Range tree

Luca Wehrstedt

1 Il problema

Supponiamo che ci venga data una sequenza di N numeri interi e che ci venga chiesto di eseguire Q operazioni di questo tipo:

1. **Update** - modificare un sottoinsieme di elementi senza restituire nulla
   ad esempio sostituirli o incrementarli di un certo valore
2. **Query** - analizzare un sottoinsieme di elementi e restituire un risultato
   ad esempio restituire la somma dei valori o il loro massimo

Supponiamo inoltre che tutti i sottoinsiemi siano sottosequenze contigue, cioè intervalli nella forma $[a, b)$ (che comprendono tutti gli elementi dall'a-esimo, incluso, al b-esimo, escluso).

1.1 Soluzione ingenua

La prima soluzione che viene in mente è di eseguire ogni operazione esattamente come è stata descritta, cioè di simulare alla lettera il processo che viene descritto nel problema:

- per l'update si vanno a modificare gli elementi uno alla volta
- per la query si vanno ad analizzare gli elementi uno alla volta

Una soluzione del genere ha complessità $O(N)$ per ogni singola operazione, e quindi ha in totale una complessità $O(Q \cdot N)$.

1.2 L'idea

L'idea cruciale per risolvere questo problema in modo efficiente è di affiancare agli elementi "reali" della sequenza altri elementi "finti", che chiameremo **hub**, che rappresentano una sottosequenza contigua di elementi "reali".

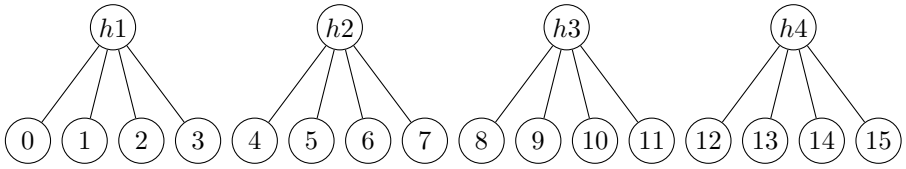
In questo modo, quando un update coinvolgerà tutti gli elementi rappresentati da un certo hub, andremo a modificare solo l'elemento hub e non gli elementi che esso rappresenta. Allo stesso modo, quando una query interesserà tutti gli elementi rappresentati da un certo hub, andremo ad analizzare solo l'elemento hub e non gli elementi che esso rappresenta.

Un modo per immaginarsi questo procedimento è il seguente: usiamo gli elementi hub come "promemoria" per ricordarci di un'operazione che dobbiamo andare a fare sugli elementi "reali". In quanto siamo pigri, non andiamo a toccare gli elementi "reali" fintanto che riusciamo a cavarcela con i soli elementi hub.

1.3 Soluzione migliorata

Un modo di sfruttare questa idea potrebbe essere di creare $\sqrt{N}$ elementi hub, ciascuno dei quali rappresenta un intervallo di elementi lungo proprio $\sqrt{N}$.

Con $N = 16$ una struttura dati del genere potrebbe essere rappresentata graficamente nel modo seguente (dove $0, \ldots, 15$ sono gli elementi della sequenza e $h1, \ldots, h4$ sono gli elementi hub).



Ora, per eseguire un'operazione su un intervallo $[a, b)$, andiamo a considerare innanzitutto gli elementi hub e, per l'elemento che rappresenta l'intervallo $[l, r)$, possono verificarsi tre casi:

- *nessun elemento rappresentato dall'elemento hub va considerato* `b <= l || r <= a`

  In tal caso ignoriamo l'elemento hub.

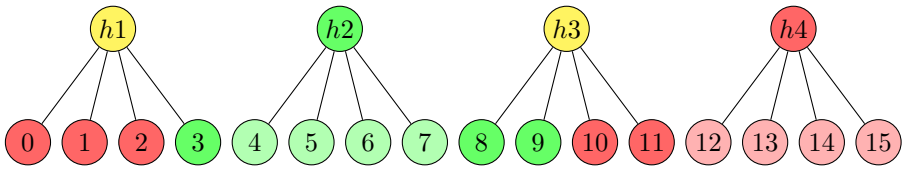
- *tutti gli elementi rappresentati dall'elemento hub vanno considerati* `a <= l && r <= b`

  In tal caso modifichiamo/analizziamo soltanto l'elemento hub.

