Sorting and Searching

Lorenzo Ferrari, Davide Bartoli

April 28, 2023

Table of contents

Subarray Distinct Values

Subarray Distinct Values

Dato un array a di $n \leq 2 \cdot 10^5$ interi, conta il numero di subarray con al più k valori distinti.

https://cses.fi/problemset/task/2428

Subarray Distinct Values

Subarray Distinct Values

Dato un array a di $n \leq 2 \cdot 10^5$ interi, conta il numero di subarray con al più k valori distinti.

https://cses.fi/problemset/task/2428

- idee?

Subarray Distinct Values

Subarray Distinct Values

Dato un array a di $n \leq 2 \cdot 10^5$ interi, conta il numero di subarray con al più k valori distinti.

https://cses.fi/problemset/task/2428

- idee?
- notiamo che, dato un array con $\leq k$ elementi distinti, anche ogni suo sottoarray ha $\leq k$ elementi distinti

Subarray Distinct Values

Subarray Distinct Values

Dato un array a di $n \leq 2 \cdot 10^5$ interi, conta il numero di subarray con al più k valori distinti.

https://cses.fi/problemset/task/2428

- idee?
- notiamo che, dato un array con $\leq k$ elementi distinti, anche ogni suo sottoarray ha $\leq k$ elementi distinti
- per ogni indice i, troviamo il massimo j tale che $\{a_i, a_{i+1}, \ldots, a_j\}$ ha al più k elementi distinti.

Subarray Distinct Values

Subarray Distinct Values

Dato un array a di $n \leq 2 \cdot 10^5$ interi, conta il numero di subarray con al più k valori distinti.

https://cses.fi/problemset/task/2428

- idee?
- notiamo che, dato un array con $\leq k$ elementi distinti, anche ogni suo sottoarray ha $\leq k$ elementi distinti
- per ogni indice i, troviamo il massimo j tale che $\{a_i, a_{i+1}, \ldots, a_j\}$ ha al più k elementi distinti.
- anche tutti i subarray $a[i:i], a[i:i+1], \ldots, a[i:j]$ (in totale $j-i+1$) hanno al più k valori distinti

Subarray Distinct Values

Subarray Distinct Values

Dato un array a di $n \leq 2 \cdot 10^5$ interi, conta il numero di subarray con al più k valori distinti.

https://cses.fi/problemset/task/2428

- idee?
- notiamo che, dato un array con $\leq k$ elementi distinti, anche ogni suo sottoarray ha $\leq k$ elementi distinti
- per ogni indice i, troviamo il massimo j tale che $\{a_i, a_{i+1}, \ldots, a_j\}$ ha al più k elementi distinti.
- anche tutti i subarray $a[i:i], a[i:i+1], \ldots, a[i:j]$ (in totale $j-i+1$) hanno al più k valori distinti
- iterando su tutti gli i, stiamo contanto tutti i subarray

Tasks and Deadlines

https://cses.fi/problemset/task/2428

Tasks and Deadlines

https://cses.fi/problemset/task/2428

- inizialmente il problema sembra complesso, ci sono tante variabili da considerare.

Tasks and Deadlines

https://cses.fi/problemset/task/2428

- inizialmente il problema sembra complesso, ci sono tante variabili da considerare.
- un'idea comune in questo tipo di problemi é cercare di trovare un ordinamento ottimale (potrebbe non esistere).

Tasks and Deadlines

https://cses.fi/problemset/task/2428

- inizialmente il problema sembra complesso, ci sono tante variabili da considerare.
- un'idea comune in questo tipo di problemi é cercare di trovare un ordinamento ottimale (potrebbe non esistere).
- trovando un ordinamento adatto, potremmo risolvere il problema in $O(n \log n + f(n))$, dove $f(n)$ è il costo per processare n elementi ordinati in modo ottimale.

Tasks and Deadlines

Dimostrazione

Consideriamo il seguente caso: ci troviamo all'istante k e dobbiamo scegliere quale task da fare prima tra x e y.

Tasks and Deadlines

Dimostrazione

Consideriamo il seguente caso: ci troviamo all'istante k e dobbiamo scegliere quale task da fare prima tra x e y.

