

$$S = \{S_1, \dots, S_m\}$$

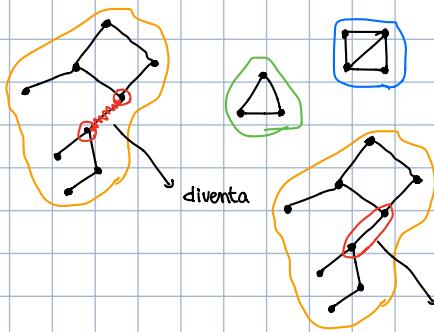
Si insieme di oggetti es. $S_i = \{x_{i1}, x_{i2}, \dots, x_{ij}\}$

$$S_i \cap S_j = \emptyset \text{ se } i \neq j \quad S_i \neq \emptyset$$

- OPERAZIONI :
- **Make(x)** costruire $\{x\}$ come SINGOLETTO
 - **Find(x)** restituire l'insieme S_i t.c. $x \in S_i$
 - **Union(x,y)** se x, y appartengono ad insiemi diversi, $x \in S_i$ e $y \in S_j$, sostituire S_i e S_j con $S_i \cup S_j$
↳ riduce il # di insiemi

es: $G = (V, E)$

Comp. connesse \Rightarrow



- COSTRUISCO $\{x\}$ per ogni $x \in V$

- Per ogni arco $\{x, y\} \in E$, faccio UNION(x,y)

- Alla fine ho un insieme di modi per ogni comp. connessa

eseguo $\Theta(|V|)^{\# \text{modi}}$ op. di MAKE

$\Theta(|E|)$ op. di UNION e FIND (non metto Θ perché per alcuni modi magari li ho già uniti) \longrightarrow es.



con gli archi \rightarrow ho già unito tutto, non servono tutti gli archi

il costo effettivo dipende da come sono implementati gli insiemi disgiunti

NOTA

analizzeremo il costo di m operaz. complessive (Make, Union, Find) di cui m operaz. MAKE

$$m = m + u + f$$

\downarrow \downarrow \downarrow
MAKE UNION FIND

$$m \geq m$$

$$u \leq m-1$$

ogni volta che faccio la UNION, riduco di 1 ie # di insiemi \rightarrow al max $m-1$ $\Theta(m)$

LISTE CONCATENATE $O(m+n)$

Implementazioni

ALBERI $O(m \cdot m)$

Ottimizzazioni

(nelle LISTE) WEIGHTED UNION \rightarrow la UNION normale mette in 2 posizioni la lista più corta $O(m + m \log m)$

$O(m \log n)$

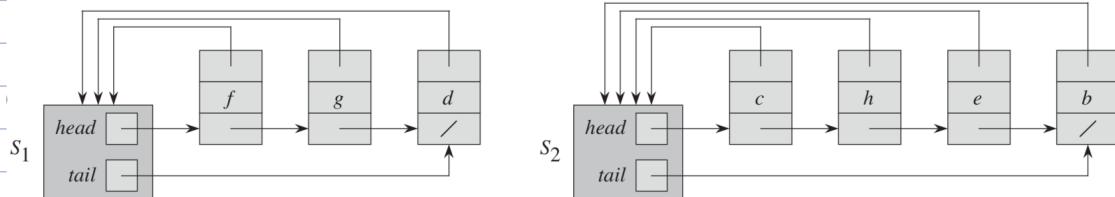
(merge ALBERI) UNION BY RANK \rightarrow la UNION normale crea ALBERI SBILANCIALI

PATH COMPRESSION

$O(m \cdot d(m, m))$ \rightarrow modifica la FIND per abbassare l'altezza dell'albero

Applicazione: Minimum Spanning Tree

M/F/U con LISTE CONCATENATE

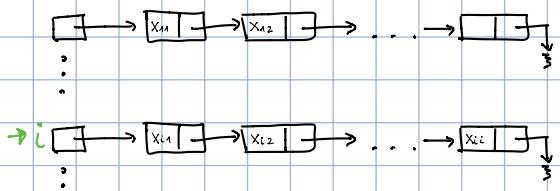


X è già nel formato corretto, è un oggetto record di una lista concatenata

X. key \rightarrow info

X. next \rightarrow punt. al succ.

- Find(x) ? \rightarrow x è un PUNTATORE, so già dove cercarlo ma non so in che insieme si.