- *solo alcuni elementi rappresentati dall'elemento hub vanno considerati*

  In tal caso, se sull'elemento hub ci eravamo segnati che gli elementi "reali" da lui rappresentati andavano modificati, propaghiamo questa modifica agli elementi "reali" stessi, cioè li andiamo a modifcare uno ad uno in modo che loro risultino settati al loro valore corrente e che l'elemento hub sia resettato al suo stato iniziale (cioè che non contenga nessuna modifica degli elementi che rappresenta). Poi consideriamo ciascuno degli elementi "reali" rappresentati dall'elemento hub e, se appartiene all'intervallo $[a, b)$, procediamo a modificarlo/analizzarlo.

Ad esempio, per $N = 16$, un'operazione sull'intervallo $[3, 10)$ si svolge così:



I nodi in giallo sono gli elementi hub che rientrano nel terzo caso, quelli in verde sono gli elementi che abbiamo considerato (cioè il secondo caso) mentre quelli in rosso sono quelli che abbiamo ignorato (cioè il primo caso).

I colori scuri indicano gli elementi che abbiamo modificato/analizzato direttamente cioè di cui siamo andati a scrivere/leggere il valore. I colori chiari indicano gli elementi il cui valore non è stato considerato direttamente ma è stato appreso tramite un elemento hub.

Notiamo che al massimo due elementi hub rientrano nel terzo caso, cioè quelli che "coprono" l'inizio e la fine dell'intervallo $[a, b)$.

È facile notare quindi che la complessità di ogni operazione risulta essere di $O(\sqrt{N})$, ottenendo quindi una complessità totale di $O(Q \cdot \sqrt{N})$.

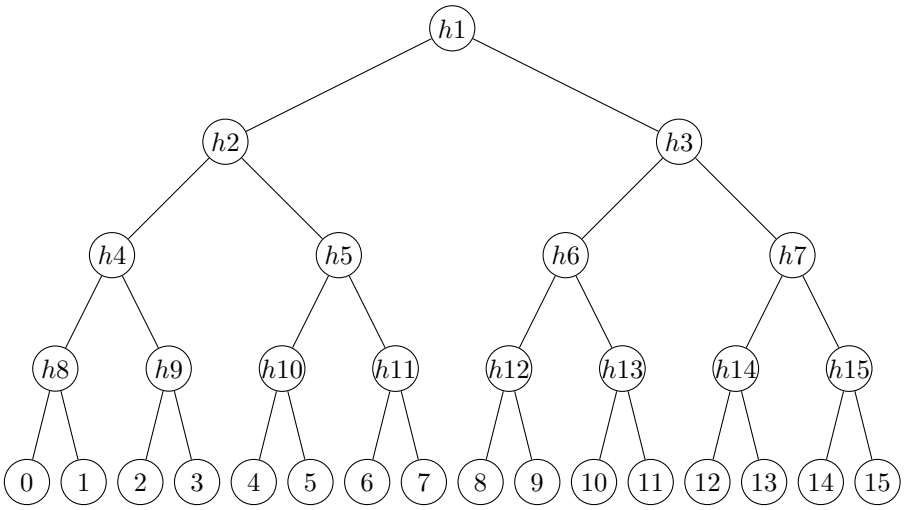
1.4 Soluzione ottima

Però si può fare ancora meglio. L'idea è di applicare il concetto di "elementi hub" anche agli elementi hub stessi, creando quindi degli elementi "super-hub" che rappresentano altri elementi hub. Questo processo può essere ripetuto fino ad ottenere un elemento che rappresenta tutti gli elementi "reali" della sequenza.

In questo modo si ottiene un albero, le cui foglie sono gli elementi "reali" della sequenza e i cui restanti nodi sono gli elementi hub, ognuno dei quali rappresenta gli elementi "reali" corrispondenti alle foglie del suo sottoalbero.

La versione più semplice di un albero del genere è un albero binario, cioè un albero in cui ogni elemento hub rappresenta esattamente altri due elementi.

Con $N = 16$, questo si può rappresentare graficamente nel seguente modo:



Un'operazione su questa struttura si svolge a grandi linee nello stesso modo descritto in precedenza, con un paio considerazioni in più.

