E:LISum

原案:yebi

ジャッジ解:T.M,noy,btk

解説:T.M

概要

- 数列が与えられる
- LISのうち総和の最大値は？

DP（想定TLE）

- dpl[i]=i番目を末尾にした時の増加列の最大の長さ
- dps[i]=i番目を末尾にした時の増加列の最大の総和

- 増加列の末尾の値、合計、長さがわかればそれ以外はどうでもいい

遷移

- i番目をj番目の後ろに追加することを考える
- d[j]<d[i]　　//末尾より大きい
- dpl[i]<dpl[j]+1　　//暫定一位より長い
- dpl[i]==dpl[j]+1&&dps[i]<dps[j]+d[i]
  //暫定一位と同じ長さで合計が暫定一位より大きい
- この条件を満たすなら暫定一位を更新する
- dpl[i]=dpl[j]+1
- dps[i]=dps[j]+d[i]

最後に答えを見つける

- 最も長く、最も大きいものを探せばOK
- 全体でO(N^2)
- 間に合わない

高速化する

- dpl[i]=i番目に大きいものを末尾にした時の増加列の最大の長さ
- dps[i]=i番目に大きいものを末尾にした時の増加列の最大の総和
- 列の先頭から順番に更新するのではなく小さい順に埋めていく

- 座圧してi番目に大きいもの→iでもOK

遷移

- i番目に大きいものを末尾に追加するとする

- dpl[i]=MAX(dpl[j])(j<i)+1
- dps[i]=MAX(dps[i])(j<i)+d[i]

- MAXの部分はセグ木を使って高速化
- 値が大きくなり続けるのでBITでもOK
- O(NlogN)

別解

- 経路復元でLISを構築する
- するとその総和は一番小さくなる
- 同じ長さなら後ろのことを考えて小さいほうを取るから
- つまり……

後ろからLDSを求める

- 後ろから最長減少部分列を求める
- 前からは最長増加部分列になる
- LDSは後(?)のことを考えて大きいほうを取る
- そのため復元すると総和が最大になる

- O(NlogN)
- がんばると前からも求められる

証明の雰囲気

- ある数字より大きい数字が来る
- 総和は大きくなり、後にも使いやすいため保存される

- ある数字より小さい数字が来る
- LDSが長くなるので使うしかない
- 長くなるので保存される

ジャッジ解

• T.M（前から）	C	32行	682B
• noy	C++	66行	1.56KB
• btk(BIT)	C++	62行	1.21KB

いつもの

- オンサイト
- FA　　homtemchonさん　　16min

- オンライン
- FA　　homtemchonさん　　16min

- AC/Submit　　45/93　　48%