Per rendere questa operazione meno costosa (ovvero sapere subito come trovare si)

\rightarrow in ogni elemento della lista metto anche un puntatore al primo elemento della lista (così dato x, guardo X. rappre. e so in H(1) in che ins. si trova)

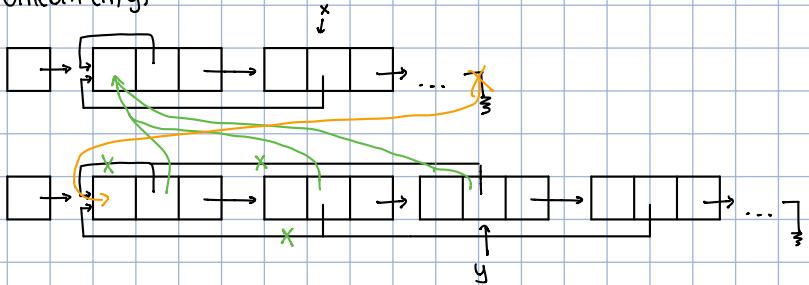


X. rappre. \rightarrow puntatore al rappre.

X. key \rightarrow info

X. next \rightarrow punt. al succ.

- Union(x,y) ?



Concateno le liste di y a quella di x

$\Theta(|Sx| + |Sy|)$

- scandisco tutta Sx per arrivare all'ultimo

- scandisco tutta Sy per modificare tutti i rappresentanti

Posso risparmiare una delle due scansioni?

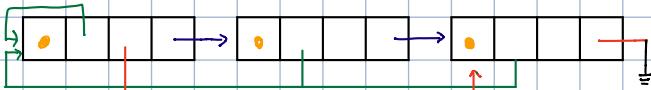
Salvo un puntatore all'ultimo elemento della lista \rightarrow campo aggiuntivo negli oggetti della lista

X. **rap** \rightarrow puntatore al rapp.

X. **key** \rightarrow info

X. **mext** \rightarrow punt. al succ.

X. **last** \rightarrow punta all'ultimo (solo il rappresentante)



\rightarrow setto il mext dell'ultimo elemento di S1 con il primo el. di S2
aggiorno il last di S1 con l'ultimo elemento di S2
modifico per tutti gli Sx il rappresentante, mettendo il 1° elem. di Sx

Per eseguire $\text{Umiom}(x, y)$ scandisco solo S_y

X è un PUNTATORE ad un oggetto, non una chiave

```
Make(x) {  
    // x è già del tipo corretto  
    x.rap  $\leftarrow$  x // x.Key è già inizializzato  
    x.mext  $\leftarrow$  NIL  
    x.last  $\leftarrow$  x // costruisce il SINGOLETTO  
    return x  
}
```

```
Fimd(x) {  
    return x.rap  
}
```

```
Umiom(x, y) {  
    z  $\leftarrow$  Fimd(x)  
    w  $\leftarrow$  Fimd(y)  
  
    if (z  $\neq$  w)  
        Link(z, w)  
}
```

```
Link(z, w) {  
    // z, w sono rapp. di due liste diverse z  $\neq$  w  
    z.last.mext  $\leftarrow$  w  
    z.last  $\leftarrow$  w.last  
    while (w  $\neq$  NIL) {  
        w.rap  $\leftarrow$  z  
        w  $\leftarrow$  w.mext  
    } // scandisce tutta la seconda lista per aggiornare  
    // il rappresentante di tutti gli elementi della lista (vecchia) Sy
```

Costi: 1 **MAKE** $\Theta(1)$
1 **FIND** $\Theta(1)$
1 **UNION** $\Theta(|L_y|)$ lunghezza della seconda lista

m operaz. comp. M/U/F di cui m **MAKE** :

