

ANALISI AMMORTIZZATA  
ALBERI SPLAY: ANALISI

---

**INFORMATICA**

# ANALISI DI SEQUENZE DI OPERAZIONI

- ▶ Possiamo valutare sequenze di operazioni invece di singole operazioni
- ▶ Data una sequenza di  $m$  operazioni su una struttura dati, valutiamo il loro costo computazionale  $T(m)$
- ▶ Il costo ammortizzato di una singola operazione è quindi dato da  $\frac{T(m)}{m}$
- ▶ Questo anche se la singola operazione ci può mettere di più!

# ANALISI DI SEQUENZE DI OPERAZIONI

- ▶ In poche parole, se anche abbiamo alcune delle operazioni che sono costose, il loro costo è *ammortizzato* da molte operazioni poco costose
- ▶ Ci sono tre principali metodi per compiere una analisi ammortizzata del tempo di calcolo:
  - ▶ Metodo dell'aggregazione (*aggregate analysis*)
  - ▶ Metodo del banchiere (*accounting method*)
  - ▶ Metodo del potenziale (*potential method*)

## FORMULAZIONE DEL PROBLEMA

Data una sequenza di  $m$  operazioni, vogliamo trovare il tempo  $T(m)$  che queste  $m$  operazioni richiedono nel caso peggiore

Indicheremo con  $c_1, c_2, \dots, c_m$  il costo delle  $m$  operazioni

Ne segue che  $T(m) = \sum_{i=1}^m c_i$

# ALGORITMO D'ESEMPIO PER LO STUDIO

- ▶ Come algoritmo di esempi consideriamo uno stack implementato come array
- ▶ Quando l'array è pieno, la sua dimensione viene raddoppiata tramite copia
- ▶ Le operazioni che possiamo svolgere sono push e pop

# ARRAY CON RADDOPPIO DI DIMENSIONE



push(3)



push(5)



push(2)



push(7)



push(9)

## METODO DELL'AGGREGAZIONE

Significa semplicemente che si calcola direttamente che valore assume  $T(m)$  e poi si divide per  $m$  per trovare il costo ammortizzato

Valutiamo il costo dell'inserimento in un array:

- ▶ Il costo di inserimento è 1 se non dobbiamo raddoppiare
- ▶ Altrimenti è pari al numero di elementi attualmente contenuti dell'array (dato che dobbiamo copiarli)

## METODO DELL'AGGREGAZIONE

Quindi il costo è  $c_i = \begin{cases} i & \text{se } i - 1 \text{ è una potenza di } 2 \\ 1 & \text{altrimenti} \end{cases}$

Indichiamo con  $d_i = c_i - 1$ , ovvero  $d_i$  è il costo di copia degli elementi diversi da quello inserito

Ne segue che  $d_i = \begin{cases} i - 1 & \text{se } i - 1 \text{ è una potenza di } 2 \\ 0 & \text{altrimenti} \end{cases}$

## METODO DELL'AGGREGAZIONE

La somma  $\sum_{i=1}^m c_i$  può essere quindi spezzata in due parti:

$$T(m) = \sum_{i=1}^m 1 + \sum_{i=1}^m d_i$$

Che corrisponde a  $T(m) = m + \sum_{i=1}^m d_i$

Ricordiamo che  $d_i = 0$  quando  $i - 1$  non è una potenza di 2

## METODO DELL'AGGREGAZIONE

Maggioriamo la somma  $\sum_{i=1}^m d_i$  con una somma di una quantità logaritmica di potenze di 2:

$$\sum_{i=1}^m d_i \leq \sum_{j=0}^k 2^j \text{ con } k = \lceil \log_2(m-1) \rceil$$

La somma delle prima  $k$  potenze di 2 è  $2^{k+1} - 1$

Ma  $2^{k+1} - 1 = O(m)$

## METODO DELL'AGGREGAZIONE

Quindi  $T(m) = m + O(m) = O(m)$

Il costo ammortizzato di una singola operazione è quindi costante:  $\frac{O(m)}{m} = O(1)$

**Conclusione:** raddoppiando la dimensione quando l'array è pieno il costo ammortizzato di ogni singola operazione è costante

## METODO DEL BANCHIERE

Il metodo del banchiere si basa sull'idea di associare ad ogni operazione un costo  $\hat{c}_i$  che può essere diverso dal costo reale.