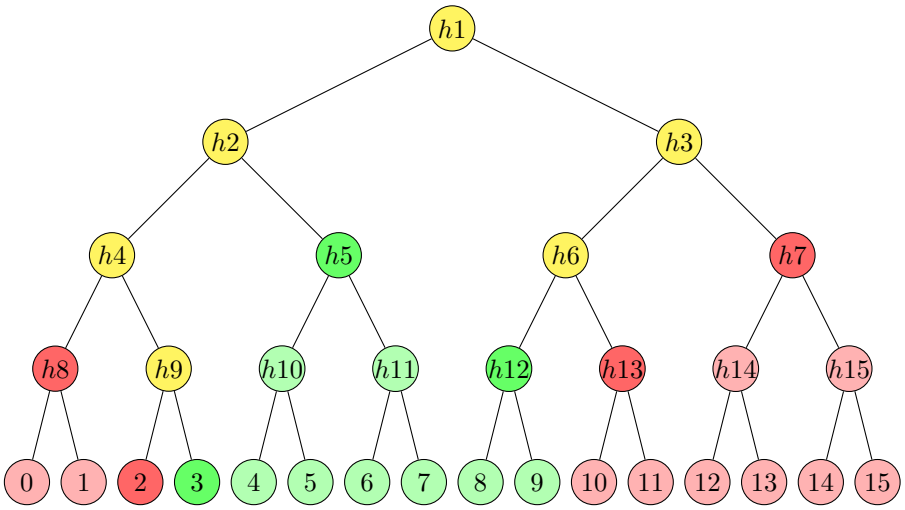
Intanto notiamo che un elemento hub non va considerato se è già stato considerato uno dei suoi antenati nell'albero.

Inoltre notiamo che, se un elemento hub rientra nel terzo caso, allora possiamo limitarci a propagare le informazioni in esso contenute solamente ai suoi figli (e non a tutti gli elementi che esso rappresenta) in quanto, quando passeremo ad analizzare i figli, saranno loro a propagarle a loro volta, se necessario.

Queste osservazioni suggeriscono un approccio ricorsivo al problema, cioè:

- *consideriamo il nodo radice* - se non è sufficiente modificare/analizzare questo nodo, propaghiamo le informazioni ai figli e ricorriamo su di essi:
  - *consideriamo il figlio sinistro* - se non è sufficiente modificare/analizzare questo nodo, propaghiamo le informazioni ai figli e ricorriamo:
    - *consideriamo il figlio sinistro* ...
    - *consideriamo il figlio destro* ...
  - *consideriamo il figlio destro* - se non è sufficiente modificare/analizzare questo nodo, propaghiamo le informazioni ai figli e ricorriamo:
    - *consideriamo il figlio sinistro* ...
    - *consideriamo il figlio destro* ...

Vediamo ora, per $N = 16$, come si svolge un'operazione sull'intervallo $[3, 10)$.



Notiamo che gli elementi hub che rientrano nel terzo caso sono quelli che sono "a cavallo" dell'inizio o della fine dell'intervallo $[a, b)$.

Da questo si deduce facilmente che tutte le volte che ci troviamo nel terzo caso (tranne, al massimo, una volta) almeno una delle chiamate ricorsive si ritroverà in uno dei primi due casi, e quindi non proseguirà nella ricorsione.

Quindi il numero di elementi che verrà considerato sarà proporzionale all'altezza dell'albero (al massimo quattro elementi per ogni "livello") e quindi la complessità di ogni operazione sarà $O(\log N)$, dando una complessità totale di $O(Q \cdot \log N)$.

1.5 Implementazione

Innanzitutto, per semplicità, supponiamo che N sia una potenza di 2. Se così non fosse, si può aumentare N fino a farlo diventare tale.

Ora costruiamo l'albero utilizzando un array di $2N$ elementi. Mettiamo la radice nella posizione di indice 1 e, se un nodo è nella posizione di indice i, mettiamo il suo figlio sinistro nella posizione di indice $2i$ e il suo figlio destro nella posizione di indice $2i + 1$ (quindi il padre sarà nella posizione $\lfloor i/2 \rfloor$).

In questo modo tutte le posizioni dell'array vengono utilizzate tranne quella di indice 0 (possiamo permetterci di sprecare una posizione, in cambio di una così facile relazione tra gli indici dei nodi)

Per risalire dall'indice i di un nodo all'intervallo $[l, r)$ che esso rappresenta si può procedere in questo modo:

- indichiamo con j l'indice del bit più significativo di i (cioè 2^j è la più grande potenza di 2 minore o uguale a i)
- indichiamo con d il valore $N/2^j$
- allora $l = (i - 2^j) \cdot d$ e $r = l + d$

In alternativa si possono passare l e r come parametri alla funzione ricorsiva.