SoundHound Inc. Programming Contest 2018

-Masters Tournament-

本戦 解説

2018.7.30

文責(A,B,C): evima

A - Feel the Beat: 問題概要

- 2 で何回か割ると区間 [140, 170) に収まる ~~BPM~~ 整数は C 以上 D 未満に何個あるか？

- $140 \leq C < D \leq 10^{15}$

A - Feel the Beat: 解法

- 整数を一つ一つ 2 で割っていくのではなく
  [140, 170) の方を 2 倍していけばよい

- [C, D) と [140, 170) の共通区間の長さ +
  [C, D) と [140×2, 170×2) の共通区間の長さ +
  [C, D) と [140×2^2, 170×2^2) の共通区間の長さ + … が答え

- O(logD) 時間

B - Neutralize: 問題概要

- 長さ N の ~~薬品~~ 整数列 $a_1, a_2, \ldots, a_N$ がある
- 連続する K 個を選んですべて 0 にする、という操作を何度でも行える
- 和 $a_1 + a_2 + \ldots + a_N$ を最大化せよ
- $1 \leq K \leq N \leq 10^5$, $1 \leq a_i \leq 10^9$

B - Neutralize: 解法

- 左から順に、各項を放置するか 0 にするか選んでいく
- 0 にするときは K 連続以上で 0 にしなければならない
- 逆に、上さえ守れば何をしてもよい
- 実装例…

  dp1[i] := $a_i, a_{i+1}, \ldots, a_N$ に対する問題の答え

  dp2[i] := 上とほぼ同じ、ただし a_i を単独で 0 にすることが許される

  dp1[i] の計算に dp2[i+K] を使う
- O(N) 時間

C - Not Too Close: 問題概要

- ラベル付き N 頂点の単純無向グラフであって
  頂点 1 と頂点 2 の最短距離が D であるものは何通りか？

- $1 \leq D < N \leq 30$

C - Not Too Close: 解法

- 頂点 2 を「その他大勢」の一部とみなし、
  頂点 1 から各頂点 i までの距離 d_i に注目する

- 重要な事実: 頂点 i と j が隣接するなら $|d_i - d_j| \leq 1$
  → 頂点 1 の近くからグラフを構成していけばよい

- dp[i][j][k] := 頂点 1 から距離 i-1 以下の部分が構築済みで、
  未使用の頂点が j 個、距離 i-1 の頂点が k 個のとき、
  グラフの残りの部分が何通り考えられるか（i = D のとき要注意）

- $O(N^4)$ 時間（前計算など省くと $O(N^6)$ まで増えうるが間に合う）

D: Propagating Edges

問題概要

- 無向グラフ(最初は辺がない)に飛んでくるクエリを処理する
- 辺の追加(add)
- 辺の存在判定(check)
- ある頂点から連結な頂点を列挙して，その頂点たちを完全グラフに(complete)

解法

- まずはこれだけだと？
  - 辺の追加(add)
  - 辺の存在判定(check)

- 簡単
  - ハッシュマップ，平衡木等
    - C++ならset<Edge>, set<pair<int, int>>など
    - JavaならHashMapなど

解法

- ある頂点から連結な頂点を列挙して，その頂点たちを完全グラフに(complete)
    - これで追加される辺だけ管理すれば良くなった
        - でもこのクエリは辺の数をとんでもないことにする(最大でO(N^2)本)
- データの持ち方を考える必要がある

解法

- ある頂点から連結な頂点を列挙して，その頂点たちを完全グラフに

- ある頂点から連結な頂点を列挙して，その頂点たちを一個の頂点にまとめる
    - こう考えても良い
    - こう考えると，頂点の間に辺があるか→頂点が同じ頂点にまとめられているか，と言い換えられる
- 頂点をまとめたり，同じ頂点にまとめられているか判定したりする…？
    - Union-Find(Disjoint Set Union)！

解法

- addクエリで追加される辺は全てmap<pair<int, int>>(Edgelist)等で持っておく
- completeクエリを処理するため，別にグラフ(グラフX)を持つ，このグラフでは頂点をどんどん縮約(いくつかの頂点を1つにまとめる)していく
- addクエリでは，Edgelistに辺を追加するだけでなく，グラフXで対応する頂点の間にも辺を貼る
- Completeクエリが来たら，グラフX上でdfsして，到達できた頂点を全部1つにまとめる
- これらの操作は全部Unionfindと，vector<set<edge>>等で可能

E: Hash Swapping

問題概要

- 文字列が20個ぐらいある
- 文字列同士で同じ位置の部分文字列をswap
- ある文字列についてのローリングハッシュを得る

解法

- 文字列に変更がないなら，部分文字列のローリングハッシュは累積和で定数時間で求められます

- 変更があると？

解法

- 数列が20個ぐらいある，以下のクエリを行う
    - 2つの数列の，対応する位置をswap
    - 数列の区間のsumを取得
- これが解ければいい
- 実は平衡木で高速に処理可能
    - std::setでは無理なので，ちゃんと自分で平衡木を書く必要があります

平衡木

- 赤黒木，AVL 木などと呼ばれるやつです
- 色々種類がある
    - 赤黒木，AVL 木(鉄板)
    - RBST, Treap(ランダム性を利用，競プロでよく使われる)
    - Weight-balanced Tree(個人的おすすめ)
- 実際の実装等は省略します(ごめんなさい)
    - 解説するとこれだけで終わってしまう
    - 調べれば日本語でも資料がたくさん出てきます

平衡木

- 平衡木は値の集合を管理するのによく使われる(std::set, treeset, …)
- でも，実は数列を管理するのにも使える
    - 切ったりつなげたりできるSegment treeみたいなものになる
    - 当然swapもできる(切って入れ替えてつなげる)
    - 区間sumをもたせればAC

平方分割

- コンテスト中に平衡木を1から実装するのは大変...
  - じゃあこの問題は平衡木を準備しておかない解決不能？
- 困ったときの平方分割
  - 数列をB = sqrt(N)個ずつのブロックに区切る
    - 各ブロックは，pair(int[B], ←の総和)へのポインタを20個持つ
    - Swapクエリは，端っこは愚直に入れ替えて，真ん中はポインタを入れ替える
    - Addクエリは，端っこは愚直に足して，真ん中は”←の総和”を足していく

おまけ

- 平衡木を使わずO(NM + QlogN)で解くことも可能です
  - 実は区間sumの取れるsegtreeを実装すれば，全く同じ形の木の全く同じところをswapすることになる
  - Segtreeを配列ではなく，ポインタを使って実装すれば交換可能です
  - 配列ベースのSegtreeでも，ノードにpermutationを持たす，みたいな実装をすれば可能です