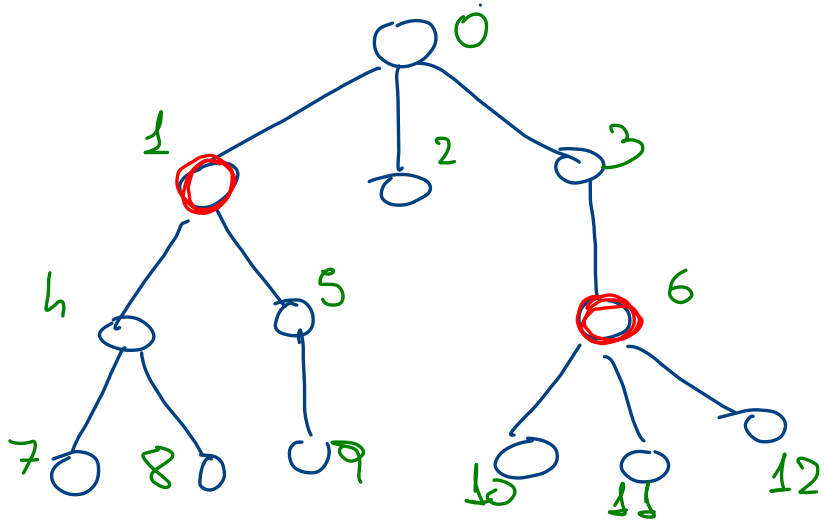
TECNICHE PER PROBLEMI SU ALBERI

Tecnica generale Usare l'ordine di DFS



Qual è l'ordine di DFS? (Partendo da 0)

0 1 4 7 8 5 9 2 3 6 10 11 12

(uno dei possibili)

Proprietà: I sottoalberi sono intervalli nella lista ottenuta!

Questo permette di gestire query/update sui nodi e sottoalberi costruendo una qualsiasi DS sull'array della DFS.

Esempio: I nodi hanno un peso.

Update: sostituisci il peso di v con w

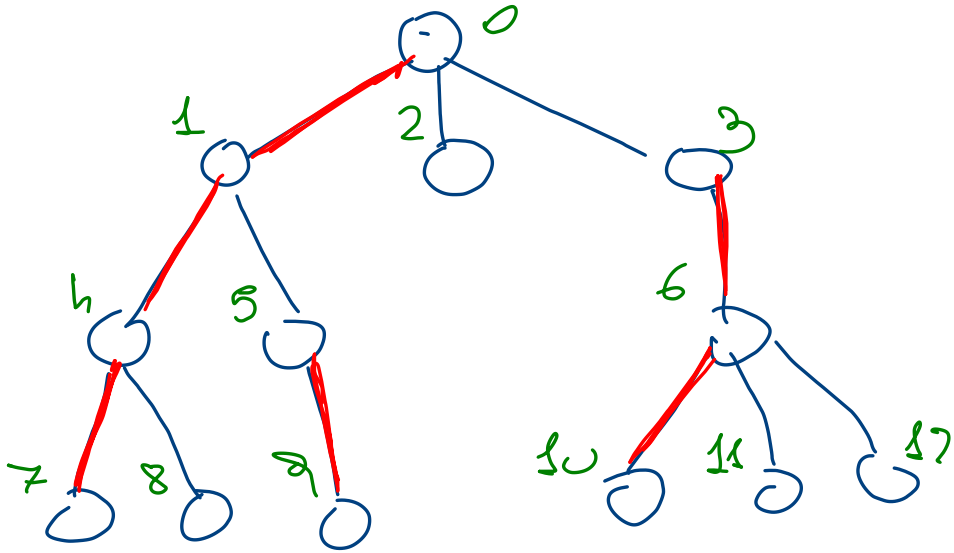
Query: somma dei pesi di un sottoalbero.

Esempio: Trovare l'LCA ($O(1)$ con $O(n \log n)$ prep.)

Remark: Verrà salvati i tempi di apertura e di chiusura di ogni nodo.

Heavy - Light Decomposition

Permette di trattare queries su cammini semplici.



Definizione: Un arco $u - v$ con u più vicino alla radice si dice "pesante" se

dimensione sottoalbero → $size(v) = \max_{w \in child(v)} size(w)$

(e a parità di size, quello di idx minore).

Proprietà: Gli archi pesanti formano delle catene (cammini che partono da un nodo e arrivano a una foglia).