La differenza  $\hat{c}_i - c_i$  può essere positiva (in questo caso l'operazione  $i$ -esima "deposita" la differenza di costo)

Se la differenza  $\hat{c}_i - c_i$  è negativa, allora l' $i$ -esima operazione deve "prelevare" usando quanto "depositato" dalle operazioni precedenti

## METODO DEL BANCHIERE

Abbiamo quindi che, per ogni  $1 \leq k \leq m$  deve valere che:

$$\sum_{i=1}^k \hat{c}_i - \sum_{i=1}^k c_i \geq 0,$$

ovvero ad ogni operazione abbiamo sempre "pagato" abbastanza da sostenere il costo (i.e., il bilancio è non negativo)

Abbiamo quindi che per  $k = m$  vale  $\sum_{i=1}^m \hat{c}_i \geq \sum_{i=1}^m c_i$

e quindi  $\sum_{i=1}^m \hat{c}_i$  ci fornisce un upper bound per  $T(m)$

# METODO DEL BANCHIERE

Ovviamente dipende tutto dalla scelta dei diversi  $\hat{c}_i$ .

Se scegliamo valori troppo piccoli la disuguaglianza non vale e non otteniamo un bound per  $T(m)$ .

Se li scegliamo troppo grandi non otteniamo dei bound stretti per il costo ammortizzato.

## METODO DEL BANCHIERE

Per il nostro esempio scegliamo  $\hat{c}_i = 3$  indipendentemente dall'operazione

Ipotesi: tutte le operazioni tra un raddoppio e il successivo "depositano" abbastanza per pagare il raddoppio

Consideriamo le prime  $i$  operazioni mostriamo che il bilancio rimane sempre non negativo.

## METODO DEL BANCHIERE

$$3 - 1 = 2 \quad \text{operazione 1}$$

$$6 - (1 + 2) = 3 \quad \text{operazione 2}$$

$$9 - (1 + 2 + 3) = 3 \quad \text{operazione 3}$$

Il bilancio rimane sempre positivo dopo le prime tre operazioni. Questo è il caso base.

Assumiamo quindi di avere bilancio non negativo dopo un'operazione di ridimensionamento e mostriamo che il bilancio acquisto durante le successive operazioni ci permette di "pagare" la successiva operazione di ridimensionamento

## METODO DEL BANCHIERE

Consideriamo la variazione del bilancio tra l'operazione  $k + 2$  l'operazione  $2k$  dove  $k$  è una potenza di 2 con  $k \geq 2$ .

L'ultima operazione di ridimensionamento è stata con l'operazione  $k + 1$  e la prossima sarà all'operazione  $2k + 1$

Quindi 
$$\sum_{i=k+2}^{2k} \hat{c}_i - c_i = 3(2k - (k + 1)) - (2k - (k + 1)) = 2k - 2$$

Questo è quanto "depositato" tra un'espansione e la successiva

## METODO DEL BANCHIERE

Consideriamo ora l'operazione  $2k + 1$ , che è una espansione.

Il costo è  $2k + 1$ , ma:

$$\sum_{i=1}^{2k+1} \hat{c}_i - \sum_{i=1}^{2k+1} c_i \geq 2k - 1 + \hat{c}_{2k+1} - c_{2k+1} = 2k - 2 + 3 - (2k + 1) = 0$$

Questo mostra che il bilancio rimane non negativo dopo l'operazione di ridimensionamento.

Quindi il costo ammortizzato  $\hat{c}_i = 3$  è sufficiente

## METODO DEL BANCHIERE

$$T(m) = \sum_{i=1}^m c_i \leq \sum_{i=1}^m \hat{c}_i = 3m$$

Quindi il costo di una singola operazione è costante.

Nota: la parte difficile è la scelta di un buon costo  $\hat{c}_i$  per le operazioni: usare 2 sarebbe stato troppo poco, usare  $m$  avrebbe funzionato ma ci avrebbe dato un risultato non stretto

## METODO DEL POTENZIALE

Se le  $m$  operazioni che svolgiamo sono su una struttura dati che potenzialmente viene modificata da ogni operazione, otteniamo una sequenza stati della struttura dati:

$D_0, D_1, D_2, \dots, D_m$  dove  $D_i$  è lo stato della struttura dati dopo l'esecuzione dell' $i$ -esima operazione

$D_0$  è lo stato iniziale

$D_m$  è lo stato finale

## METODO DEL POTENZIALE

Definiamo una funzione  $\Phi$  che mappa gli stati della struttura dati in valori reali non negativi.