Make $\Theta(1) \cdot m \rightarrow \Theta(m)$

Fimd $\Theta(1) \cdot f_{\leq m} \rightarrow \Theta(1) \cdot O(m) \rightarrow O(m)$

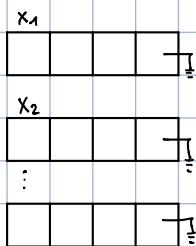
$$m = m + 1 + f$$

$$\Rightarrow O(m) + O(m^2) = O(m + m^2)$$

Umiom $O(m-1) \cdot m \stackrel{m-1}{\rightarrow} O(m) \cdot O(m) = O(m^2)$
↳ lunghezza della seconda lista

È possibile arrivare, nel caso peggiore, a $\Theta(m+m^2)$?

Sí \rightarrow eseguo m operaz. di unione e creo i singoletti



poi $\text{Union}(x_2, x_1)$

$\text{Union}(x_3, x_1)$

$\text{Union}(x_4, x_1)$

:

$\text{Union}(x_i, x_1) \leftarrow x_i \text{ è lunga } 1 \text{ e } x_1 \text{ è lunga } (i-1) \rightarrow \text{costo } \Theta(i)$

:

$\text{Union}(x_m, x_1)$

\rightarrow nella UNION conta solo la lunghezza della seconda lista

nel CASO PEGGIORIHO la seconda lista che cresce di 1 ad ogni chiamata

il costo di tutte le Union: $\sum_{i=2}^m \Theta(i) = \frac{m \cdot m+1}{2} = \Theta(m^2)$

WEIGHTED UNION

per evitare il caso peggiore, aggiungo il campo **length**

X. **rappr.** \rightarrow puntatore al rappre.

X. **key** \rightarrow info

\Rightarrow Nelle UNION metto come seconda lista la PIÙ CORTA

X. **next** \rightarrow punt. di succ.

X. **last** \rightarrow punta all'ultimo (solo il rappresentante)

X. **length** \rightarrow aggiornato solo nel rappresentante (viene inserito in tutti i modi ma si aggiorna solo quello del primo)

OPERAZIONI:

Make(x) {
 $x.\text{rap} \leftarrow x$ // x è già del tipo corretto
 $x.\text{key} \leftarrow \text{key}$ // x.key è già inizializzato
 $x.\text{next} \leftarrow \text{NIL}$
 $x.\text{last} \leftarrow x$ \rightarrow costruisce il SINGOLETTO
 $x.\text{length} \leftarrow 1$
 return x
}

Find(x) {
 return $x.\text{rap}$
}

WeighUnion(x,y) {
 $z \leftarrow \text{Find}(x)$
 $w \leftarrow \text{Find}(y)$

 if ($z \neq w$) {
 if ($z.\text{length} > w.\text{length}$)
 WLink(z,w)
 else
 WLink(w,z)
 }
}

WLink(z,w) {
 $z.\text{last}.\text{next} \leftarrow w$
 $z.\text{last} \leftarrow w.\text{last}$
 $z.\text{length} \leftarrow z.\text{length} + w.\text{length}$
 while ($w \neq \text{NIL}$) {
 $w.\text{rap} \leftarrow z$
 $w \leftarrow w.\text{next}$
 }
 // scandisce tutta la seconda lista per aggiornare
 // il rappresentante di tutti gli elementi della lista (vecchia) Sy
}

COSTI: 1 Make $\Theta(1)$

1 Find $\Theta(1)$

1 Union $\Theta(|L_2|)$ $|L_2| = \min(|L_x|, |L_y|) \rightarrow O(m)$

m operaz. M/F/U di cui m MAKE usando Weighted Union

Posso valutare il costo di tutte le UNION analizzando quanti puntatori x .rap vengono modificati complessivamente

Concentriamoci su un oggetto x

Quante volte al più modifica x .rap? \rightarrow quante volte al più x può finire nella seconda lista?

