

Tecniche avanzate su alberi

Linearizzazione e Small to Large

Lorenzo Ferrari, Davide Bartoli

March 1, 2023

Table of contents

Linearizzazione di un albero

Problemi

Problema motivazionale

Subtree queries

Subtree queries

È dato un albero radicato in 1 con $N \leq 100'000$ nodi. Ogni nodo ha un valore. Performa Q di queste operazioni:

1. cambia il valore del nodo s a x
2. calcola la somma dei valori nel subtree del nodo s

<https://cses.fi/problemset/task/1137>

Come si risolve un problema del genere?

- un'idea è fare update in $O(1)$ e query in $O(N)$ con una dfs

Problema motivazionale

Subtree queries

Subtree queries

È dato un albero radicato in 1 con $N \leq 100'000$ nodi. Ogni nodo ha un valore. Performa Q di queste operazioni:

1. cambia il valore del nodo s a x
2. calcola la somma dei valori nel subtree del nodo s

<https://cses.fi/problemset/task/1137>

Come si risolve un problema del genere?

- un'idea è fare update in $O(1)$ e query in $O(N)$ con una dfs
- un'alternativa è fare update in $O(N)$ e query in $O(1)$

Problema motivazionale

Subtree queries

Subtree queries

È dato un albero radicato in 1 con $N \leq 100'000$ nodi. Ogni nodo ha un valore. Performa Q di queste operazioni:

1. cambia il valore del nodo s a x
2. calcola la somma dei valori nel subtree del nodo s

<https://cses.fi/problemset/task/1137>

Come si risolve un problema del genere?

- ▶ un'idea è fare update in $O(1)$ e query in $O(N)$ con una dfs
- ▶ un'alternativa è fare update in $O(N)$ e query in $O(1)$
- ▶ la complessità di entrambe le soluzioni è $O(NQ)$

Problema motivazionale

Subtree queries

Subtree queries

È dato un albero radicato in 1 con $N \leq 100'000$ nodi. Ogni nodo ha un valore. Performa Q di queste operazioni:

1. cambia il valore del nodo s a x
2. calcola la somma dei valori nel subtree del nodo s

<https://cses.fi/problemset/task/1137>

Come si risolve un problema del genere?

- un'idea è fare update in $O(1)$ e query in $O(N)$ con una dfs
- un'alternativa è fare update in $O(N)$ e query in $O(1)$
- la complessità di entrambe le soluzioni è $O(NQ)$

Si può fare meglio di così?

Ordine DFS

Facciamo una **DFS** sull'albero, per ogni nodo salviamo:

- ▶ il tempo $tin[v]$ in cui la dfs entra nel nodo v
- ▶ il tempo $tout[v]$ in cui la dfs esce dal nodo v

Ordine DFS

Facciamo una **DFS** sull'albero, per ogni nodo salviamo:

- ▶ il tempo $tin[v]$ in cui la dfs entra nel nodo v
- ▶ il tempo $tout[v]$ in cui la dfs esce dal nodo v

Per come funziona la dfs, per ogni nodo u nel sottoalbero di v vale

$$tin[v] < tin[u] < tout[u] < tout[v]$$

subtree = subarray

Nell'**ordine DFS**, i nodi in un subtree formano un subarray!

Ordine DFS

Facciamo una **DFS** sull'albero, per ogni nodo salviamo:

- ▶ il tempo $tin[v]$ in cui la dfs entra nel nodo v
- ▶ il tempo $tout[v]$ in cui la dfs esce dal nodo v

Per come funziona la dfs, per ogni nodo u nel sottoalbero di v vale

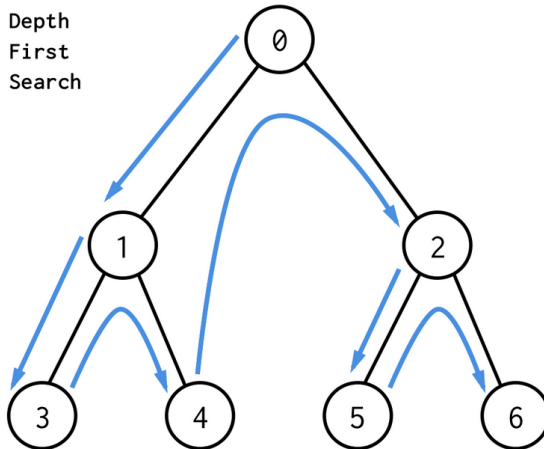
$$tin[v] < tin[u] < tout[u] < tout[v]$$

subtree = subarray

Nell'**ordine DFS**, i nodi in un subtree formano un subarray!

Possiamo lavorare sui subtree come se lavorassimo su subarray dell'ordine dfs e usare le strutture dati che conosciamo per processare query e update efficientemente.

