Primo allenamento

Olimpiadi Italiane di Informatica - Selezione territoriale

Luca Chiodini
luca@chiodini.org

13 marzo 2019

Struttura delle OII - IOI

Dove siamo?

Struttura delle OII - IOI

Dove siamo?

- Selezione scolastica (15 novembre 2018)

Struttura delle OII - IOI

Dove siamo?

- Selezione scolastica (15 novembre 2018)
- Selezione territoriale (16 aprile 2019)

Struttura delle OII - IOI

Dove siamo?

- Selezione scolastica (15 novembre 2018)
- Selezione territoriale (16 aprile 2019)
- Finale nazionale (Matera, 17-19 settembre 2019)

Struttura delle OII - IOI

Dove siamo?

- Selezione scolastica (15 novembre 2018)
- Selezione territoriale (16 aprile 2019)
- Finale nazionale (Matera, 17-19 settembre 2019)
- Finale internazionale (Singapore, 19-26 luglio 2020)

Struttura delle OII - IOI

Dove siamo?

- Selezione scolastica (15 novembre 2018)
- Selezione territoriale (16 aprile 2019)
- Finale nazionale (Matera, 17-19 settembre 2019)
- Finale internazionale (Singapore, 19-26 luglio 2020)

Per superare le territoriali...

Per superare le territoriali...

Serve

- saper scrivere codice C/C++

Per superare le territoriali...

Serve

- saper scrivere codice C/C++
- conoscere algoritmi, tecniche e idee per risolvere problemi

Per superare le territoriali...

Serve

- saper scrivere codice C/C++
- conoscere algoritmi, tecniche e idee per risolvere problemi
- esercitarsi risolvendo vecchi problemi

Per superare le territoriali...

Serve

- saper scrivere codice C/C++
- conoscere algoritmi, tecniche e idee per risolvere problemi
- esercitarsi risolvendo vecchi problemi

Programma

1. Lettura e analisi di un problema
2. Soluzione naïve
3. Spiegazione teorica
4. Soluzione ottima

Lettura e analisi di un problema

“Anno luce” (finale OIS 2016)

“Anno luce” (finale OIS 2016)

Grazie ad una recente ricerca che ha confermato l’esistenza delle onde gravitazionali, sempre più persone si stanno interessando allo spazio. Purtroppo però, lo spazio è ancora una realtà poco accessibile alle persone comuni. Sebbene sia un po’ demoralizzato da questo fatto, William è convinto che sia possibile sfruttare la recente attenzione mediatica delle onde gravitazionali per pubblicizzare un business: ha deciso infatti di aprire una startup di viaggi interstellari.

C’è da dire però che, a parte il Sole che è la stella a noi più vicina, le altre stelle sono piuttosto distanti. Proxima Centauri, la “seconda stella più vicina”, dista dal Sole ben 4.24 anni luce: questo vuol dire che sarebbero necessari più di quattro anni per raggiungere questa stella! (supponendo di poter viaggiare alla velocità della luce).

“Anno luce” (finale OIS 2016)

William pensa di riuscire a costruire un’astronave in grado di viaggiare alla velocità della luce (ha trovato un tutorial su YouTube che gli sembra piuttosto convincente) e ha perciò acquistato un telescopio per tracciare una mappa 3D della Via Lattea. Ogni stella è indicata nella mappa 3D mediante un punto (x, y, z) dello spazio. Il Sole è sempre presente nella mappa ed è sempre identificato dal punto $(0, 0, 0)$.

Scrivi un programma che data la mappa stellare sia in grado di rispondere a Q query: ogni query fornisce un numero intero D e chiede quante sono le stelle raggiungibili avendo a disposizione D anni di viaggio.

“Anno luce” (finale OIS 2016)

input.txt	output.txt
3 0 0 0 2 2 2 1 1 1 5 0 1 2 3 4	1 1 2 2 3

Soluzione naïve

“Anno luce” - Soluzione naïve

Coordinate

Ogni stella è caratterizzata dall’avere tre coordinate (x, y, z). Cosa è *davvero* interessante?

“Anno luce” - Soluzione naïve

Coordinate

Ogni stella è caratterizzata dall’avere tre coordinate (x, y, z). Cosa è *davvero* interessante?