Quante volte al più riesco ad unire la lista in cui sta x ad una lista lunga?

all'inizio x sta in una lista lunga 1) \downarrow UNION

ora lista lunga 2) \downarrow UNION con una lista \geq a quella di x (2)

ora lista lunga 4) \downarrow
: 8
: :

ora lista lunga m

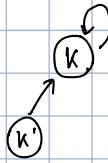
perché nel caso peggiore avrei la UNION tra liste sempre uguali

\hookrightarrow riesco a modificare le rappresentanti di x al più $\Theta(\log m)$ volte \Rightarrow una UNION è $O(m)$ e viene eseguita al max $\Theta(\log m)$

Quindi su tutti i possibili x $O(m \log m)$

COSTO COMPL WEIGHTED UNION: $O(m \log m)$

Complessivamente m operaz M/U/F di cui m Make con **WEIGHTED UNION**: $O(m + m \log m)$

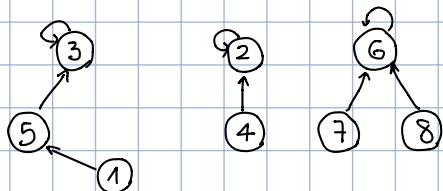


da ogni modo posso risalire al padre (non posso scendere nel figlio)

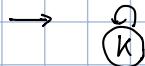
campi $x.parent$
 $x.Key$

- ↳ se mom ha padre, x.parent punta a se stesso
- ↳ ogni modo può avere un numero arbitrario di figli

es.

 $\{ \{1, 3, 5\}, \{2, 4\}, \{6, 7, 8\} \}$
**Make(x){**

$\forall x$ è un oggetto di tipo modo
 $x.parent \leftarrow x$

 $x.Key$ contiene già il valore

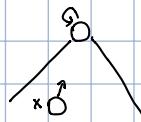
return x

}

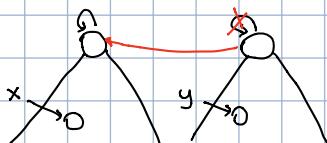
Find(x){

```
if( $x \neq x.parent$ ){
    return Find(x.parent)
} else {
    // caso base
    return x
}
```

La rappresentante di un albero è la radice

**Union(x,y){**

$z \leftarrow \text{Find}(x)$ \rightarrow rappresentante
 $w \leftarrow \text{Find}(y)$



if($z \neq w$){

 Link(z,w)

}

}

Link(z,w){

z, w sono due radici

$w.parent \leftarrow z$

}

COSTI:

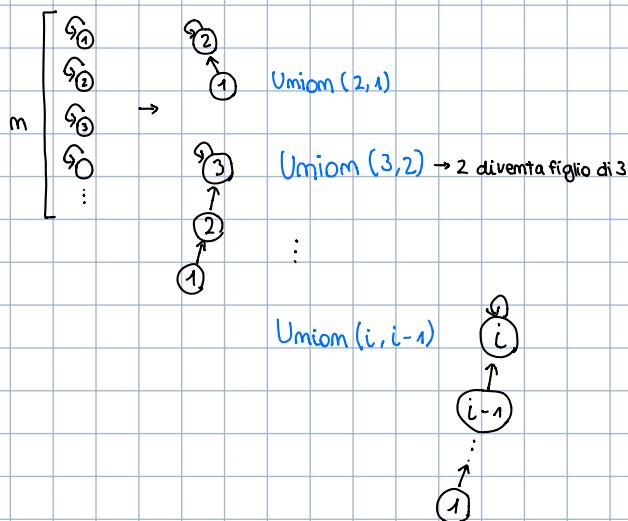
1 MAKE $\Theta(1)$ 1 FIND $O(m)$ 1 UNION $O(m)$