- se facciamo prima x allora otteniamo $d_x - (k + a_x) + d_y - (k + a_x + a_y) = d_x + d_y - 2k - 2a_x - a_y$
- se invece facciamo prima y allora otteniamo $d_y - (k + a_y) + d_x - (k + a_y + a_x) = d_x + d_y - 2k - 2a_y - a_x$

Tasks and Deadlines

Dimostrazione

Consideriamo il seguente caso: ci troviamo all'istante k e dobbiamo scegliere quale task da fare prima tra x e y.

- se facciamo prima x allora otteniamo $d_x - (k + a_x) + d_y - (k + a_x + a_y) = d_x + d_y - 2k - 2a_x - a_y$
- se invece facciamo prima y allora otteniamo $d_y - (k + a_y) + d_x - (k + a_y + a_x) = d_x + d_y - 2k - 2a_y - a_x$
- quando ci conviene fare prima x? quando $d_x + d_y - 2k - 2a_x - a_y \geq d_x + d_y - 2k - 2a_y - a_x$

Tasks and Deadlines

Dimostrazione

Consideriamo il seguente caso: ci troviamo all'istante k e dobbiamo scegliere quale task da fare prima tra x e y.

- se facciamo prima x allora otteniamo $d_x - (k + a_x) + d_y - (k + a_x + a_y) = d_x + d_y - 2k - 2a_x - a_y$
- se invece facciamo prima y allora otteniamo $d_y - (k + a_y) + d_x - (k + a_y + a_x) = d_x + d_y - 2k - 2a_y - a_x$
- quando ci conviene fare prima x? quando $d_x + d_y - 2k - 2a_x - a_y \geq d_x + d_y - 2k - 2a_y - a_x$
  $-2a_x - a_y \geq -2a_y - a_x$

Tasks and Deadlines

Dimostrazione

Consideriamo il seguente caso: ci troviamo all'istante k e dobbiamo scegliere quale task da fare prima tra x e y.

- se facciamo prima x allora otteniamo $d_x - (k + a_x) + d_y - (k + a_x + a_y) = d_x + d_y - 2k - 2a_x - a_y$
- se invece facciamo prima y allora otteniamo $d_y - (k + a_y) + d_x - (k + a_y + a_x) = d_x + d_y - 2k - 2a_y - a_x$
- quando ci conviene fare prima x? quando
  $d_x + d_y - 2k - 2a_x - a_y \geq d_x + d_y - 2k - 2a_y - a_x$
  $-2a_x - a_y \geq -2a_y - a_x$
  $-a_x \geq -a_y$
  $a_x \leq a_y$ non dipende da k!

Tasks and Deadlines

Dimostrazione

Consideriamo il seguente caso: ci troviamo all'istante k e dobbiamo scegliere quale task da fare prima tra x e y.

- se facciamo prima x allora otteniamo $d_x - (k + a_x) + d_y - (k + a_x + a_y) = d_x + d_y - 2k - 2a_x - a_y$
- se invece facciamo prima y allora otteniamo $d_y - (k + a_y) + d_x - (k + a_y + a_x) = d_x + d_y - 2k - 2a_y - a_x$
- quando ci conviene fare prima x? quando
  $d_x + d_y - 2k - 2a_x - a_y \geq d_x + d_y - 2k - 2a_y - a_x$
  $-2a_x - a_y \geq -2a_y - a_x$
  $-a_x \geq -a_y$
  $a_x \leq a_y$ non dipende da k!

Possiamo ordinare i task in base alla loro durata e risolvere il problema in $O(n \log n)$.

Problemi

ois_antennas

ois_antennas

N antenne sono in fila. Ognuna è caratterizzata dalla sensibilità di L_i decibel, la potenza di P_i decibel e due interi S_i, T_i.

