Correzione *divisors*

sabato 10 aprile 2021 14:21

$$\sum_{k=1}^{n} d(k), \quad d(k) = \#\text{divisori positivi di } n$$

$n \le 10^{12} \rightsquigarrow O(\sqrt{n})$

Soluzione

- $n \le 1000$: soluzione naive ($\underbrace{\text{calcolo } d(k)}_{\text{scorro gli interi } 1 \le i \le k \text{ e controllo se sono divisori}}$ per ogni k)

- $n \le 100000$: stessa cosa di sopra, ma osservo che, se $d \mid k$, allora $\frac{k}{d} \mid k$.

  questo mi permette di restringermi a $i \in \{1, \dots, \lfloor \sqrt{k} \rfloor\}$

- $n \le 10^6$: double-counting su $\sum_{k=1}^{n} d(k)$

  posso vederla come: per ogni $1 \le d \le n$, per quanti k distinti si ha che $d \mid k$? $\lfloor \frac{n}{d} \rfloor$ $(d, 2d, 3d, \dots, \lfloor \frac{n}{d} \rfloor d)$

  quindi: $ans = \sum_{d=1}^{n} \lfloor \frac{n}{d} \rfloor$

- $n \le 10^{12}$: altra idea: $\lfloor \frac{n}{d} \rfloor$ assume "pochi" valori distinti

  posso partizionare $\{1, \dots, n\}$ in intervalli, su ciascuno dei quali $\lfloor \frac{n}{d} \rfloor$ è costante

  Supponiamo che $\lfloor \frac{n}{d} \rfloor = m \Rightarrow \boxed{n = dm + r}$, $0 \le r < d$

  per quali $i \ge 0$ $\lfloor \frac{n}{d+i} \rfloor = m$? $\quad \hookrightarrow n = (d+i)m + (r - im)$

  mi serve $r - im \ge 0 \Rightarrow i \le \lfloor \frac{r}{m} \rfloor$

  algoritmo:
  - parto da $d = 1$
  - while $d \le n$, calcolo i e sommo $\lfloor \frac{n}{d} \rfloor \cdot (i+1)$ ad ans

    $(d+0, d+1, \dots, d+i)$

  ma quanti sono questi intervalli? $O(\sqrt{n})$

  idea: divido $\underbrace{1, 2, 3, 4, \sqrt{n}}_{\text{gli intervalli sono lunghi} \ge 1} \underbrace{\dots\ n}_{\text{gli intervalli sono lunghi} \ge c \cdot \sqrt{n}}$

Correzione *figurines*

sabato 10 aprile 2021 14:21

vengono dati $n+1$ sottoinsiemi: $S(0), S(1), \dots, S(n) \subseteq \{0, \dots, n-1\}$

$S(i+1)$ è ottenuto da $S(i)$ tramite aggiunta / rimozione di elementi

ogni elemento è aggiunto e rimosso solo una volta

$S(0) = S(n) = \emptyset$

query (online): $(d, x) \rightsquigarrow$ quanti sono gli $y \geq x$ in $S(d)$?

Soluzione

ST inesistente: se (d_1, x_1) e (d_2, x_2) sono query consecutive, $d_1 \leq d_2$

allora è sufficiente un segment tree!

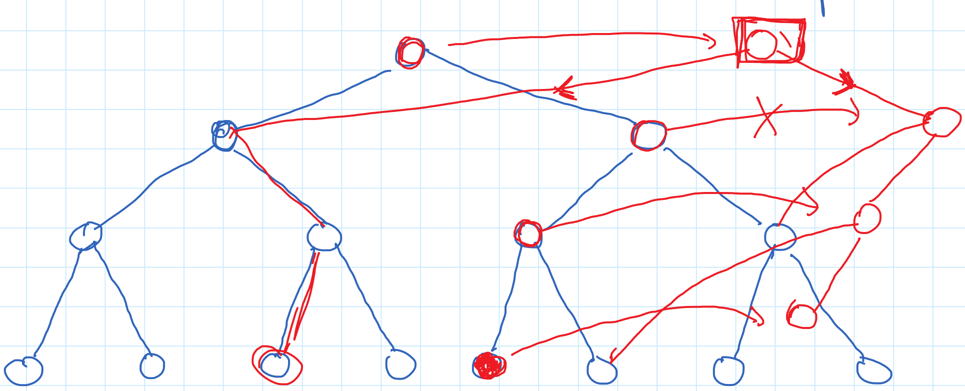
Soluzione da 100: uso un segment tree persistente.