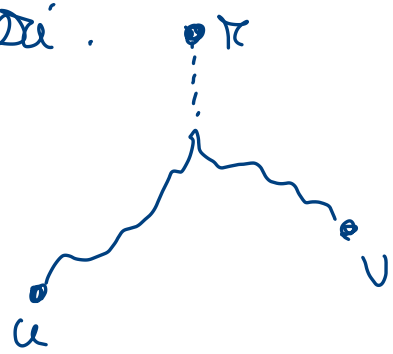
Proprietà: Per ogni v, il cammino $0 \longrightarrow v$ interseca $O(\log n)$ catene.

Dimostrazione: Se parto da v e risalgo, ogni volta che attraverso un arco leggero, size(u) raddoppia (almeno).

nodo corrente

⇒ Attraverso al più $\log_2 n$ archi leggeri. ∎

Corollario: Ogni cammino può essere decomposto in $O(\log n)$ "sottocatene" e $O(\log n)$ archi leggeri.



Quindi posso costruire DS su ogni catena.

$v_0 \longmapsto v_1$ = testa della catena di v_0

$\longmapsto v_2$ = padre di v_1

$\longmapsto \ldots$

$\longmapsto$ radice o $LCA(v_0, u_0)$

Trucco implementativo: Uso l'ordine di DFS, ma visito sempre per primo il figlio di dimensione maggiore. In questo modo le catene sono intervalli!

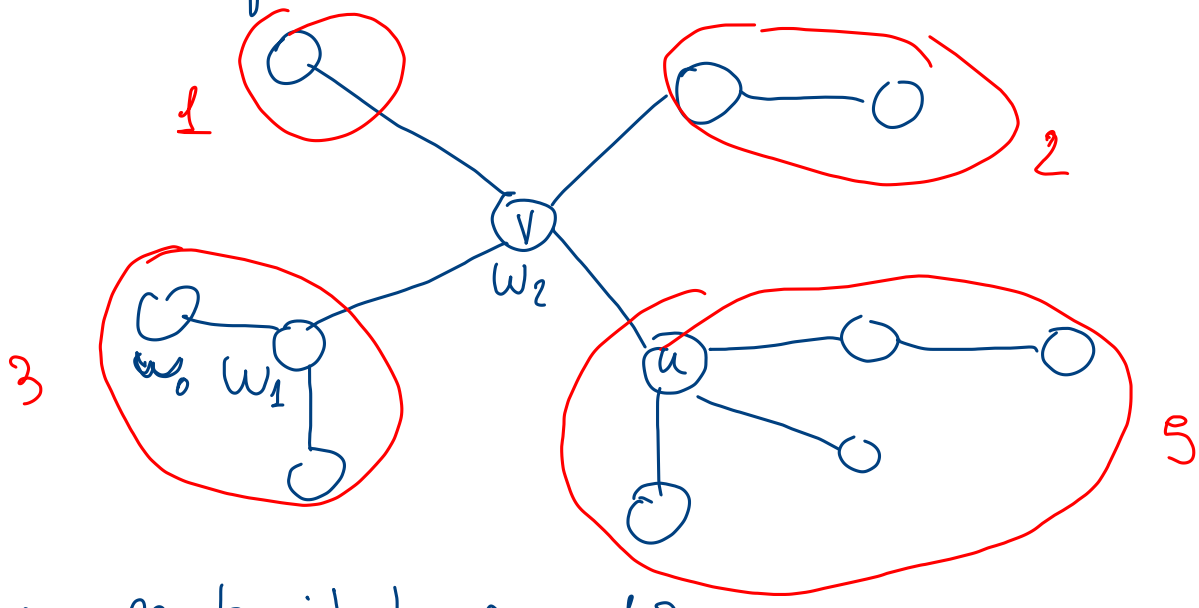
Nota: La complessità di una query è $O(\log n \cdot T(n))$ dove $T(n)$ è la complessità associata alla DS.

Centroid Decomposition

Torna utile "spesso" quando si ha a che fare con cammini.

Definizione: Si chiama "centroide" un nodo tale che, rimuovendo lui e tutti i suoi archi, i sottoalberi che si formano hanno size $\leq \frac{n}{2}$

Remark: In questo caso l'albero non è radicato.



v è un centroide! $n = 12$

Proprietà: Ogni albero ha almeno un centroide.