Distanza

Calcoliamo la distanza euclidea nello spazio tra ciascuna stella e il Sole.

$$distanza = \sqrt{x^2 + y^2 + z^2}$$

“Anno luce” - Soluzione naïve

Riprendiamo l’esempio iniziale e visualizziamo cosa abbiamo ottenuto, supponendo di memorizzare la distanza dell’i-esima stella nell’i-esima posizione di un vettore `distanza`.

“Anno luce” - Soluzione naïve

Riprendiamo l’esempio iniziale e visualizziamo cosa abbiamo ottenuto, supponendo di memorizzare la distanza dell’i-esima stella nell’i-esima posizione di un vettore `distanza`.

input.txt	output.txt
3	1
0 0 0	1
2 2 2	2
1 1 1	2
5	3
0	
1	
2	
3	
4	

0	$2\sqrt{3}$	$\sqrt{3}$

“Anno luce” - Soluzione naïve

0	$2\sqrt{3}$	$\sqrt{3}$

Data una query (ovvero il numero di anni luce per cui è concesso viaggiare), qual è la risposta?

“Anno luce” - Soluzione naïve

0	$2\sqrt{3}$	$\sqrt{3}$

Data una query (ovvero il numero di anni luce per cui è concesso viaggiare), qual è la risposta?

È sufficiente iterare su ogni posizione del vettore e controllare se l’elemento è minore del numero di anni luce specificato nella query!

“Anno luce” - Soluzione naïve

0	$2\sqrt{3}$	$\sqrt{3}$

Data una query (ovvero il numero di anni luce per cui è concesso viaggiare), qual è la risposta?

È sufficiente iterare su ogni posizione del vettore e controllare se l’elemento è minore del numero di anni luce specificato nella query!

Esempio

Se $D = 3$, la risposta al problema è 2.

“Anno luce” - Soluzione naïve

```
int query(int D) {
    int raggiungibili = 0;

    for (int i = 0; i < N; i++)
        if (distanza[i] <= D)
            raggiungibili++;

    return raggiungibili;
}
```

Spiegazione teorica

Efficienza di una soluzione

Misura dell'efficienza

Come è possibile valutare quanto è "buono" un algoritmo? Si possono misurare, essenzialmente, due parametri: tempo di esecuzione e memoria occupata.

Efficienza di una soluzione

Misura dell'efficienza

Come è possibile valutare quanto è "buono" un algoritmo? Si possono misurare, essenzialmente, due parametri: tempo di esecuzione e memoria occupata.

Misura del tempo

- Numero di linee di codice
- Tempo di CPU

Efficienza di una soluzione

Misura dell'efficienza

Come è possibile valutare quanto è "buono" un algoritmo? Si possono misurare, essenzialmente, due parametri: tempo di esecuzione e memoria occupata.

Misura del tempo

- Numero di linee di codice
- Tempo di CPU
- *Numero di istruzioni eseguite*

Contare il numero istruzioni

Esempio: quante istruzioni vengono eseguite?

```
int query(int D) {
    int raggiungibili = 0;

    for (int i = 0; i < N; i++)
        if (distanza[i] <= D)
            raggiungibili++;

    return raggiungibili;
}
```

Contare il numero istruzioni

Esempio: quante istruzioni vengono eseguite?

```
int query(int D) {
    int raggiungibili = 0;

    for (int i = 0; i < N; i++)
        if (distanza[i] <= D)
            raggiungibili++;

    return raggiungibili;
}
```

$$T(n) = 1 + 1 + (n + 1) + (n + 1) + n + n + 1 = 4n + 5$$

Soluzioni *troppo* lente

Confrontiamo tre algoritmi con il seguente numero di istruzioni: $1000N$, $100N^2$, 2^N. Qual è il più veloce*?

* Assumendo 10^6 operazioni al secondo.

Soluzioni *troppo* lente

Confrontiamo tre algoritmi con il seguente numero di istruzioni: $1000N$, $100N^2$, 2^N. Qual è il più veloce*?

* Assumendo 10^6 operazioni al secondo.

N	t($1000N$)	t($100N^2$)	t(2^N)
10	10 ms	10 ms	1 ms
20	20 ms	40 ms	1 s

Soluzioni *troppo* lente

Confrontiamo tre algoritmi con il seguente numero di istruzioni: $1000N$, $100N^2$, 2^N. Qual è il più veloce*?

* Assumendo 10^6 operazioni al secondo.