La funzione  $\Phi$  è detta **funzione potenziale** e  $\Phi(D_i)$  è il **potenziale associato alla struttura dati  $D_i$**

Definiamo quindi il costo ammortizzato come il costo reale sommato ad una variazione di potenziale:

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$

## METODO DEL POTENZIALE

Osserviamo ora come è definita la somma di tutti i costi ammortizzati

$$\sum_{i=1}^m \hat{c}_i = \sum_{i=1}^m (c_i + \Phi(D_i) - \Phi(D_{i-1}))$$

La serie  $\sum_{i=1}^m \Phi(D_i) - \Phi(D_{i-1})$  è una serie telescopica:

$$\Phi(1) - \Phi(0) + \Phi(2) - \Phi(1) + \dots + \Phi(m) - \Phi(m-1) = \Phi(m) - \Phi(0)$$

## METODO DEL POTENZIALE

Possiamo quindi riscrivere la somma dei costi ammortizzati come

$$\sum_{i=1}^m \hat{c}_i = \Phi(D_m) - \Phi(D_0) + \sum_{i=1}^m c_i$$

E, quindi, il costo reale della sequenza di  $m$  operazioni come:

$$\sum_{i=1}^m c_i = \Phi(D_0) - \Phi(D_m) + \sum_{i=1}^m \hat{c}_i$$

ovvero la somma dei costi ammortizzati meno la differenza di potenziale (solitamente definiamo  $\Phi(D_0) = 0$ )

## METODO DEL POTENZIALE

Se riusciamo a definire  $\Phi(D_m) \geq \Phi(D_0)$ , abbiamo

$$\sum_{i=1}^m \hat{c}_i = \Phi(D_m) - \Phi(D_0) + \sum_{i=1}^m c_i \geq \sum_{i=1}^m c_i$$

E quindi la somma dei costi ammortizzati ci fornisce un buon sulla somma dei costi reali

Come per i due metodi precedenti, la scelta della giusta funzione potenziale è quella che permette all'analisi del costo ammortizzato di funzionare.

## METODO DEL POTENZIALE

Nel nostro esempio scegliamo come funzione potenziale:

$$\Phi(D_i) = 2i - k$$

dove  $i$  è il numero di elementi contenuti nell'array e  $k$  è la dimensione dell'array

Notiamo che  $\Phi(D_i) \geq 0$  per ogni  $1 \leq i \leq m$ , dato che ogni array è sempre pieno almeno per metà

Calcoliamo il costo ammortizzato di una operazione

## METODO DEL POTENZIALE

Se l'operazione non richiede un incremento di dimensione dell'array allora:

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 2i + k - 2(i - 1) - k = 3$$

Se invece fosse richiesto di incrementare la dimensione dell'array:

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = i + (2i - 2(i - 1)) - i - 1 = 3$$

Quindi otteniamo che il costo ammortizzato di ogni operazione rimane costante

## METODO DEL POTENZIALE: ALBERI SPLAY

Per calcolare il costo ammortizzato di una sequenza di  $m$  operazioni in un albero splay che contiene  $n$  elementi useremo il metodo del potenziale

Per fare questo dobbiamo definire una funzione potenziale.

## METODO DEL POTENZIALE: ALBERI SPLAY

Definiamo:

$\text{size}(x)$  come il numero di elementi contenuti nel sottoalbero a radice  $x$

Noi considereremo il rango di un nodo:

$$\text{rank}(x) = \log_2(\text{size}(x))$$

Il potenziale è definito quindi come la somma del rango di

tutti i nodi dell'albero  $\Phi(D_i) = \sum_{x \in D_i} \text{rank}(x)$

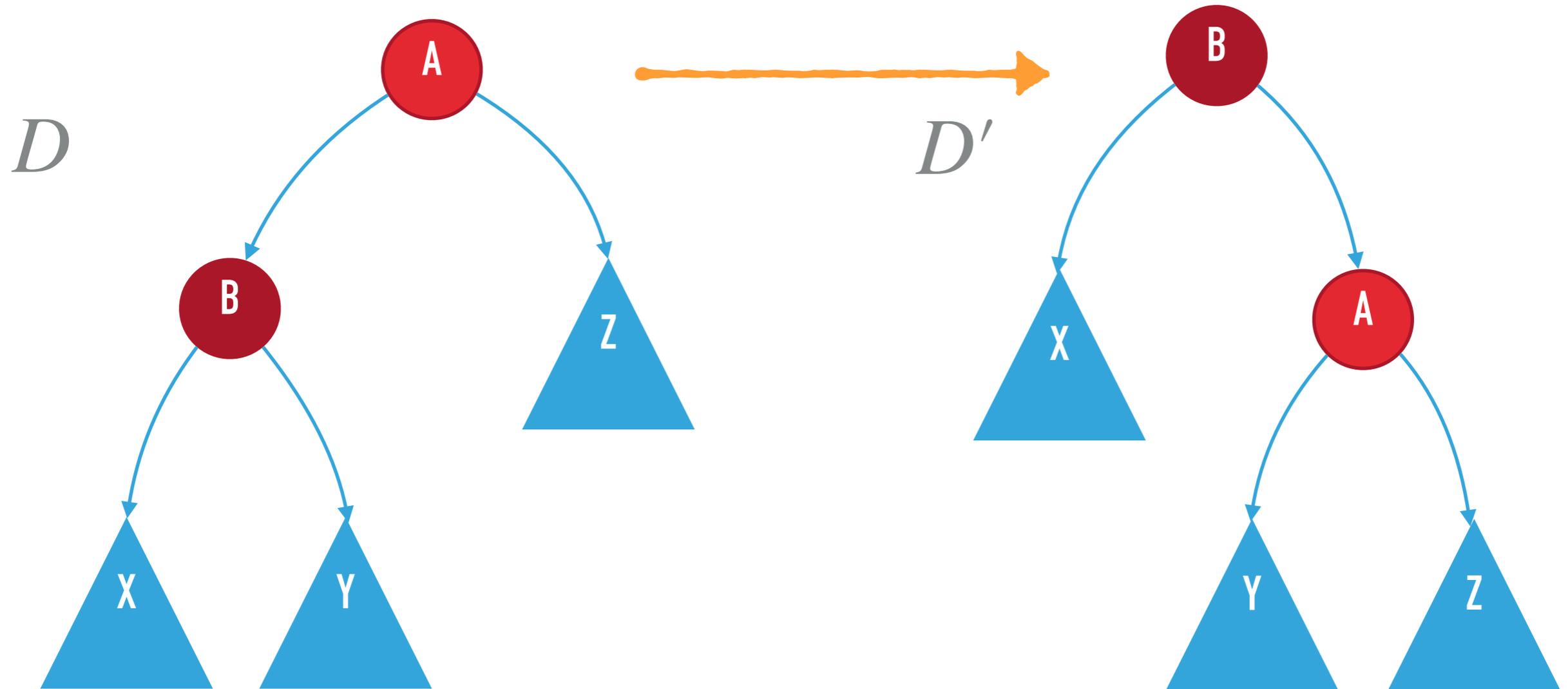
## METODO DEL POTENZIALE: ALBERI SPLAY

Studiamo ora come ognuna delle operazioni usate per spostare il nodo cercato alla radice modifica il potenziale.

Le operazioni sono quindi zig, zig zig e zig zag. Ognuna di queste può però venire applicata più volte all'interno di una singola operazione  $c_i$  di ricerca.

## CASO ZIG

Il nodo da muovere è figlio sinistro della radice



## METODO DEL POTENZIALE: ALBERI SPLAY

Per il caso zig la variazione di potenziale è

$$\Phi(D') - \Phi(D) = \text{rank}'(A) + \text{rank}'(B) - \text{rank}(A) - \text{rank}(B)$$

Perché questi sono gli unici nodi per cui cambia il valore

Ma  $\text{rank}(A) = \text{rank}'(B)$ , quindi  $\Phi(D') - \Phi(D) = \text{rank}'(A) - \text{rank}(B)$