DS persistente: è una DS che viene "copiata" a ogni modifica

Come si fa in modo intelligente?

Osservazione: quando faccio un update, modifico solo $O(\log n)$ nodi

$\rightsquigarrow$ m. basta duplicare solo loro!



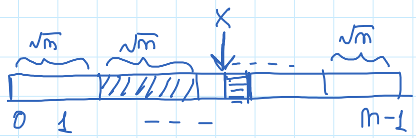
Soluzione 2

Merge sort tree (feat. Valerio Stancanelli)

Soluzione 3

$\sqrt{n}$ $\sqrt{n}$ x $\sqrt{n}$

Soluzione 3



Divido $\{0, ..., n-1\}$ in blocchi da $\sqrt{n}$

Per ogni blocco $\{a, ..., a+\lfloor\sqrt{n}\rfloor-1\}$ e per ogni d, mi salvo quanti x appartenenti al blocco sono tali che il loro intervallo contiene d.

Correzione *subtreemode*

sabato 10 aprile 2021 14:21

abbiamo un albero su n nodi, con una label su ogni nodo
per ogni sottoalbero, voglio sapere la label più frequente.

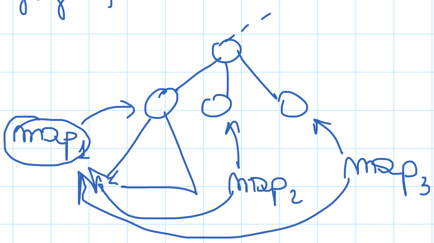
<u>Soluzione</u>

Il modo naive è avere $\underbrace{\text{una map}}_{\text{ma anche un array!}}$: labels → frequenza per ogni sottoalbero.

C'è modo di risparmiare tempo sull'operazione di merge dei figli?

Sì, se uso una map.

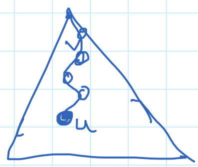
Idea: scelgo un figlio, e uso la sua map come "accettore" delle map degli altri figli.



scelgo la map del figlio più "pesante"

Questa tecnica permette di sostituire un fattore n con uno log n.

Dimostrazione: per esercizio.



$$\sum_{v \text{ nodo}} \underbrace{\sum_{\substack{u \in sons_v \\ u \neq heavy}} size(u)} = O(n \log n)$$

complessità: $O(n \log^2 n)$ o $O(n \log n)$ ← la costante è gigante!

questa soluzione prende tutti i subtask tranne l'ultimo.

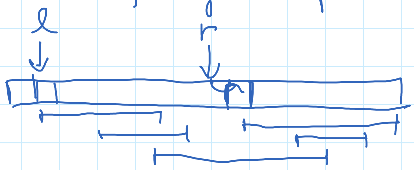
Proviamo qualcosa di diverso.

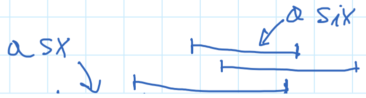
Appiattisco l'albero. faccio una BFS e metto i nodi in un array in ordine di visita.

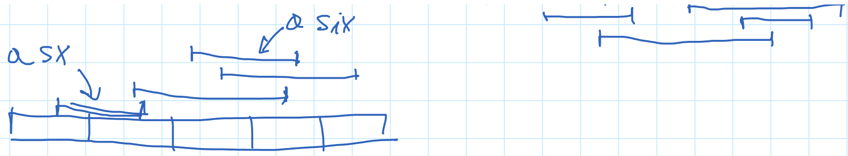
Proprietà interessante: ogni sottoalbero è un intervallo.

Mi sono ridotto al seguente problema: ho degli intervalli di un array e, per ciascuno di essi, voglio sapere la moda.

Uso <u>Mo's Algorithm</u>.







Questo mi dà una soluzione $O(n\sqrt{n} \log n)$.

La soluzione da 100 si ottiene combinando le idee dei due approcci precedenti