↳ ee due find

$$\begin{aligned}
 & \text{m MAKE e m OPERAZ.} \\
 & \Rightarrow \text{compiessivamente: } \frac{m \cdot \Theta(1)}{\text{MAKE}} + \frac{F \cdot O(m)}{\text{FIND}} + \frac{m \cdot O(m)}{\text{UNION}} \leq \frac{\Theta(m)}{\text{MAKE}} + \frac{m \cdot O(m)}{\text{FIND}} + \frac{(m-1) \cdot O(m)}{\text{UNION}} \\
 & \qquad \qquad \qquad = O(m \cdot m)
 \end{aligned}$$

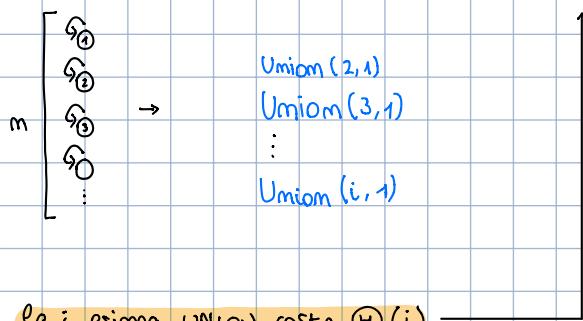
Si raggiunge la complessità $\Theta(m \cdot m)$ nel caso peggiore? si, facendo la UNION partendo dalle foglie

in **Make**



La i -esima UNION costa $\Theta(1)$

se invece scrivo $\text{Umiom}(\dots, 1)$ → così ogni volta parte dal fondo dell'albero (1) e lo risale tutto (ha altezza i) per trovare le PAPP da usare in **Link**



La i -esima UNION costa $\Theta(i)$

Alla fine ho  eseguo f $\text{Find}(1)$

$$\Theta(m) + \sum_{i=2}^m \Theta(i) + f \cdot \Theta(m) = \Theta(m) + \Theta(m^2) + f \cdot \Theta(m) = \Theta(m \cdot m)$$

Come **MIGLIORARE** il costo?

Gli alberi non devono sbilanciarsi con le UNION.



- Devo aggiungere l'albero più basso a quello più alto
- L'altezza cresce solo quando unisco due alberi aventi la STESSA ALTEZZA

UNION BY RANK (rank ≈ altezza) → il rank è un'approssimaz. dell'altezza

Devo memorizzare il rank dell'albero

In ogni modo aggiungo un campo $x.rank$ che indica il rank (altezza) dell'albero radicato in x

Make(x) {

$\therefore x$ è un oggetto di tipo nodo

 →



$x.parent \leftarrow x$

$\therefore x$ key contiene già il valore

$x.rank \leftarrow 0$

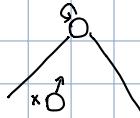
\therefore altezza di una foglia è 0

 return x

}

Find(x) {

\therefore La rappresentante di un albero è la radice



 if ($x \neq x.parent$) {

 return Find($x.parent$)

 } else { \therefore caso base

 return x

}

UnionByRank(x, y) {

\therefore il RANK AUMENTA solo quando faccio la UNION di due alberi con lo stesso rank

$z \leftarrow \text{Find}(x)$

$w \leftarrow \text{Find}(y)$] → rappresentanti

 if ($z \neq w$) {

 if ($z.rank > w.rank$) {

 Link(z, w)

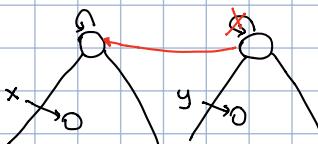
 } else {

 Link(w, z) $\therefore w$ ora è nuovo rapp.

 if ($w.rank = z.rank$)

$w.rank \leftarrow w.rank + 1$

 }



 → costa ancora $O(|S_x| + |S_y|) = O(m)$

}

Link(z, w) {

$\therefore z, w$ sono due radici

$w.parent \leftarrow z$

}

es: Costruire gli insiemi disgiunti con chiavi 1, 2, ..., 10

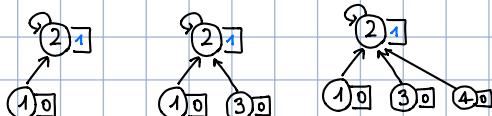
Eseguire $\text{Umiom}(2,1)$, $\text{Umiom}(3,1)$, $\text{Umiom}(4,1)$, $\text{Umiom}(7,8)$, $\text{Umiom}(1,8)$