N	t($1000N$)	t($100N^2$)	t(2^N)
10	10 ms	10 ms	1 ms
20	20 ms	40 ms	1 s
50	50 ms	250 ms	35 anni

Soluzioni *troppo* lente

Confrontiamo tre algoritmi con il seguente numero di istruzioni: $1000N$, $100N^2$, 2^N. Qual è il più veloce*?

* Assumendo 10^6 operazioni al secondo.

N	t($1000N$)	t($100N^2$)	t(2^N)
10	10 ms	10 ms	1 ms
20	20 ms	40 ms	1 s
50	50 ms	250 ms	35 anni
100	100 ms	1 s	$\gg$

$\gg$ = maggiore dell'età dell'universo

Complessità computazionale

Notazione asintotica $\mathcal{O}$

Si dice che una funzione $f(n)$ è asintotica a $\mathcal{O}(g(n))$ se da un certo punto in poi $f(n) < c \cdot g(n)$.

Complessità computazionale

Notazione asintotica $\mathcal{O}$

Si dice che una funzione $f(n)$ è asintotica a $\mathcal{O}(g(n))$ se da un certo punto in poi $f(n) < c \cdot g(n)$.

Esempi

- $4n + 5 = \mathcal{O}(n)$
- $n^2 + 100n = \mathcal{O}(n^2)$
- $2^n + 3n^4 = \mathcal{O}(2^n)$

Complessità computazionale

Notazione asintotica $\mathcal{O}$

Si dice che una funzione $f(n)$ è asintotica a $\mathcal{O}(g(n))$ se da un certo punto in poi $f(n) < c \cdot g(n)$.