Ordine DFS



Altre applicazioni

Ancestors query

range update/point query

- ▶ su un segment, potevamo passare facilmente da point update/range query a range update/point query
- ▶ la stessa idea si può usare su un albero linearizzato

Altre applicazioni

Ancestors query

range update/point query

- ▶ su un segment, potevamo passare facilmente da point update/range query a range update/point query
- ▶ la stessa idea si può usare su un albero linearizzato

Path queries

È dato un albero radicato in 1 con $N \leq 100'000$ nodi. Ogni nodo ha un valore. Performa Q di queste operazioni:

1. cambia il valore del nodo s a x
2. calcola la somma dei valori sul percorso dal nodo s alla radice

<https://cses.fi/problemset/task/1137>

Problema motivazionale

Distinct Colors

Distinct Colors

Dato un albero radicato in 1 con $N \leq 200'000$ nodi. Ogni nodo ha un colore. Per ogni nodo vogliamo calcolare il numero di colori distinti nel suo sottoalbero.

<https://cses.fi/problemset/task/1139>

Potremmo pensare di linearizzare l'albero come abbiamo visto prima, in questo modo dobbiamo fare N query, e una query ci chiede quanti colori distinti ci sono in un intervallo.

Problema motivazionale

Distinct Colors

Distinct Colors

Dato un albero radicato in 1 con $N \leq 200'000$ nodi. Ogni nodo ha un colore. Per ogni nodo vogliamo calcolare il numero di colori distinti nel suo sottoalbero.

<https://cses.fi/problemset/task/1139>

Potremmo pensare di linearizzare l'albero come abbiamo visto prima, in questo modo dobbiamo fare N query, e una query ci chiede quanti colori distinti ci sono in un intervallo. Come facciamo a risolvere questo problema? Non è ovvio, ma possiamo risolverlo utilizzando un segment tree.

<https://cses.fi/problemset/task/1139>

Problema motivazionale

Distinct Colors

Questo problema è su un albero, possiamo quindi sfruttare questa proprietà per risolverlo in modo più semplice.

Problema motivazionale

Distinct Colors

Questo problema è su un albero, possiamo quindi sfruttare questa proprietà per risolverlo in modo più semplice.


Proviamo a risolvere il problema facendo una singola dfs sul nostro albero, per ora senza preoccuparci della complessità.

Problema motivazionale

Distinct Colors

Questo problema è su un albero, possiamo quindi sfruttare questa proprietà per risolverlo in modo più semplice.

Proviamo a risolvere il problema facendo una singola dfs sul nostro albero, per ora senza preoccuparci della complessità.



```
set<int> dfs(int pos, int par) {  
    set<int> res;  
    for (int x : adj[pos]) {  
        if (x == par) continue;  
        set<int> tmp = dfs(x, pos);  
        for (int y : tmp) res.insert(y);  
    }  
    ans[pos] = res.size();  
    res.insert(color[pos]);  
    return res;  
}
```

Problema motivazionale

Distinct Colors

Qual è la complessità di questa soluzione? $O(N^2)$, dato che unire due set ci richiede $O(N)$.

Problema motivazionale

Distinct Colors

Qual è la complessità di questa soluzione? $O(N^2)$, dato che unire due set ci richiede $O(N)$.

Possiamo modificare leggermente questa soluzione per renderla più veloce?

Problema motivazionale

Distinct Colors

Qual è la complessità di questa soluzione? $O(N^2)$, dato che unire due set ci richiede $O(N)$.

Possiamo modificare leggermente questa soluzione per renderla più veloce?

La risposta è sì, possiamo utilizzare la tecnica chiamata **small to large**. In particolare quando uniamo due set, possiamo mettere il set più piccolo dentro al set più grande.

Problema motivazionale

Distinct Colors

Qual è la complessità di questa soluzione? $O(N^2)$, dato che unire due set ci richiede $O(N)$.

Possiamo modificare leggermente questa soluzione per renderla più veloce?

La risposta è sì, possiamo utilizzare la tecnica chiamata **small to large**. In particolare quando uniamo due set, possiamo mettere il set più piccolo dentro al set più grande.

Ma è più veloce? Sì, infatti consideriamo un singolo elemento: ogni volta che viene spostato, si trova in un set grande almeno il doppio di quello in cui era prima.

Problema motivazionale

Distinct Colors

Qual è la complessità di questa soluzione? $O(N^2)$, dato che unire due set ci richiede $O(N)$.

Possiamo modificare leggermente questa soluzione per renderla più veloce?

La risposta è sì, possiamo utilizzare la tecnica chiamata **small to large**. In particolare quando uniamo due set, possiamo mettere il set più piccolo dentro al set più grande.


Ma è più veloce? Sì, infatti consideriamo un singolo elemento: ogni volta che viene spostato, si trova in un set grande almeno il doppio di quello in cui era prima.

Quindi viene spostato al massimo $O(\log N)$ volte, e la complessità totale è $O(N \log N)$.

Problema motivazionale

Distinct Colors

Dobbiamo però fare attenzione a come passiamo/ritorniamo il set dalle funzioni, in quanto non vogliamo che venga copiato (altrimenti la complessità cambia).



```
set<int> k[200005];
void dfs(int pos, int par) {
    for (int x : adj[pos]) {
        if (x == par) continue;
        dfs(x, pos);
        if (k[x].size() > k[pos].size()) swap(k[x], k[pos]);
        for (int y : k[x]) k[pos].insert(y);
    }
    k[pos].insert(color[pos]);
    ans[pos] = k[pos].size();
}
```

Problemi

(belli)

<https://cses.fi/problemset/task/1137>

<https://cses.fi/problemset/task/1138>

<https://cses.fi/problemset/task/1139>

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

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

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