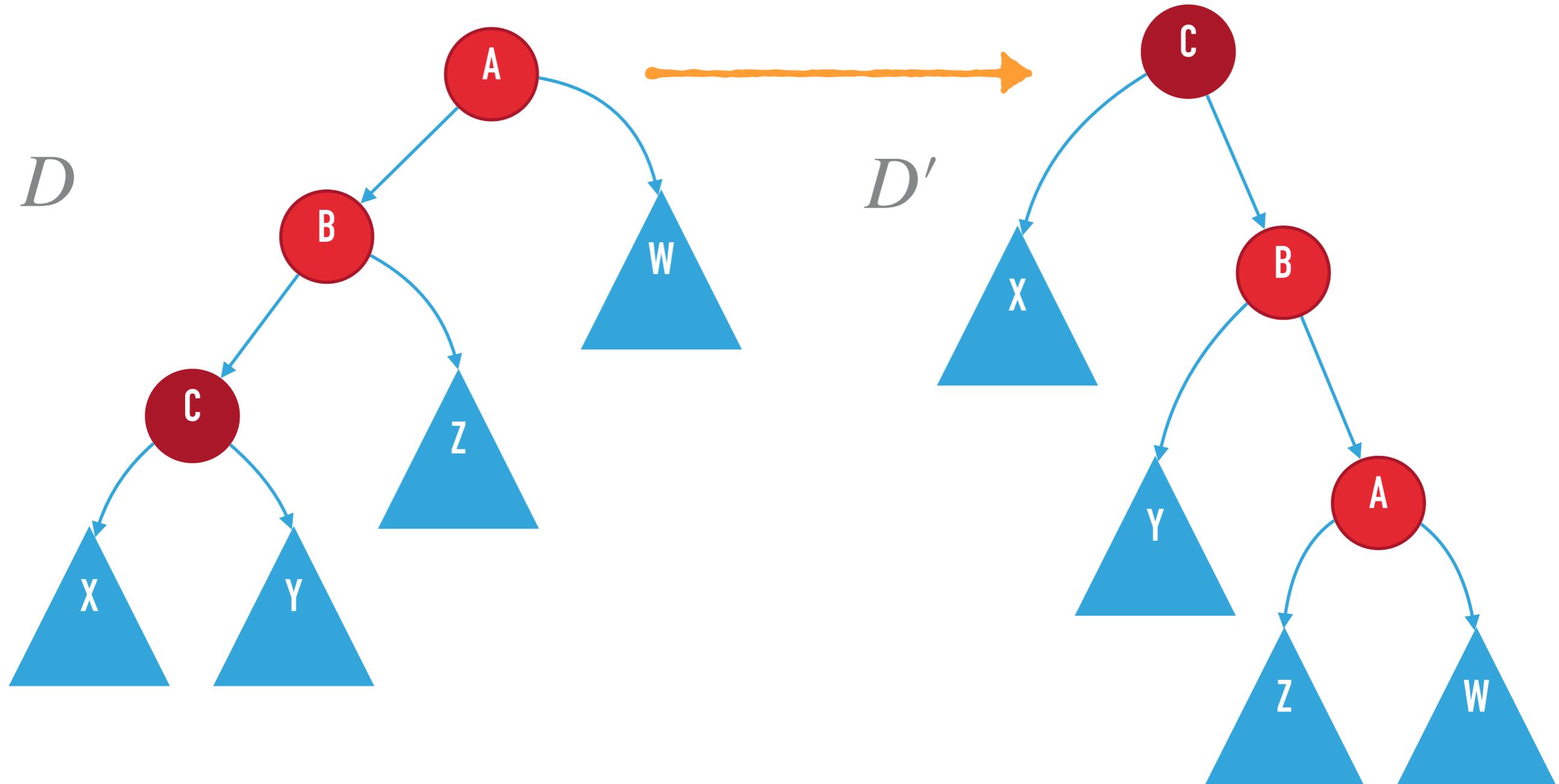
Dato che  $\text{rank}'(A) \leq \text{rank}'(B)$  possiamo scrivere:

$$\Phi(D') - \Phi(D) \leq \text{rank}'(B) - \text{rank}(B)$$

ovvero la variazione del rango solo del nodo cercato

# CASO ZIG ZIG

Il nodo da muovere è figlio sinistro di un nodo che è a sua volta figlio sinistro



## METODO DEL POTENZIALE: ALBERI SPLAY

Per il caso zig zig la variazione di potenziale è

$$\Phi(D') - \Phi(D) = \text{rank}'(A) + \text{rank}'(B) + \text{rank}'(C) - \text{rank}(A) - \text{rank}(B) - \text{rank}(C)$$

Perché questi sono gli unici nodi per cui cambia il valore

Ma  $\text{rank}(A) = \text{rank}'(C)$ , quindi

$$\Phi(D') - \Phi(D) = \text{rank}'(A) + \text{rank}'(B) - \text{rank}(B) - \text{rank}(C)$$

