

Segment Tree

Problemi

Lorenzo Ferrari, Davide Bartoli

February 15, 2023

Table of contents

Segment Tree

Maximum Subarray Sum

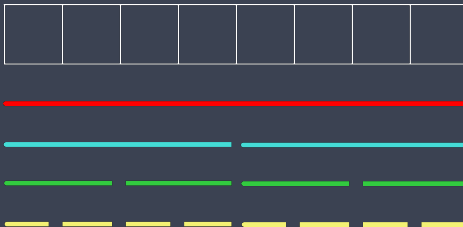
lower_bound query

Problemi

Segment Tree

Idea

Per rispondere efficientemente a query¹ su un range $[l, r]$ di valori, ci salviamo la risposta per alcuni intervalli la cui lunghezza è una potenza di 2.

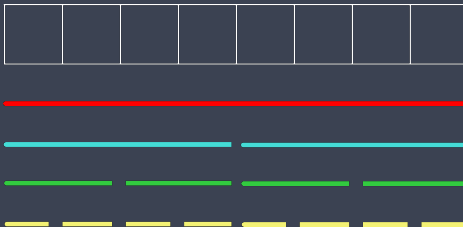


¹siano esse somma, minimo, o operazioni più complesse

Segment Tree

Idea

Per rispondere efficientemente a query¹ su un range $[l, r]$ di valori, ci salviamo la risposta per alcuni intervalli la cui lunghezza è una potenza di 2.



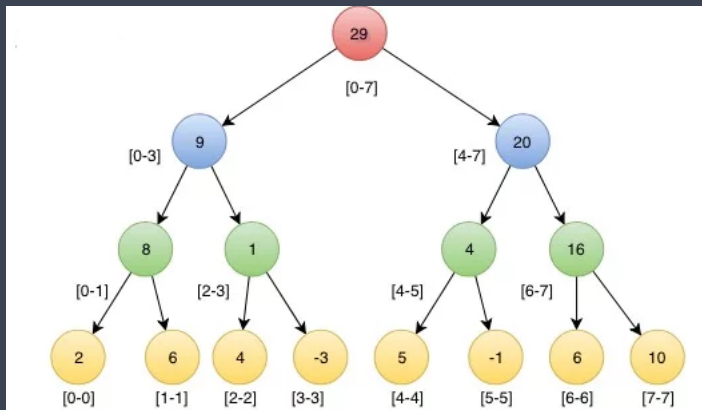
Gli intervalli sono in totale $N + N/2 + N/4 + \dots \leq 2N$

¹siano esse somma, minimo, o operazioni più complesse

Segment Tree

Idea

L'insieme di intervalli si può vedere come un albero binario, che rende più intuitiva l'implementazione.



Segment Tree

caratteristiche

Nominiamo i nodi a partire da 1 per livelli: la radice ha indice 1, il secondo livello contiene i nodi 2 e 3, il terzo livello i nodi 4, 5, 6 e 7 ...

Segment Tree

caratteristiche

Nominiamo i nodi a partire da 1 per livelli: la radice ha indice 1, il secondo livello contiene i nodi 2 e 3, il terzo livello i nodi 4, 5, 6 e 7 ...

L'albero binario così costruito ha le seguenti caratteristiche:

- ▶ la radice ha indice 1
- ▶ il figlio sinistro di un nodo i ha indice $2i$
- ▶ il figlio destro di un nodo i ha indice $2i + 1$
- ▶ il padre di un nodo i ha indice $i/2$
- ▶ i nodi sono numerati da 1 a $2N - 1$
- ▶ l'albero ha altezza $O(\log N)$



```
struct Segment {  
    int n;  
    vector<long long> t;  
    Segment(int _n, vector<int> a) {  
        for (n = 1; n < _n; n <= 1);  
        t.resize(2 * n);  
        for (int i = 0; i < _n; ++i) {  
            t[i + n] = a[i];  
        }  
        for (int i = n-1; i > 0; --i) {  
            t[i] = t[2*i] + t[2*i+1];  
        }  
    }  
};
```


Segment Tree

Update

Per gli **update**, notiamo che ogni nodo è contenuto in esattamente $\log N$ intervalli, possiamo quindi aggiornarli tutti in $O(\log N)$.



```
void update(int i, int tl, int tr, int p, int v) {  
    if (p < tl || tr < p) return;  
    if (tl == tr) {  
        t[i] = v;  
    } else {  
        int tm = (tl + tr) / 2;  
        update(2*i, tl, tm, p, v);  
        update(2*i+1, tm+1, tr, p, v);  
        t[i] = t[2*i] + t[2*i+1];  
    }  
}  
  
void update(int p, int v) {  
    update(1, 0, n-1, p, v);  
}
```