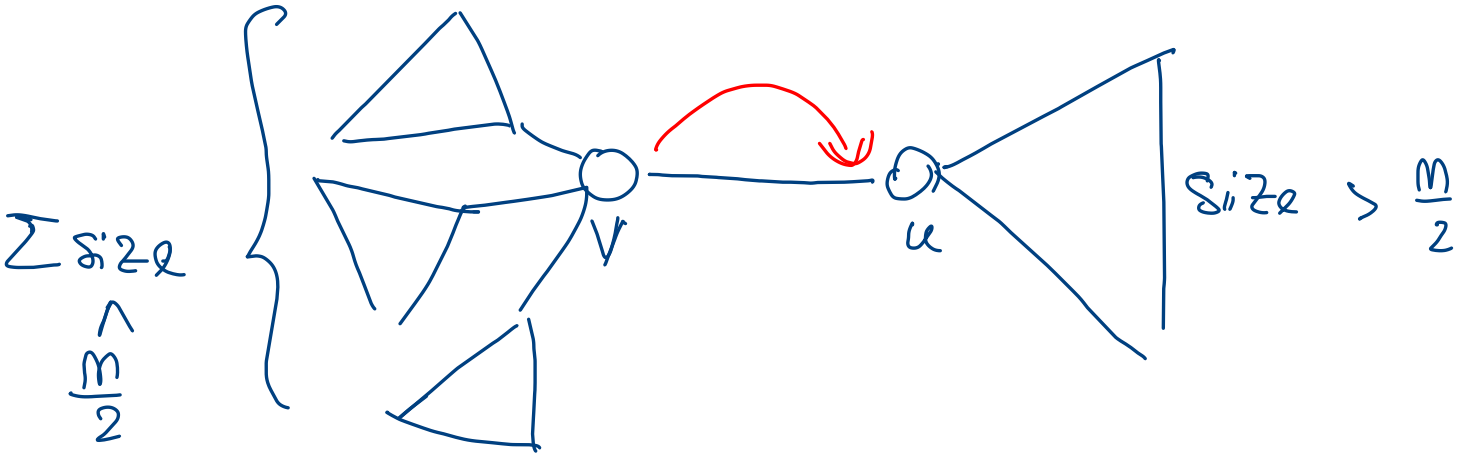
Proprietà: Se n è dispari, il centroide è unico.
Altrimenti, ci sono 1 o 2 centroidi, e se sono 2 sono adiacenti.

Dimostrazione della prima: Scelgo v a caso.

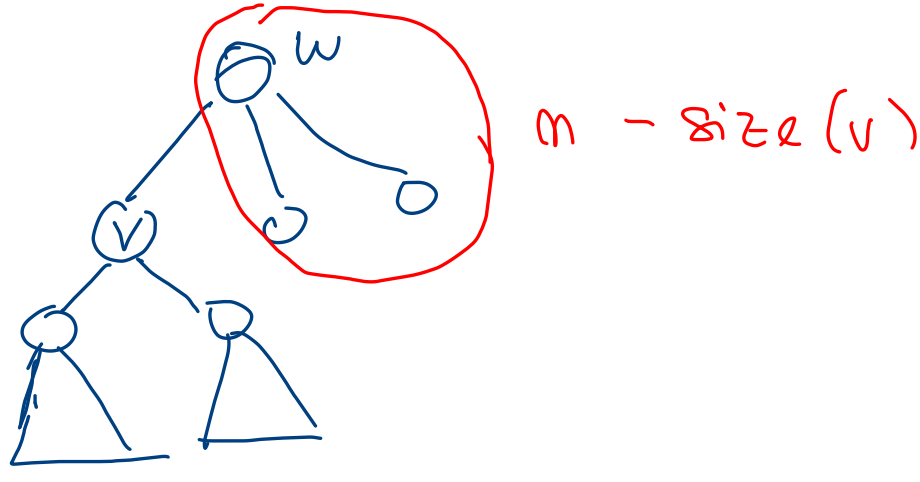
Se è un centroide ho finito.

Altrimenti, esiste uno e un solo vicino di v di size $> \frac{n}{2}$. Allora sostituisco v con tale vicino e ripeto.

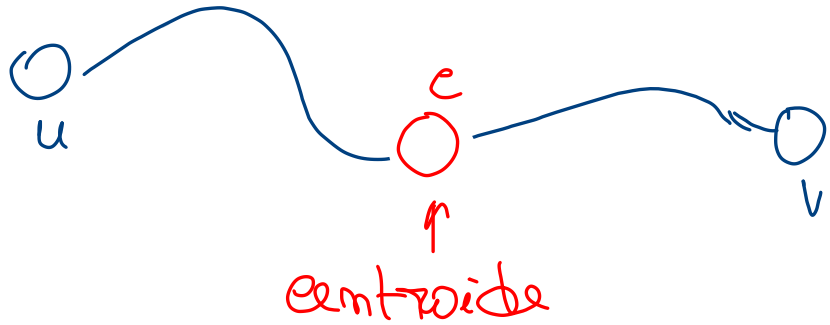
Prima o poi raggiungo un centroide.