Utilizzando Umiom By Rank

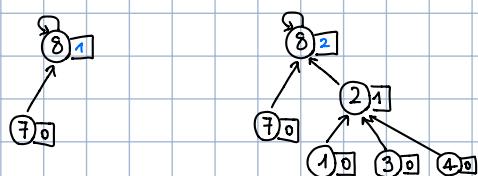
Nel caso di alberi aventi lo stesso rango, si utilizza come radice quello con chiave maggiore



$\text{Umiom}(2,1)$, $\text{Umiom}(3,1)$, $\text{Umiom}(4,1)$:



$\text{Umiom}(7,8)$, $\text{Umiom}(1,8)$



COMPLESSITÀ: (nel caso peggiore)

1 Make $\Theta(1)$

1 Find $O(h)$

1 Umiom By Rank $O(h)$

Complessivamente: $m \cdot \Theta(1) + f \cdot O(h) + u \cdot O(h) = \Theta(m) + O(m) \cdot O(h) \rightarrow h$ cosa ci esprime in m ?

$$m = m + f + u$$

Lemma: Utilizzando Umiom By Rank a partire da m operaz di Make si ottengono alberi aventi altezza $O(\log m)$

DIM: per induz. sull'ALTEZZA h

Ts) Un albero di altezza h costruito con Umiom By Rank contiene almeno 2^h modi

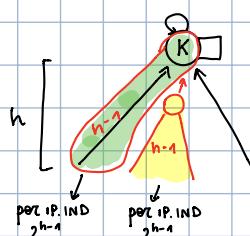
$$\hookrightarrow h \leq \log m$$

BASE: $h=0$

$$\Theta(0) \quad 2^0 = 1 \text{ OK}$$

PASSO: H_p ind) Un albero di altezza $h < h$ costruito con Umiom By Rank contiene almeno 2^h modi

Ts) Un albero di altezza h costruito con Umiom By Rank contiene almeno 2^h modi



Avevo unito in uno dei passi precedenti due alberi di altezza $h-1$ ottenendo un albero di altezza h

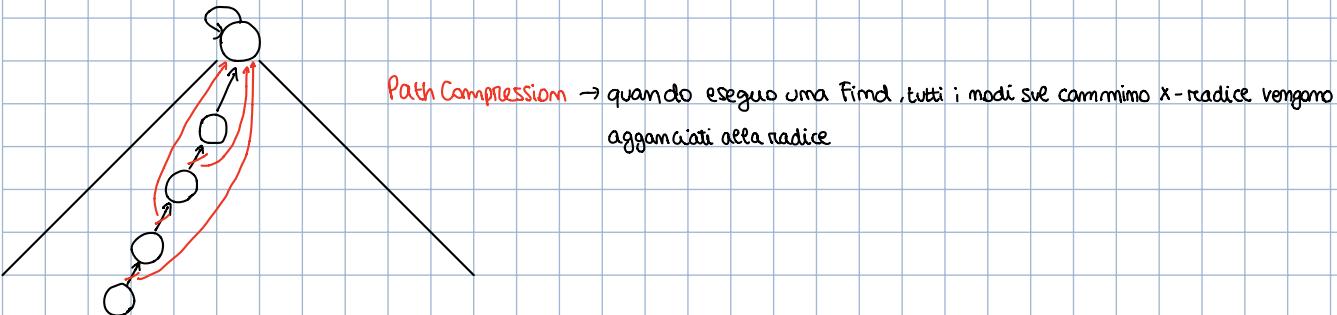
$$\text{almeno } 2^{h-1} + 2^{h-1} = 2 \cdot 2^{h-1} = 2^h$$

COSTI:

- (H)(1) Make
- O(log m) Find
- O(log m) UnionByRank

Complessivamente: $(H)(m) + O(m \log m) = O(m \log m)$

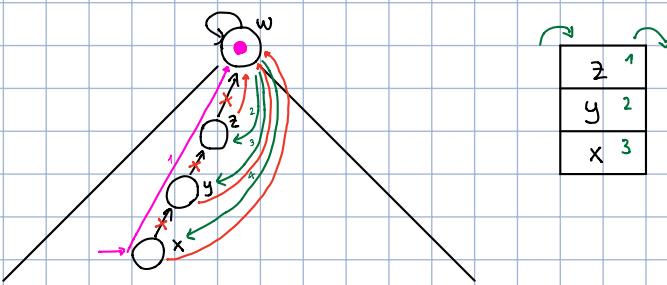
possiamo abbassare il costo della Find(x)? (se x è in fondo all'albero, ho costo $(H)(\log m)$ e ogni volta percorro lo stesso ramo)



Domanda: il ramo coincide con l'altezza? qui (dopo la Compression è \geq di h)

```
FindPC(x){    // La rappresentante di un albero è la radice
    if(x != x.parent){    // caso base
        x.parent ← FindPC(x.parent)
    } else {    // caso base
        return x
    }
}
```

Quando faccio Find(x) • , arrivavo al w salvando tutto STACK delle ch. ricorsive ie parent



COSTI:

m operazioni M/U/F con UnionByRank e P.C. di cui m Make hanno costo

$O(m \cdot \alpha(m, m))$

e' INVERSA

→ inversa funz. Ackermann (cresce molto lentamente), quella di Ackermann cresce più veloce dell'esponenziale)

→ è poco più di una costante

e^m e la sua inversa $\log m$

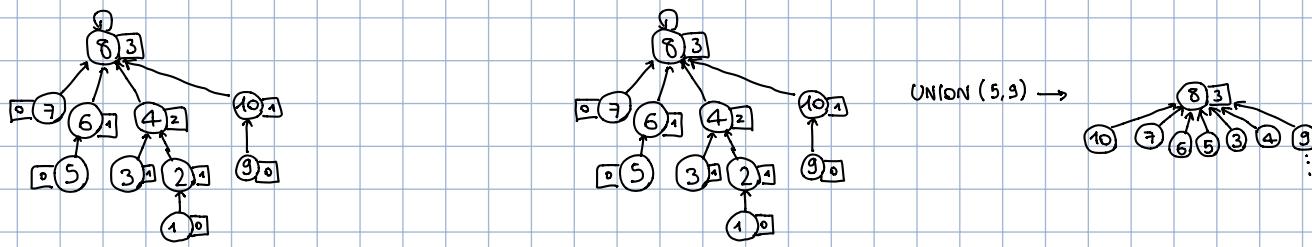
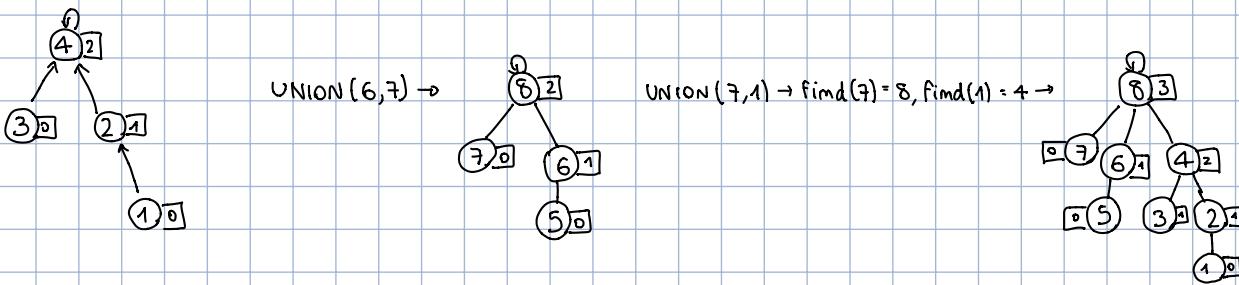
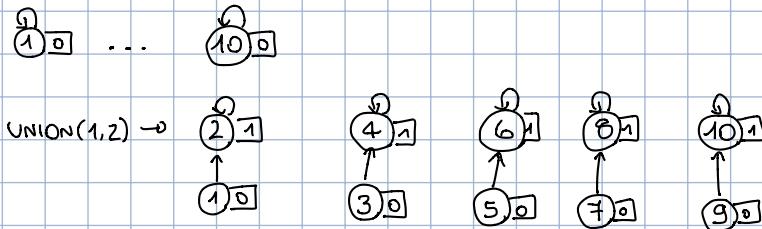
Nella migliore delle ipotesi, avremmo avuto comunque un costo orario \sqrt{m})