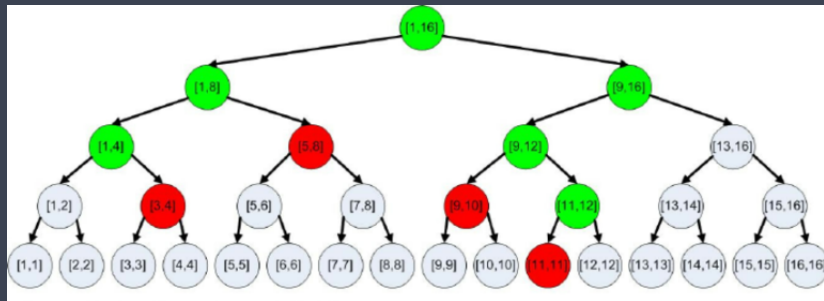


```
void update(int p, int v) {  
    for (t[p += n] = v; p > 1; p >= 1) {  
        t[p >> 1] = t[p] + t[p ^ 1];  
    }  
}
```

Segment Tree

Query

Per le **query** dobbiamo trovare un insieme di intervalli da unire per ottenere la risposta desiderata.



Segment Tree

Query

Per rispondere a una generica query $[l, r]$, possiamo utilizzare una dfs sull'albero. Ogni volta che raggiungiamo un nodo abbiamo 3 possibilità:

Segment Tree

Query

Per rispondere a una generica query $[l, r]$, possiamo utilizzare una dfs sull'albero. Ogni volta che raggiungiamo un nodo abbiamo 3 possibilità:

- ▶ l'intervallo è completamente contenuto in $[l, r]$, quindi possiamo aggiungerlo alla risposta e fermarci
- ▶ l'intervallo è completamente fuori da $[l, r]$, quindi possiamo ignorarlo e fermarci
- ▶ l'intervallo è parzialmente contenuto in $[l, r]$, quindi ricorriamo nei figli

Segment Tree

Query

Per rispondere a una generica query $[l, r]$, possiamo utilizzare una dfs sull'albero. Ogni volta che raggiungiamo un nodo abbiamo 3 possibilità:

- ▶ l'intervallo è completamente contenuto in $[l, r]$, quindi possiamo aggiungerlo alla risposta e fermarci
- ▶ l'intervallo è completamente fuori da $[l, r]$, quindi possiamo ignorarlo e fermarci
- ▶ l'intervallo è parzialmente contenuto in $[l, r]$, quindi ricorriamo nei figli

Si può dimostrare che questo processo visita $O(\log N)$ nodi.

Segment Tree

Query

Per rispondere a una generica query $[l, r]$, possiamo utilizzare una dfs sull'albero. Ogni volta che raggiungiamo un nodo abbiamo 3 possibilità:

- ▶ l'intervallo è completamente contenuto in $[l, r]$, quindi possiamo aggiungerlo alla risposta e fermarci
- ▶ l'intervallo è completamente fuori da $[l, r]$, quindi possiamo ignorarlo e fermarci
- ▶ l'intervallo è parzialmente contenuto in $[l, r]$, quindi ricorriamo nei figli

Si può dimostrare che questo processo visita $O(\log N)$ nodi.

Questa struttura dati impiega solo $O(\log N)$ per query/update.



```
long long query(int i, int tl, int tr, int l, int r) {  
    if (r < tl || tr < l) return 0;  
    if (l <= tl && tr <= r) return t[i];  
    else {  
        int tm = (tl + tr) / 2;  
        return query(2*i, tl, tm, l, r) +  
               query(2*i+1, tm+1, tr, l, r);  
    }  
}  
  
long long query(int l, int r) {  
    return query(1, 0, n-1, l, r);  
}
```

Il segment nell'implementazione fa update e calcola la somma su range in $O(\log N)$, ma i segment tree sono strutture dati molto generiche e versatili, che si possono utilizzare per risolvere molti altri problemi.

Il segment nell'implementazione fa update e calcola la somma su range in $O(\log N)$, ma i segment tree sono strutture dati molto generiche e versatili, che si possono utilizzare per risolvere molti altri problemi.

In generale ...

Se si salvano le informazioni su un intervallo in una struct nodo e le informazioni del padre si possono facilmente ottenere combinando le informazioni dei figli, allora si può usare un segment tree.

Maximum Subarray Sum

Problema

Maximum Subarray Sum

Dato un array di N numeri, rispondi alle seguenti query:

- ▶ modifica il valore di un elemento
- ▶ calcola la somma massima di un sottoarray di un intervallo $[l, r]$

<https://training.olinfo.it/#/task/rangetree3/statement>

Maximum Subarray Sum

Problema

Maximum Subarray Sum

Dato un array di N numeri, rispondi alle seguenti query:

- modifica il valore di un elemento
- calcola la somma massima di un sottoarray di un intervallo $[l, r]$

<https://training.olinfo.it/#/task/rangetree3/statement>

Cerchiamo di capire come utilizzare un segment tree per risolvere questo problema.