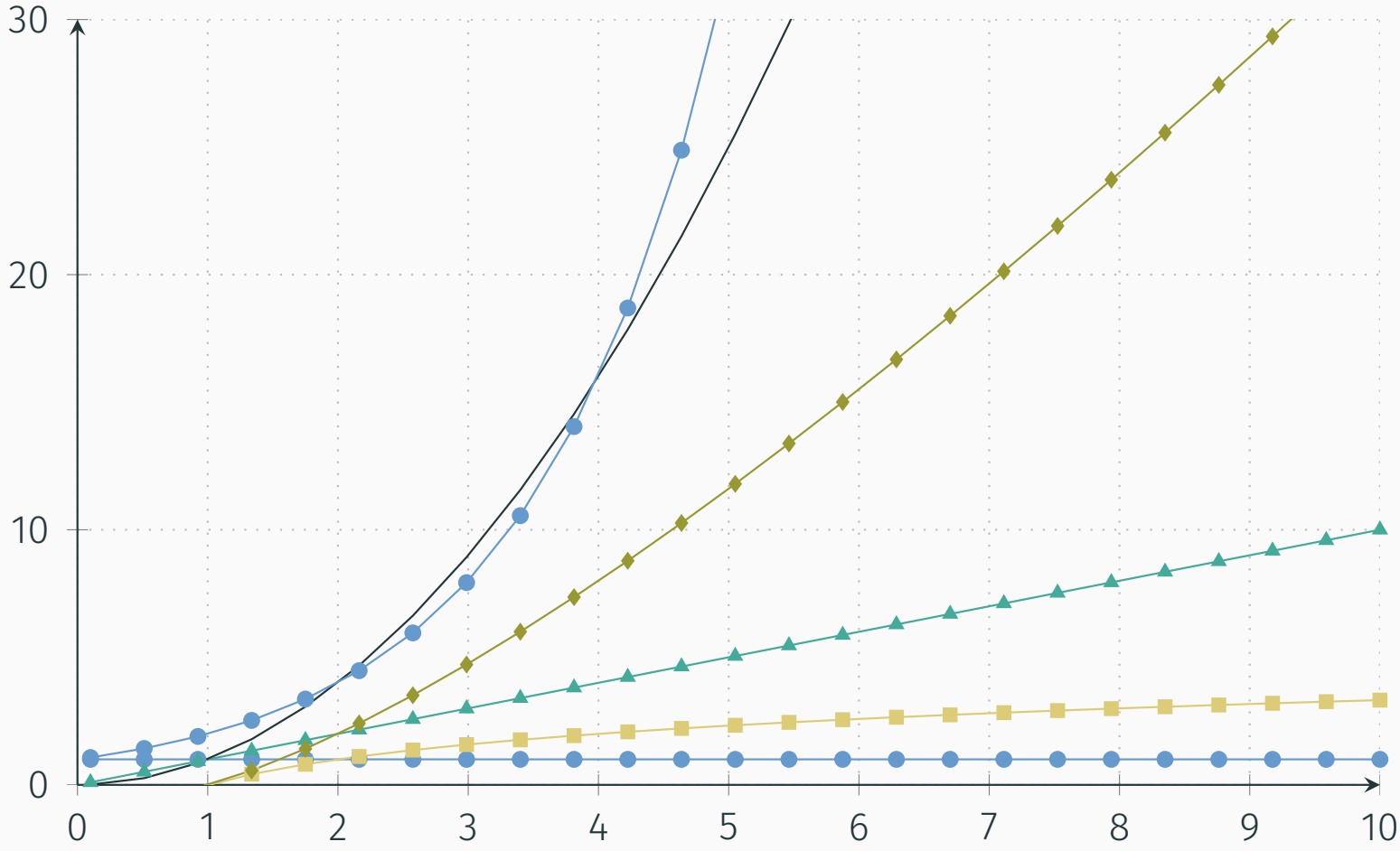
Esempi

- $4n + 5 = \mathcal{O}(n)$
- $n^2 + 100n = \mathcal{O}(n^2)$
- $2^n + 3n^4 = \mathcal{O}(2^n)$

Complessità computazionale

La stima asintotica del tempo di esecuzione di un algoritmo è detta "complessità computazionale".

Gerarchia degli infiniti



$$1 \ll \log n \ll n \ll n \log n \ll n^2 \ll 2^n \ll n!$$

Complessità computazionale della soluzione naïve

Che complessità computazionale ha la soluzione naïve?

Implementazione della soluzione naïve

```
...
for (int i = 0; i < N; i++)
    distanza[i] = sqrt(...);
...
for (int i = 0; i < Q; i++)
    cout << query(D[i]) << endl;
...
```

Complessità computazionale della soluzione naïve

Che complessità computazionale ha la soluzione naïve?

Implementazione della soluzione naïve

```
...
for (int i = 0; i < N; i++)
    distanza[i] = sqrt(...);
...
for (int i = 0; i < Q; i++)
    cout << query(D[i]) << endl;
...
```

La soluzione progettata per risolvere il problema impiega $\mathcal{O}(N)$ per calcolare il vettore `distanza` e $\mathcal{O}(N)$ per ogni query.

L'algoritmo impiega quindi $N + Q \cdot N = \mathcal{O}(Q \cdot N)$.

Tempo della soluzione naïve

Abbiamo determinato che l'algoritmo ha complessità $\mathcal{O}(Q \cdot N)$.

Consideriamo vari scenari con N stelle e Q query.

N	Q	$N \cdot Q$	tempo*
10	100	1000	1 ms
1000	1000	1 000 000	1 s
100 000	100 000	10^{10}	2,7 h

* Assumendo 10^6 operazioni al secondo.

Ordinamento

Dato un vettore *V* contenente *N* valori, è possibile ordinare *secondo un certo criterio* i valori.

5	2	3	8	1

Un ordinamento classico, dato un vettore di numeri, è quello crescente.

1	2	3	5	8

Algoritmi di ordinamento

Esistono numerosissimi algoritmi di ordinamento. In linea di massima, esistono algoritmi *lenti* e algoritmi *veloci*: ovviamente a noi interessano questi ultimi!

Esempio

- BUBBLE-SORT ordina con complessità $\mathcal{O}(N^2)$
- QUICK-SORT ordina con complessità $\mathcal{O}(N \log N)$

Algoritmi di ordinamento

Esistono numerosissimi algoritmi di ordinamento. In linea di massima, esistono algoritmi *lenti* e algoritmi *veloci*: ovviamente a noi interessano questi ultimi!

Esempio

- Bubble-Sort ordina con complessità $\mathcal{O}(N^2)$
- Quick-Sort ordina con complessità $\mathcal{O}(N \log N)$

STL

In C++ esiste una funzione, messa a disposizione dalle librerie standard, che esegue l'ordinamento in $\mathcal{O}(N \log N)$. Non è quindi necessario saperlo scrivere, ma solo sapere che esiste!

Ordinamento in C++

```cpp
#include <algorithm>

using namespace std;

int main()
{
    int vettore[10];
    for (int i = 0; i < 10; i++)
        cin >> vettore[i];

    sort(vettore, vettore + 10);
}
```

Ordinamento personalizzato in C++

```cpp
#include <algorithm>

using namespace std;

bool confronta(int a, int b) {
    return (a > b);
}

int main()
{
    int vettore[10];
    for (int i = 0; i < 10; i++)
        cin >> vettore[i];

    sort(vettore, vettore + 10, confronta);
}
```

Ricerca di un elemento in un vettore

Scrivere un algoritmo che cerchi un valore K all'interno di un vettore V contenente N interi.

