのとき
  - Aの要素じゃないのを入れる：i,j,kから選択肢の数がわかる
  - Aの要素を入れる：kをインクリメント（具体的には決めない）
- popのとき
  - kをデクリメント（このときもまだ決めない）
  - k=1,f=1のとき、j→j'の変化を決め打つ（このときにA_j'+1~A_j-1をどこでpushしたか選ぶ!）

統計情報

- AC / Trying teams
  - 1 / 2 (50%)

- First Acceptance
  - オンサイト参加: ? (??:??:??)
  - 全体: HoMaMaOvO (04:31:41)

目次　全体講評

Q - Quotient Sum

原案: tokusakurai
準備: tokusakurai
tester: tabr, miscalc

問題概要

- N 個の整数 A_1, …, A_N が与えられる
- これらを並び替えることで、以下の値を最小化せよ
  floor(A_2/A_1) + … + floor(A_N / A_{N-1}) + floor(A_1 / A_N)

制約

- $1 \leq N \leq 2 \times 10^5$
- $1 \leq A\_i \leq 10^{18}$
- A_i は相異なる

考察

最適解の 1 つは**山型**になる

$$a_1 < \ldots < a_k > \ldots > a_N > a_1$$

例：A = (1,2,3,4,5,6,7) の場合

1, 2, 5, 7, 6, 4, 3

問題の言い換え

- A を昇順ソート
- N 頂点の重み付き有向グラフを考える:

  i から j に重み **w(i,j) := floor(A_j / A_i)** の辺 (i < j)

  問題の言い換え : 1 から N への最短経路長はいくつか？

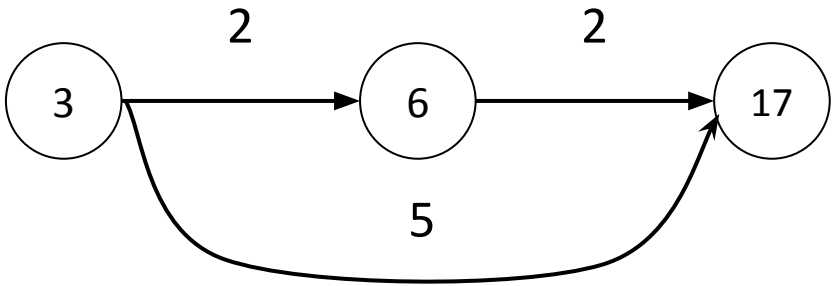
  愚直な DP : $O(N^2)$

DP 高速化

- 遷移先は**高々 2 個**見ればよい

  - w(i,j) が同じ中では j が最大のものだけに遷移
  - 上を満たす j の中で、**小さい方から 2 つ**のみに遷移すればよい

∵ 2 ≤ w(i,j) < w(i,k) なら **w(i,j) + w(j,k) ≦ w(i,k)**



まとめ

- 最初にソート
- DP の遷移先 2 個は二分探索で求められる

  計算量 O(N log N)

統計情報

- AC / Trying teams
  - 25 / 48 (52%)

- First Acceptance
  - オンサイト参加: japan406364961 (00:45:31)
  - 全体: japan406364961 (00:45:31)

目次　全体講評