ARC044 解説

DEGwer

A問題「素数判定」

問題概要

- 整数N(1≦N≦10^9)が素数っぽいかどうか判定せよ
- Nは以下のいずれかの条件を満たす時素数っぽい
  - 素数である
  - 合成数であり、かつ1の位が2でも5でも割り切れず、各桁の和が3で割り切れない

解法1

- 愚直に計算
- 素数判定にO($\sqrt{N}$)
- 各桁の和を求めるのにO(log N)
- あわせてO($\sqrt{N}$)

解法2

- 各桁の和とかは求めなくていい
  - Nの1の位が2でも5でも割り切れず、各桁の和が3の倍数であることは、Nが2,3,5で割り切れないことと同値
- 実は素数判定も要らない
  - 2,3,5以外のNについて、Nが素数ならNは2,3,5で割り切れない

解法2

- 以上をまとめると
  - N=1ならNot Prime(1は素数でも合成数でもない)
  - それ以外でN=2,3,5ならPrime
  - それ以外でNが2,3,5で割り切れるならNot Prime
  - それ以外ならPrime
- と出力すればよい
- O(1)で解けた

B問題「最短路問題」

問題概要

- 整数Nと整数列$A_1, A_2,..., A_N$が与えられる
- 頂点数Nですべての辺のコストが1の単純無向グラフであって、任意のiに対して頂点1から頂点iまでの距離がA_iであるようなものの個数をmod 10^9+7で求めよ
- N ≦ 10^5

解法

- まず、自明に答えが0になる場合を取り除く
  - $A_1 \neq 0$だとだめ(頂点1から頂点1への距離は0)
  - 0が複数あってもだめ(頂点1から頂点1以外への距離は正)

解法

- 2点間に辺を張れる条件は?
  - その2点の、A_1からの距離の差が1以下
  - 距離の差が2以上あると、その辺を使うことで最短距離の性質に矛盾する

- 2点間の距離が0の場合と1の場合に分けて考える

解法

- 2点間の距離が0のとき辺はどう張ってもいい
  - 辺があろうがなかろうが最短距離は変わらない
- $A_i = C$なるiがK_i個あるとき、距離Cの頂点の間に辺を張る場合の数は$2^{\frac{K_i(K_i-1)}{2}}$通り
- これは、N乗までの2冪をループで順にもとめていき、さらにそれらの掛け算で2の三角数乗を順に求めていけばあわせてO(N)で求まる
  - 二分累乗ならO(Nlog N)

解法

- 2点間の距離が1のとき、そのA_1からの距離をそれぞれi, i+1とし、A_1からの距離がiであるような頂点の数をt, i+1であるような頂点の数をsとする
- 距離i+1の頂点たちからは、1本以上の辺が距離iの頂点のいずれかに張られていなければならない
  - 逆に、それ以外はどう張ってもよい

解法

- このような場合の数は$(2^t-1)^s$通り
- これも2冪のテーブルをO(N)で前計算しておけば、合計O(N)で求まる
  - 頂点数の和はNよりこの操作にはO(s)かけられる
  - おなじく二分累乗ならO(Nlog N)
- ということでこの問題が解けた

C問題「ビーム」

問題概要

- W*Hのグリッドに高橋君がいる
- 1列全体、もしくは1行全体にQ回ビームが飛んでくる
- 高橋君は任意時刻にすきなだけ縦横に移動できる
- 全部よけるとき、高橋君の縦横への移動回数の最小値は?
- W,H,Q,ビームが飛んでくる時刻 ≦ 10^5

部分点解法(30点)

- W,H,Q≦ 100
- 1 ≦ビームが飛んでくる時刻(この最大値をTとする) ≦ 100

- DP[i][j][k]: 時刻iに、マス(j,k)にいるときの最小の移動回数 として、DP配列を更新していく
- 各iごとにdijkstra法などを用いてDP配列を順次更新していくと、O(WHT log WH)

満点解法

- x座標、y座標は独立に考えられる(重要アイデア)
  - ビームは一列、もしくは一行すべてに飛んできて、高橋君の移動距離の定義もマンハッタン距離
- x座標、y座標独立に問題を解いて、あとで足し合わせればいい
- 以下x座標についてのみ考える

満点解法

- DP[i][j]: 時刻iに座標jにいるときの移動距離の最小値 とする
- これを愚直に計算するとO((W+H)T)
- よく考えると、DP[i][j]たちとDP[i+1][j]たちは「あまりかわらない」
  - 具体的には、異なる箇所は時刻i+1に飛んでくるビームの本数分のみ

満点解法

- 結局、DP配列を使いまわしてこれらが切り替わる点のみ更新すれば、O(W+H+T)でこの問題が解ける

D問題「suffix array」

問題概要

- 10^6項以下のpermutationが与えられる
- このpermutationをsuffix arrayに持つ辞書順最小の文字列を求めよ

解法

- 順列$A_1, A_2, \ldots, A_N$が文字列$S_1 S_2 \ldots S_N$のsuffix arrayである
- ⇔すべての1≦i≦N-1に対し、文字列$S_{A_i} S_{A_i+1} \ldots S_N < S_{A_{i+1}} S_{A_{i+1}+1} \ldots S_N$
- ⇔すべての1≦i≦N-1に対し、$S_{A_i} < S_{A_{i+1}}$または、$S_{A_i} = S_{A_{i+1}}$かつ$S_{A_i+1} \ldots S_N < S_{A_{i+1}+1} \ldots S_N$

解法

- すべての1≦i≦N-1に対し、$S_{A_i} < S_{A_{i+1}}$または、$S_{A_i} = S_{A_{i+1}}$かつ$S_{A_i+1} \dots S_N < S_{A_{i+1}+1} \dots S_N$
- ⇔すべての1≦i≦N-1に対し、
  - $S_{A_i+1} \dots S_N < S_{A_{i+1}+1} \dots S_N$ならば$S_{A_i} \leqq S_{A_{i+1}}$
  - そうでないなら、$S_{A_i} < S_{A_{i+1}}$

解法

- $S_{A_i+1} \dots S_N < S_{A_{i+1}+1} \dots S_N$かどうかは、suffix arrayが持っている情報
- ということは、各iについて、S_{A_i}と$S_{A_{i+1}}$の満たすべき関係がわかる
- 前から順に見ていけば、文字列の各文字が最も小さくて何であるかがすべてわかる

解法

- 逆に、suffix arrayを前から順に見ていき、このような方法で各文字を決めていけば、そうしてできた文字列のsuffix arrayは元の順列に一致することは簡単に示せる
- よって、前から貪欲に文字を決めていけば、O(N)でこの問題が解けた