Esempio

5	2	3	8	1	7	4	6

$K = 7$

Ricerca di un elemento in un vettore

Scrivere un algoritmo che cerchi un valore *K* all'interno di un vettore *V* contenente *N* interi.

Esempio

K = 7

```
int pos = 0;
while (pos < N && V[pos] != K)
    pos++;
```

Ricerca di un elemento in un vettore

Qual è la complessità computazionale dell'algoritmo?

Algoritmo di ricerca

```
int pos = 0;
while (pos < N && V[pos] != K)
    pos++;
```

Dopo quanti passi, in media, mi aspetto di trovare l'elemento? Qual è il caso peggiore?

Ricerca di un elemento in un vettore

Qual è la complessità computazionale dell'algoritmo?

Algoritmo di ricerca

```
int pos = 0;
while (pos < N && V[pos] != K)
    pos++;
```

Dopo quanti passi, in media, mi aspetto di trovare l'elemento? Qual è il caso peggiore? $\mathcal{O}(N)$.

Ricerca di un elemento in un vettore ordinato

Scrivere un algoritmo che cerchi un valore K all'interno di un vettore V contenente N interi ordinati.

Esempio

1	2	3	4	5	6	7	8	9

$K = 3$

Ricerca di un elemento in un vettore ordinato

Scrivere un algoritmo che cerchi un valore K all'interno di un vettore V contenente N interi ordinati.

Esempio

1	2	3	4	5	6	7	8	9

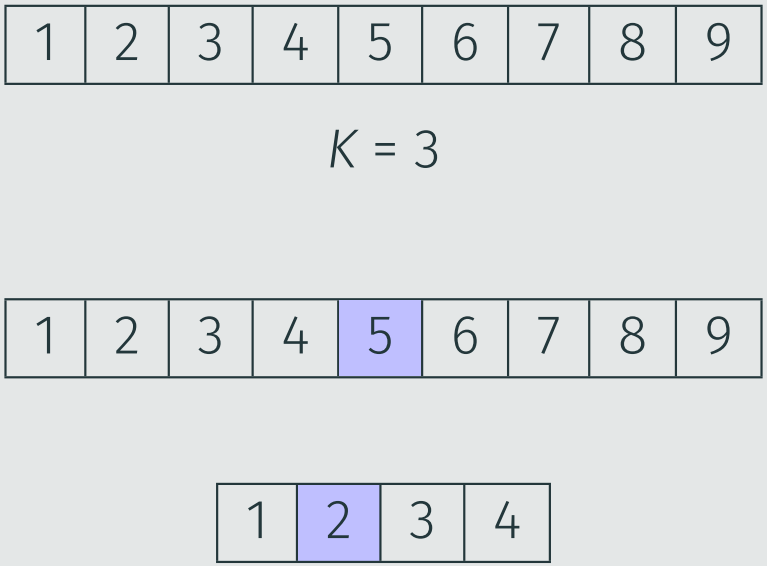
$K = 3$

1	2	3	4	5	6	7	8	9

Ricerca di un elemento in un vettore ordinato

Scrivere un algoritmo che cerchi un valore K all'interno di un vettore V contenente N interi ordinati.