Nota: Vanno precomputate le dimensioni dei sottoalberi (radicando l'albero a piacere).



La strategia generale è andare di Divide & Conquer.

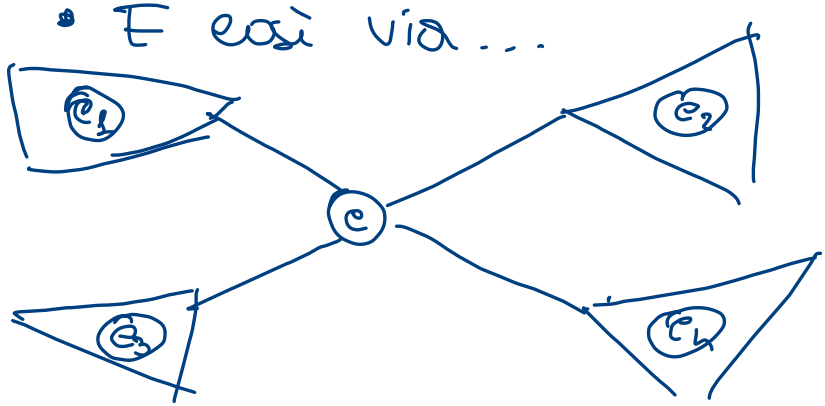


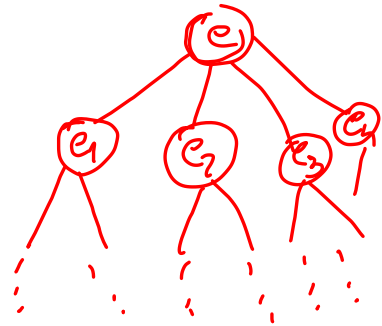
Dato un cammino u – v:

- Controllo se passa per il centroide. Se sì:
  - Spezzo in 2 nel centroide
  - Uso l'informazione precalcolata sui due pezzi.
- Se no, mi restringo al sottoalbero di u ev e ripeto.

Osservazione: È possibile costruire l'"albero dei centroidi", definito così:

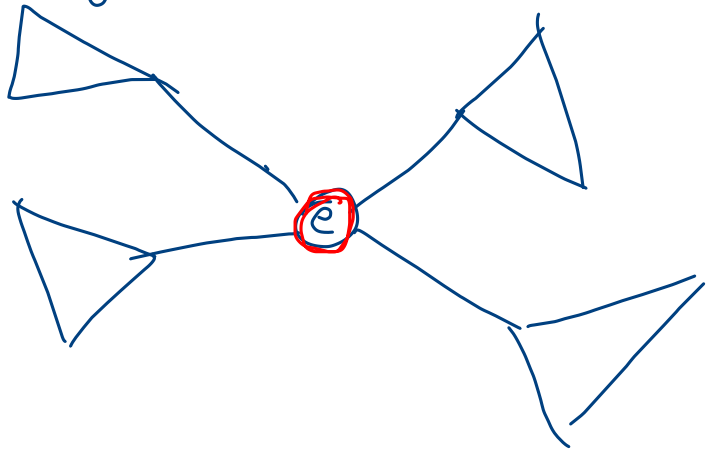
- La radice è il centroide
- I figli della radice sono i centroidi dei sottoalberi ottenuti togliendo il centroide
- E così via...





Proprietà: L'altezza dell'albero dei centroidi è $O(\log n)$.

Nota: Una stratagemma per trovare i centroidi è "colorare" i sottoalberi, cambiando colore ogni volta che scendo di livello.



"Heavy Child" Trick

Idea generale: quando devo unire due insiemi (o DS), conviene mergeare la più piccola nella più grande.

Torna utile in problemi in cui si ha bisogno di tenere una DS sui sottoalberi e "combinare" velocemente le DS dei figli di un nodo v per ottenere la DS di v.

Remark: Non funziona (di solito) in problemi dinamici.

Esempio: Dato un albero con un valore associato a ciascun nodo, determinare la moda di ogni sottoalbero, cioè il valore che compare più spesso.

Soluzione: Uso heavy child, utilizzando come DS una map valore $\rightarrow$ frequenza. Inoltre tengo la moda del sottoalbero corrente.

Il merge si fa semplicemente aggiungendo gli elementi della map più piccola a quella più grande.

Remark: Si implementa con una DFS.

- Se sono in una foglia, inizializzo una nuova DS.
- Altrimenti:
  - Ricorro sui figli
  - Unisco le DS dei figli mergeandole in quella del figlio più grande.

Nota: La complessità globale è $O(n \log n \cdot T(n))$, dove $T(n)$ è la complessità associata alla DS.