Dynamic connectivity

Problema

Dato un grafo di N nodi, vuoi supportare le operazioni:

- Aggiungi un arco (a, b).
- Rimuovi un arco (a, b).
- Conta il numero di componenti connesse

Idea

Se non avessimo operazioni di rimozione, potremmo risolvere il problema facilmente usando una dsu. Purtroppo non siamo capaci di fare l'undo di un'operazione.

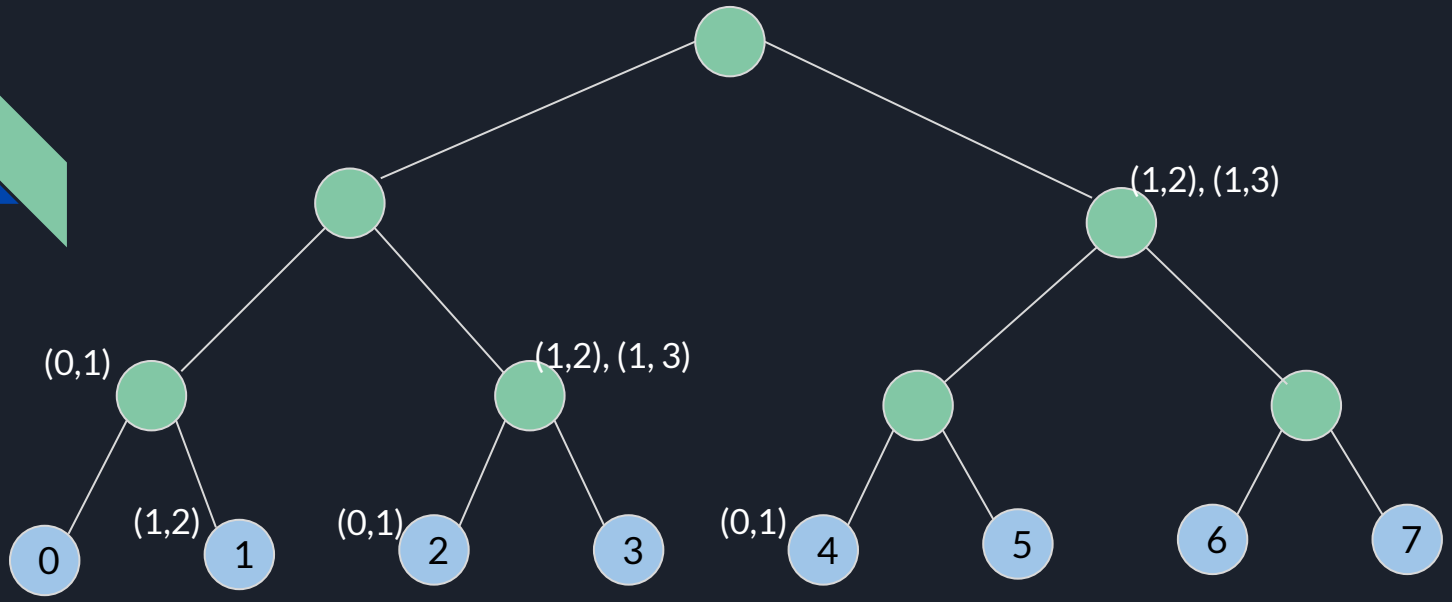
Siamo però capaci di fare un rollback dell'ultima operazione eseguita, ci basta mantenere uno stack delle operazioni e salvarci quali valori vengono cambiati per poterli riportare allo stato precedente.

Cerchiamo di usare questa operazione per risolvere il problema in modo offline.

Idea

Creiamo un segment tree sul tempo, dove ogni foglia quindi rappresenta un istante di tempo.

Notiamo che ogni arco (a,b) è presente nel grafo in uno (o più) intervalli di tempo. Aggiungiamo quindi questo arco ai nodi del segment tree corrispondenti.



Ad esempio se abbiamo le seguenti operazioni:

(1, (0,1)), (1, (1,2)), (1, (1, 3)), (0, (0, 1)), (1, (0,1)), (0, (0,1))

Gli archi sono presenti nei seguenti intervalli di tempo:

(0,1): [0, 2] e [4, 4] (1, 2): [1: ∞] (1, 3): [2: ∞]

Idea

Notiamo che per ogni query in input creiamo un intervallo, quindi in totale le operazioni sono O(Qlog(Q)).

Notiamo che ogni query avviene in un istante, e quindi rappresenta una foglia del nostro segment tree. Inoltre gli archi presenti nel grafo al tempo t sono tutti quelli che si trovano nel percorso tra la radice e la foglia t.

Soluzione

Per rispondere a tutte le query possiamo fare una DFS sul segment tree, quando entriamo in un nodo aggiungiamo tutti gli archi nel nodo alla DSU, quando usciamo da un nodo facciamo l'undo di quelle operazioni. Notiamo che le operazioni di cui fare l'undo sono un suffisso delle operazioni effettuate: ci basta fare il rollback!

Quando la DFS raggiunge una foglia calcoliamo la risposta per la query corrispondente.

Osservazioni

Questa tecnica si può applicare anche ad altri problemi e strutture dati, in generale:

Data una struttura dati con inserimenti in O(T(N)) **non ammortizzato** e query offline, possiamo supportare anche le rimozioni in O(T(N) log N).

Attenzione che nella DSU non possiamo fare la path compression dato che ha una complessità ammortizzata, ma possiamo fare union by size/rank.

Nel problema della connettività dinamica la complessità finale è O(Qlog(Q)log(N)) (un logaritmo è dato dal segment tree e uno dalla DSU)