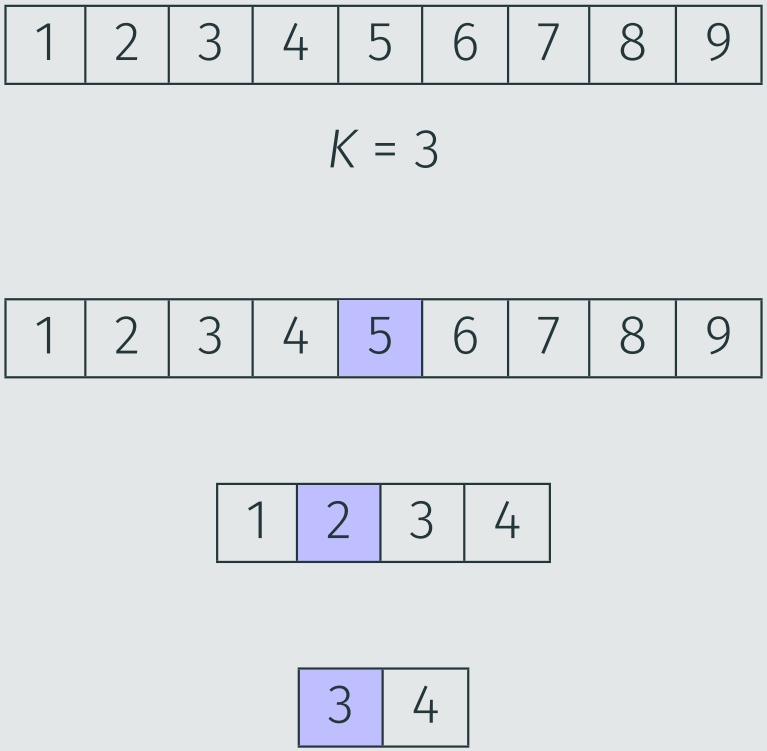
Esempio



Ricerca di un elemento in un vettore ordinato

Scrivere un algoritmo che cerchi un valore K all'interno di un vettore V contenente N interi ordinati.

Esempio



Ricerca binaria in C++

```
int ricerca_binaria(int K) {
    int i = 0, f = N - 1, m;
    while (i <= f)
    {
        m = (i + f) / 2;
        if (V[m] == K)
            return m;
        if (V[m] < K)
            i = m + 1;
        else
            f = m - 1;
    }
    return -1;
}
```

Ricerca binaria in C++

```
int ricerca_binaria(int K) {
    int i = 0, f = N - 1, m;
    while (i <= f)
    {
        m = (i + f) / 2;
        if (V[m] == K)
            return m;
        if (V[m] < K)
            i = m + 1;
        else
            f = m - 1;
    }
    return -1;
}
```

Complessità computazionale: $\mathcal{O}(\log_2 N)$.

Ricerca binaria in C++ con STL

```cpp
#include <algorithm>

int vettore[10];

bool trovato = binary_search(vettore, vettore + 10, 3);
```

Ricerca binaria in C++ con STL

```cpp
#include <algorithm>

int vettore[10];

bool trovato = binary_search(vettore, vettore + 10, 3);
```

oppure...

```cpp
int pos = lower_bound(vettore, vettore + 10, 7) - vettore;
if (pos != 10)
    cout << "Elemento 7 trovato in posizione " << pos;
else
    cout << "Elemento 7 non trovato";
```

Soluzione ottima

“Anno luce” - Soluzione ottima

0	$2\sqrt{3}$	$\sqrt{3}$	1	3

Il quesito era:

Query

Data una query (ovvero il numero di anni luce per cui è concesso viaggiare), qual è la risposta?

“Anno luce” - Soluzione ottima

0	$2\sqrt{3}$	$\sqrt{3}$	1	3

Il quesito era:

Query

Data una query (ovvero il numero di anni luce per cui è concesso viaggiare), qual è la risposta?

E se decidessimo di ordinare il vettore `distanza`?

0	1	$\sqrt{3}$	3	$2\sqrt{3}$

“Anno luce” - Soluzione ottima

0	1	$\sqrt{3}$	3	$2\sqrt{3}$

Esempio

Quante stelle sono raggiungibili in 1 anno luce?

“Anno luce” - Soluzione ottima

0	1	$\sqrt{3}$	3	$2\sqrt{3}$

Esempio

Quante stelle sono raggiungibili in 1 anno luce?

0	1	$\sqrt{3}$	3	$2\sqrt{3}$

“Anno luce” - Soluzione ottima

0	1	$\sqrt{3}$	3	$2\sqrt{3}$

Esempio

Quante stelle sono raggiungibili in 1 anno luce?

0	1	$\sqrt{3}$	3	$2\sqrt{3}$

Quante stelle sono raggiungibili in 2 anni luce?

“Anno luce” - Soluzione ottima

0	1	$\sqrt{3}$	3	$2\sqrt{3}$

Esempio

Quante stelle sono raggiungibili in 1 anno luce?

0	1	$\sqrt{3}$	3	$2\sqrt{3}$

Quante stelle sono raggiungibili in 2 anni luce?

0	1	$\sqrt{3}$	3	$2\sqrt{3}$

“Anno luce” - Soluzione ottima

Il nostro algoritmo prevede quindi un ordinamento iniziale del vettore `distanza` (che è lungo esattamente N).