Maximum Subarray Sum

idea

Come avevamo visto la scorsa volta, per poter utilizzare il segment tree è necessario riuscire a unire le informazioni di 2 nodi velocemente.

Maximum Subarray Sum

idea

Come avevamo visto la scorsa volta, per poter utilizzare il segment tree è necessario riuscire a unire le informazioni di 2 nodi velocemente.

Quali informazioni dobbiamo salvarci per risolvere questo problema? Come facciamo a unire le informazioni di 2 nodi?

Maximum Subarray Sum

idea

Come avevamo visto la scorsa volta, per poter utilizzare il segment tree è necessario riuscire a unire le informazioni di 2 nodi velocemente.

Quali informazioni dobbiamo salvarci per risolvere questo problema? Come facciamo a unire le informazioni di 2 nodi? Sicuramente una delle informazioni è la somma massima di un sottoarray, ovvero quello che ci chiede il problema.

Maximum Subarray Sum

idea

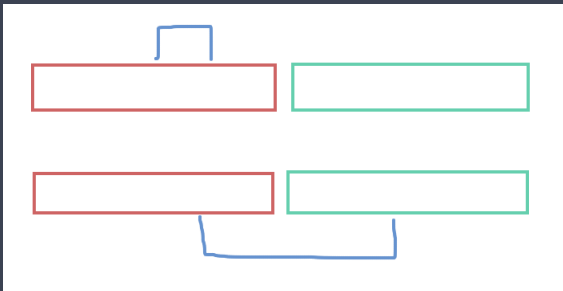
Come avevamo visto la scorsa volta, per poter utilizzare il segment tree è necessario riuscire a unire le informazioni di 2 nodi velocemente.

Quali informazioni dobbiamo salvarci per risolvere questo problema? Come facciamo a unire le informazioni di 2 nodi? Sicuramente una delle informazioni è la somma massima di un sottoarray, ovvero quello che ci chiede il problema.

Come facciamo a unire 2 nodi però?

Maximum Subarray Sum

idea

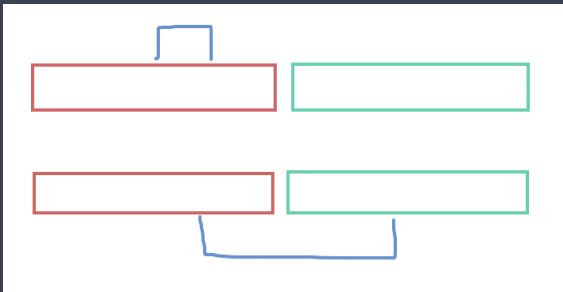


Ci sono 2 casi:

- ▶ l'intervallo massimo è completamente contenuto in uno dei 2 nodi, quindi conosciamo già il suo valore
- ▶ l'intervallo è a metà tra i 2 nodi

Maximum Subarray Sum

idea



Ci sono 2 casi:

- l'intervallo massimo è completamente contenuto in uno dei 2 nodi, quindi conosciamo già il suo valore
- l'intervallo è a metà tra i 2 nodi

Nel secondo caso non abbiamo modo di calcolare il valore che ci interessa: dobbiamo tenerci altre informazioni nei nodi.

Maximum Subarray Sum

idea

Notiamo che in questo caso il subarray massimo è formato da un suffisso del primo nodo e un prefisso del secondo nodo.

Maximum Subarray Sum

idea

Notiamo che in questo caso il subarray massimo è formato da un suffisso del primo nodo e un prefisso del secondo nodo.

Possiamo salvarci quindi anche il prefisso e il suffisso massimi di ogni nodo.

Maximum Subarray Sum

idea

Notiamo che in questo caso il subarray massimo è formato da un suffisso del primo nodo e un prefisso del secondo nodo.

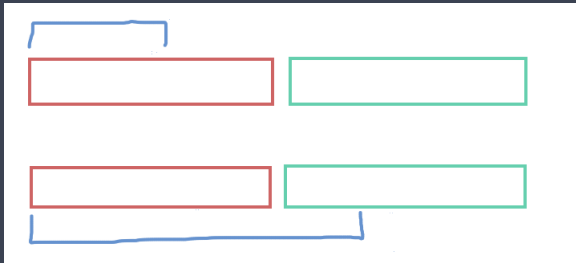
Possiamo salvarci quindi anche il prefisso e il suffisso massimi di ogni nodo.

Riusciamo però a unire 2 nodi e calcolare il suffisso massimo e il prefisso massimo del nodo padre? Al momento no.