- l'antenna i riceve un segnale se arriva con una potenza $\geq L_i$ decibel.
- ogni antenna può sempre ricevere segnali, ma trasmette solo negli istanti $S_i, S_i + T_i, S_i + 2T_i, \dots$
- i segnali viaggiano solo verso destra
- un segnale raggiunge istantaneamente tutte le antenne, ma la potenza diminuisce di D decibel passando tra due antenne consecutive

Trova l'istante in cui l'antenna $N - 1$ riceve da 0.

https://training.olinfo.it/#/task/ois_antennas/statement

Problemi

ois_antennas

- idee?

Problemi

ois_antennas

- idee?
- soluzione $O(N^2)$
  - processiamo le antenne in ordine
  - per controllare se e quanto l'antenna i riceve il segnale, controllo quando le antenne $0, 1, \ldots, i-1$ trasmettono il segnale
  - la soluzione è corretta, ma non abbastanza efficiente

Problemi

ois_antennas

- idee?
- soluzione $O(N^2)$
  - processiamo le antenne in ordine
  - per controllare se e quanto l'antenna i riceve il segnale, controllo quando le antenne $0, 1, \ldots, i-1$ trasmettono il segnale
  - la soluzione è corretta, ma non abbastanza efficiente

osservazione: un segnale (p_a, t_a) che arriva al tempo t_a con potenza p_a, è sicuramente meglio di tutti i segnali (p_b, t_b) con $p_a > p_b$ e $t_a < t_b$.

Problemi

ois_antennas

- idee?
- soluzione $O(N^2)$
  - processiamo le antenne in ordine
  - per controllare se e quanto l'antenna i riceve il segnale, controllo quando le antenne $0, 1, \ldots, i-1$ trasmettono il segnale
  - la soluzione è corretta, ma non abbastanza efficiente

osservazione: un segnale (p_a, t_a) che arriva al tempo t_a con potenza p_a, è sicuramente meglio di tutti i segnali (p_b, t_b) con $p_a > p_b$ e $t_a < t_b$. Al contrario non possiamo dire nulla su due segnali $(p_a, t_a), (p_b, t_b)$ con $p_a > p_b$ e $t_a > t_b$.

Problemi

ois_antennas

Risolviamo il subtask $D = 0$, quello in cui la potenza non diminuisce viaggiando tra antenne successive.

- teniamo un set dei segnali *potenzialmente* migliori, ossia un set $(p_a, t_b), (p_b, t_b), \ldots, (p_k, t_k)$ con $p_a < p_b < \cdots < p_k$ e $t_a < t_b < \cdots < t_k$.

Problemi

ois_antennas

Risolviamo il subtask $D = 0$, quello in cui la potenza non diminuisce viaggiando tra antenne successive.

- teniamo un set dei segnali *potenzialmente* migliori, ossia un set $(p_a, t_b), (p_b, t_b), \ldots, (p_k, t_k)$ con $p_a < p_b < \cdots < p_k$ e $t_a < t_b < \cdots < t_k$.
- dobbiamo scrivere una struttura dati che supporti:

Problemi

ois_antennas

Risolviamo il subtask $D = 0$, quello in cui la potenza non diminuisce viaggiando tra antenne successive.

- teniamo un set dei segnali *potenzialmente* migliori, ossia un set $(p_a, t_b), (p_b, t_b), \ldots, (p_k, t_k)$ con $p_a < p_b < \cdots < p_k$ e $t_a < t_b < \cdots < t_k$.
- dobbiamo scrivere una struttura dati che supporti:
  - inserimento di una coppia (p, t)
    - complessità $O(\log n)$ ammortizzato

Problemi

ois_antennas

Risolviamo il subtask $D = 0$, quello in cui la potenza non diminuisce viaggiando tra antenne successive.