Ricordando che  $\text{rank}(B) \geq \text{rank}(C)$ :

$$\Phi(D') - \Phi(D) \leq \text{rank}'(A) + \text{rank}'(B) - \text{rank}(C) - \text{rank}(C)$$

## METODO DEL POTENZIALE: ALBERI SPLAY

Ricordando anche che  $\text{rank}'(B) \leq \text{rank}'(C)$ :

$$\Phi(D') - \Phi(D) \leq \text{rank}'(A) + \text{rank}'(C) - 2\text{rank}(C)$$

poiché  $\text{rank}'(C) > \text{rank}'(A)$ :

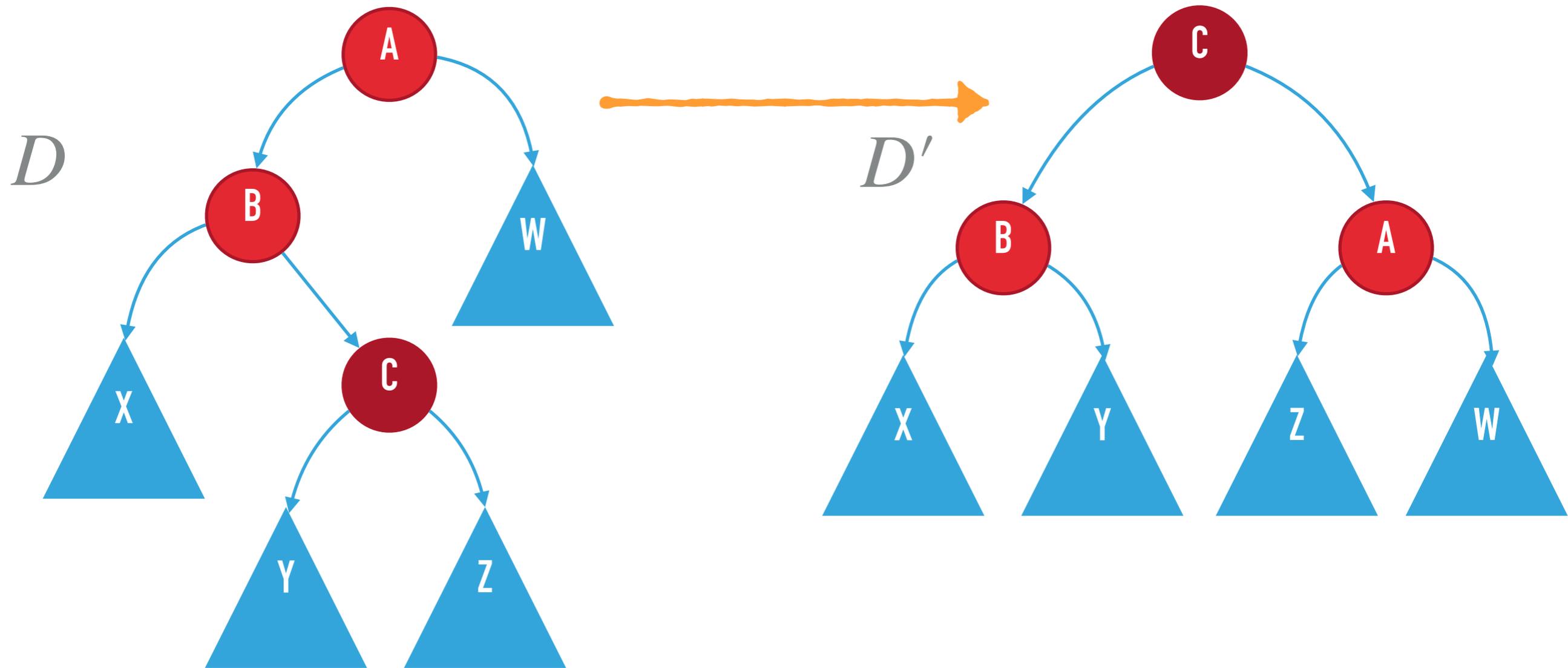
$$\Phi(D') - \Phi(D) \leq 2(\text{rank}'(C) - \text{rank}(C))$$