Maximum Subarray Sum

idea

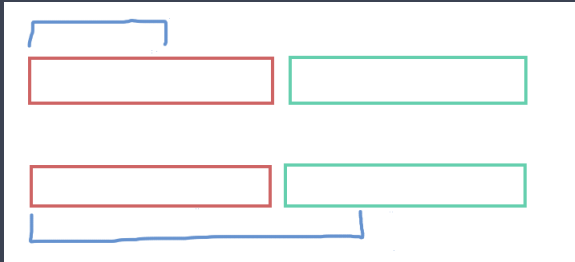
Consideriamo solo il prefisso per ora.



Maximum Subarray Sum

idea


Consideriamo solo il prefisso per ora.



Anche in questo caso abbiamo 2 casi, e notiamo che ci basta salvare la somma totale del nodo per avere finalmente tutte le informazioni necessarie.

Maximum Subarray Sum

implementazione



```
struct nodo {
    int somma, pref_max, suff_max, subarray_max;
};

nodo unisci(nodo a, nodo b) {
    nodo res;
    res.somma = a.somma + b.somma;
    res.pref_max = max(a.pref_max, a.somma + b.pref_max);
    res.suff_max = max(b.suff_max, b.somma + a.suff_max);
    int caso1 = max(a.subarray_max, b.subarray_max);
    int caso2 = a.suff_max + b.pref_max;
    res.subarray_max = max(caso1, caso2);
    return res;
}
```

Conta elementi

Problema

Conta elementi

Dato un array di N numeri, rispondi alla seguente query:

- ▶ quanti sono gli elementi $\geq k$ nell'intervallo $[l, r]$

Conta elementi

Problema

Conta elementi

Dato un array di N numeri, rispondi alla seguente query:

- ▶ quanti sono gli elementi $\geq k$ nell'intervallo $[l, r]$

In questo problema non abbiamo update, ma un segment tree può comunque farci molto comodo per risolvere il problema.

Conta elementi

Problema

Conta elementi

Dato un array di N numeri, rispondi alla seguente query:

- ▶ quanti sono gli elementi $\geq k$ nell'intervallo $[l, r]$

In questo problema non abbiamo update, ma un segment tree può comunque farci molto comodo per risolvere il problema.

In particolare se sappiamo calcolare la risposta per un nodo velocemente, calcolare la risposta in $[l, r]$ è semplice: ci basta sommare le risposte dei nodi che formano $[l, r]$.

Conta elementi

idea

Come facciamo a calcolare la risposta per un nodo?

Conta elementi

idea

Come facciamo a calcolare la risposta per un nodo?
Possiamo salvare in ogni nodo una lista ordinata dei valori che contengono, e calcolare la risposta facendo una ricerca binaria.

Conta elementi

idea

Come facciamo a calcolare la risposta per un nodo?

Possiamo salvare in ogni nodo una lista ordinata dei valori che contengono, e calcolare la risposta facendo una ricerca binaria.

Questo è possibile dato che non dobbiamo fare update, quindi una volta inizializzato il segment tree non viene più modificato.

Conta elementi

idea

Come facciamo a calcolare la risposta per un nodo?

Possiamo salvare in ogni nodo una lista ordinata dei valori che contengono, e calcolare la risposta facendo una ricerca binaria.

Questo è possibile dato che non dobbiamo fare update, quindi una volta inizializzato il segment tree non viene più modificato.

Ma quanta memoria stiamo usando in questo caso? Avevamo visto che ogni nodo è compreso in esattamente $\log N$ intervalli, quindi la memoria totale è $O(N \log N)$, e non abbiamo problemi.

lower_bound query

Alcune query possono chiedere di trovare il primo (o l'ultimo) elemento $\geq x$ in un range $[l, r]$.

lower_bound query

Alcune query possono chiedere di trovare il primo (o l'ultimo) elemento $\geq x$ in un range $[l, r]$.

Un'opzione è fare una binary search con $\log(r - l)$ query di minimo, per un costo totale di $O(\log^2 n)$

lower_bound query

Alcune query possono chiedere di trovare il primo (o l'ultimo) elemento $\geq x$ in un range $[l, r]$.

Un'opzione è fare una binary search con $\log(r - l)$ query di minimo, per un costo totale di $O(\log^2 n)$

Il problema può però essere risolto in $O(\log n)$. Come succede spesso con i segment tree, risolviamo il problema ricorsivamente e tronchiamo la ricerca quando nell'intervallo in cui siamo non ci sono soluzioni (l'elemento massimo è $< x$).

Problemi

<https://cses.fi/problemset/task/1650>

<https://cses.fi/problemset/task/2206>

<https://training.olinfo.it/#/task/muraglia/statement>

<https://training.olinfo.it/#/task/rangetree3/statement>

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

<https://training.olinfo.it/#/task/pre-egoi-parkour/statement>