- teniamo un set dei segnali *potenzialmente* migliori, ossia un set $(p_a, t_b), (p_b, t_b), \ldots, (p_k, t_k)$ con $p_a < p_b < \cdots < p_k$ e $t_a < t_b < \cdots < t_k$.
- dobbiamo scrivere una struttura dati che supporti:
  - inserimento di una coppia (p, t)
    - complessità $O(\log n)$ ammortizzato
  - trovare il primo segnale con potenza $\geq L_i$
    - complessità $O(\log n)$

```cpp
struct cool_map {
    map<LL, LL> m;

    void insert(LL p, LL t) {
        {
            auto it = m.lower_bound(p);
            if (it != m.end() && it->second <= t) {
                return;
            }
        }
        m[p] = t;
        auto it = m.find(p);
        while (it != m.begin()) {
            if (prev(it)->second >= t) {
                m.erase(prev(it));
            } else {
                break;
            }
        }
    }
    LL get_time(LL l) {
        auto it = m.lower_bound(l);
        return it == m.end() ? -1 : it->second;
    }
};
```

Problemi

ois_antennas

Abbiamo risolto il problema per $D = 0$, ma la potenza di ogni segnale diminuisce di D ogni volta.

Problemi

Abbiamo risolto il problema per $D = 0$, ma la potenza di ogni segnale diminuisce di D ogni volta.
Manteniamo una variabile `d` che indica di quanto i segnali (le key) nella `map` sono maggiori rispetto ai segnali reali.

Problemi

Abbiamo risolto il problema per $D = 0$, ma la potenza di ogni segnale diminuisce di D ogni volta.
Manteniamo una variabile `d` che indica di quanto i segnali (le key) nella `map` sono maggiori rispetto ai segnali reali.

- possiamo dimiuire tutti i segnali nella mappa semplicemente con `d += D`
- come possiamo inserire un segnale `p` al tempo `t`?

Problemi

ois_antennas

Abbiamo risolto il problema per $D = 0$, ma la potenza di ogni segnale diminuisce di D ogni volta.
Manteniamo una variabile `d` che indica di quanto i segnali (le key) nella `map` sono maggiori rispetto ai segnali reali.

- possiamo dimiuire tutti i segnali nella mappa semplicemente con `d += D`
- come possiamo inserire un segnale `p` al tempo `t`?

Problemi

ois_antennas

Abbiamo risolto il problema per $D = 0$, ma la potenza di ogni segnale diminuisce di D ogni volta.
Manteniamo una variabile `d` che indica di quanto i segnali (le key) nella `map` sono maggiori rispetto ai segnali reali.

- possiamo dimiuire tutti i segnali nella mappa semplicemente con `d += D`
- come possiamo inserire un segnale `p` al tempo `t`?
  - tutti i segnali (key) sono maggiori di `d` rispetto ai segnali reali

Problemi

ois_antennas

Abbiamo risolto il problema per $D = 0$, ma la potenza di ogni segnale diminuisce di D ogni volta.
Manteniamo una variabile `d` che indica di quanto i segnali (le key) nella `map` sono maggiori rispetto ai segnali reali.

- possiamo dimiuire tutti i segnali nella mappa semplicemente con `d += D`
- come possiamo inserire un segnale `p` al tempo `t`?
  - tutti i segnali (key) sono maggiori di `d` rispetto ai segnali reali
  - non inseriamo la coppia `(p, t)`, ma `(p+d, t)`

```cpp
struct cool_map {
    LL d = 0;
    map<LL, LL> m;

    void decrease(LL td) { d += td; }
    LL get_time(LL l) {
        auto it = m.lower_bound(l + d);
        return it == m.end() ? -1 : it->second;
    }
};
```

```cpp
void insert(LL p, LL t) {
    p += d;
    {
        auto it = m.lower_bound(p);
        if (it != m.end() && it->second <= t) {
            return;
        }
    }
    m[p] = t;
    auto it = m.find(p);
    while (it != m.begin()) {
        if (prev(it)->second >= t) {
            m.erase(prev(it));
        } else {
            break;
        }
    }
}
```

Problemi

https://cses.fi/problemset/task/2428
https://cses.fi/problemset/task/1630
https://cses.fi/problemset/task/1645
https://cses.fi/problemset/task/1661
https://cses.fi/problemset/task/2168
https://training.olinfo.it/#/task/ois_antennas/statement
https://training.olinfo.it/#/task/ois_videogame/statement
https://training.olinfo.it/#/task/ois_intervals/statement
https://training.olinfo.it/#/task/ois_wine/statement
https://training.olinfo.it/#/task/preoii_pancake/statement