Risponde poi alle Q query eseguendo per ciascuna una ricerca binaria del primo valore nel vettore maggiore della distanza che ci è concesso viaggiare.

“Anno luce” - Soluzione ottima

Il nostro algoritmo prevede quindi un ordinamento iniziale del vettore `distanza` (che è lungo esattamente N).

Risponde poi alle Q query eseguendo per ciascuna una ricerca binaria del primo valore nel vettore maggiore della distanza che ci è concesso viaggiare.

Cerchiamo di capire se è migliore di quello precedente. Sappiamo che $N \leq 100\,000$ e $Q \leq 100\,000$, quindi possiamo considerare per semplicità $Q = N$.

“Anno luce” - Soluzione ottima

Il nostro algoritmo prevede quindi un ordinamento iniziale del vettore `distanza` (che è lungo esattamente N).

Risponde poi alle Q query eseguendo per ciascuna una ricerca binaria del primo valore nel vettore maggiore della distanza che ci è concesso viaggiare.

Cerchiamo di capire se è migliore di quello precedente. Sappiamo che $N \leq 100\,000$ e $Q \leq 100\,000$, quindi possiamo considerare per semplicità $Q = N$.

Soluzione	Fase iniziale	Risposta alle query	Somma

“Anno luce” - Soluzione ottima

Il nostro algoritmo prevede quindi un ordinamento iniziale del vettore `distanza` (che è lungo esattamente N).

Risponde poi alle Q query eseguendo per ciascuna una ricerca binaria del primo valore nel vettore maggiore della distanza che ci è concesso viaggiare.

Cerchiamo di capire se è migliore di quello precedente. Sappiamo che $N \leq 100\,000$ e $Q \leq 100\,000$, quindi possiamo considerare per semplicità $Q = N$.

Soluzione	Fase iniziale	Risposta alle query	Somma
Naïve	$\mathcal{O}(N)$	$\mathcal{O}(Q \cdot N)$	$\mathcal{O}(N^2)$

“Anno luce” - Soluzione ottima

Il nostro algoritmo prevede quindi un ordinamento iniziale del vettore `distanza` (che è lungo esattamente N).

Risponde poi alle Q query eseguendo per ciascuna una ricerca binaria del primo valore nel vettore maggiore della distanza che ci è concesso viaggiare.

Cerchiamo di capire se è migliore di quello precedente. Sappiamo che $N \leq 100\,000$ e $Q \leq 100\,000$, quindi possiamo considerare per semplicità $Q = N$.

Soluzione	Fase iniziale	Risposta alle query	Somma
Naïve	$\mathcal{O}(N)$	$\mathcal{O}(Q \cdot N)$	$\mathcal{O}(N^2)$
Ottima	$\mathcal{O}(N \log N)$	$\mathcal{O}(Q \cdot \log N)$	$\mathcal{O}(N \log N)$

“Anno luce” - Soluzione ottima

Il nostro algoritmo prevede quindi un ordinamento iniziale del vettore `distanza` (che è lungo esattamente N).

Risponde poi alle Q query eseguendo per ciascuna una ricerca binaria del primo valore nel vettore maggiore della distanza che ci è concesso viaggiare.

Cerchiamo di capire se è migliore di quello precedente. Sappiamo che $N \leq 100\,000$ e $Q \leq 100\,000$, quindi possiamo considerare per semplicità $Q = N$.

Soluzione	Fase iniziale	Risposta alle query	Somma
Naïve	$\mathcal{O}(N)$	$\mathcal{O}(Q \cdot N)$	$\mathcal{O}(N^2)$
Ottima	$\mathcal{O}(N \log N)$	$\mathcal{O}(Q \cdot \log N)$	$\mathcal{O}(N \log N)$
Ottima*	$\mathcal{O}(N^2)$	$\mathcal{O}(Q \cdot \log N)$	$\mathcal{O}(N^2)$

* con ordinamento inefficiente

Tocca a voi!

Esercizio

Implementare la soluzione ottima al problema “annoluce”.

Riferimenti

Questa presentazione e soluzione in C++ dell’esercizio:

`https://files.chiodini.org/OII_Territoriali_2019/`

- Piattaforma di allenamento con correttore e forum:
  `https://training.olinfo.it`
- Guida alle selezioni territoriali del prof. Bugatti:
  `http://www.imparando.net/sito/olimpiadi_di_informatica.htm`