Perché ci servirà dopo facciamo questa maggiorazione

$$\Phi(D') - \Phi(D) \leq 3(\text{rank}'(C) - \text{rank}(C)) - 2$$

# CASO ZIG ZAG

Il nodo da muovere è figlio destro di un nodo che è figlio sinistro



## METODO DEL POTENZIALE: ALBERI SPLAY

Per il caso zig zag l'analisi è uguale al caso precedente:

$$\Phi(D') - \Phi(D) = \text{rank}'(A) + \text{rank}'(B) + \text{rank}'(C) - \text{rank}(A) - \text{rank}(B) - \text{rank}(C)$$

Perché questi sono gli unici nodi per cui cambia il valore

Ma  $\text{rank}(A) = \text{rank}'(C)$ , quindi

$$\Phi(D') - \Phi(D) = \text{rank}'(A) + \text{rank}'(B) - \text{rank}(B) - \text{rank}(C)$$

Ricordando che  $\text{rank}(B) \geq \text{rank}(C)$ :

$$\Phi(D') - \Phi(D) \leq \text{rank}'(A) + \text{rank}'(B) - \text{rank}(C) - \text{rank}(C)$$

## METODO DEL POTENZIALE: ALBERI SPLAY

Ricordando anche che  $\text{rank}'(B) \leq \text{rank}'(C)$ :

$$\Phi(D') - \Phi(D) \leq \text{rank}'(A) + \text{rank}'(C) - 2\text{rank}(C)$$

poiché  $\text{rank}'(C) > \text{rank}'(A)$ :

$$\Phi(D') - \Phi(D) \leq 2(\text{rank}'(C) - \text{rank}(C))$$

Perché ci servirà dopo facciamo questa maggiorazione

$$\Phi(D') - \Phi(D) \leq 3(\text{rank}'(C) - \text{rank}(C)) - 2$$

## METODO DEL POTENZIALE: ALBERI SPLAY

Se contiamo ogni rotazione come costo 1, abbiamo che il costo ammortizzato per spostare il nodo  $x$  è dato da

$1 + \text{rank}'(x) - \text{rank}(x)$  per le operazioni di zig

$2 + 3(\text{rank}'(x) - \text{rank}(x)) - 2$  per zig zig e zig zag

Se espandiamo una sequenza di operazioni vediamo che tutti i  $\text{rank}(x)$  e  $\text{rank}'(x)$  si cancellano tranne quello iniziale e l'ultimo, che è  $\text{rank}(\text{root}) = \log_2 n$

## METODO DEL POTENZIALE: ALBERI SPLAY

Il costo ammortizzato  $\hat{c}_i$  di una singola operazione è quindi

$$\hat{c}_i \leq \text{rank}(\text{root}) - \text{rank}_{D_{i-1}}(x_i)$$

Dove  $\text{rank}_{D_{i-1}}(x_i)$  indica il rango del nodo cercato nell'istruzione  $i$ -esima nella struttura  $D_{i-1}$ .

$$\text{Quindi } \hat{c}_i \leq \text{rank}(\text{root}) = \log_2 n$$

Per una sequenza di  $m$  operazioni il costo ammortizzato è quindi  $O(m \log n)$

## METODO DEL POTENZIALE: ALBERI SPLAY

Abbiamo completato l'analisi? No

Ricordate: 
$$\sum_{i=1}^m c_i = \Phi(D_0) - \Phi(D_m) + \sum_{i=1}^m \hat{c}_i$$

Che nel nostro caso è 
$$\sum_{i=1}^m c_i = \Phi(D_0) - \Phi(D_m) + O(m \log_2 n)$$

Quindi dobbiamo anche trovare quale sia la variazione di potenziale nell'albero

## METODO DEL POTENZIALE: ALBERI SPLAY

Possiamo sommare elemento per elemento

$$\sum_{x \in D_m} \text{rank}_{D_m}(x) - \sum_{x \in D_0} \text{rank}_{D_0}(x)$$

ma dato che l'albero contiene  $n$  nodi e la differenza massima di potenziale è  $\log n$ , sappiamo che possiamo migliorare

con  $\sum_x \log n = n \log n$

## METODO DEL POTENZIALE: ALBERI SPLAY

Otteniamo quindi che

$$\sum_{i=1}^m c_i = \Phi(D_0) - \Phi(D_m) + \sum_{i=1}^m \hat{c}_i = O(n \log n + m \log n)$$

Che è il risultato che volevamo per il **Balance Theorem**.