le RANGO: - senza Path Compression → ie RANGO = ALTEZZA

- con Path Compression → ie RANGO \geq ALTEZZA

es. 23/6/2017

con ALBERI e USO UNION BY RANK e PATH COMPRESSION 1... 10



es. COSTO LINEARE (powerpoint)

$O(m \cdot d(m, m))$

Make → Union → Find

Make → $\Theta(1) \cdot m = \Theta(m)$

Union → $\Theta(1) \cdot m = O(m)$

Find → $O(m)$

$$O(m) \cdot O(1) = O(m)$$

↓ archi

ogni arco creato con Make o Union viene percorso al più una volta.

conto il costo complessivo delle Find, contando quanti archi complessivamente scandiscono durante le find

Al più 1 volta
gli archi originali
 $O(m)$

Ogni arco nuovo può essere percorso
o per Find del modo stesso
o per una Find di uno dei figli

La prima volta che eseguo $\text{Find}(x)$ ha costo $O(\log m)$

Tutte le successive $\text{Find}(x)$ avranno costo $\Theta(1)$

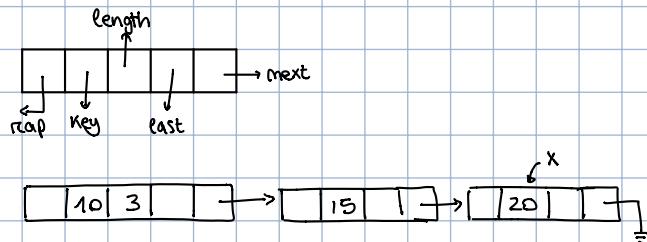
$$O(m \log m) + O(m - m) \cdot \Theta(1)$$

costo: $O(m \log m + m)$ è meglio di quello usando solo UnionByRank ? Si perché era $O(m \log m)$

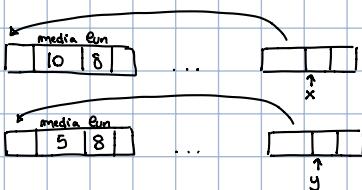
$$\text{se } m = \frac{1}{2}(m \log m) \rightarrow \Theta(m)$$

es. 19/06/18 MEAN su liste

LISTE WW



Calcolo media solo nel rappresentante:



$$\text{nuova media: } \frac{10 \cdot 10 + 5 \cdot 8}{18}$$

oppure memorizzare la somma

Union somma 1 + somma 2

Mean $\frac{\text{Somma}}{\text{lung}}$

es. 22/01/20 FIND PC inPLACE \rightarrow mom lo è perché è ricorsiva



$\text{FindPC-Impl}(x)$
 $y \leftarrow x$

while ($y \neq y.\text{parent}$) { \rightarrow con il while arriva a puntare al nullo.

$y \leftarrow y.\text{parent}$

}

while ($x \neq y$) {

$z \leftarrow x.\text{parent}$

$x.\text{parent} \leftarrow y$

$x \leftarrow z$

}

return y

COSTO: $O(\log m)$ 1 op. Find

COSTO COMPL: $O(m \cdot d(m, m))$

es. 21/6/2016 SOMMA

- Liste con W.U. + campo somma (aggiornato solo nel rappre.) $\rightarrow O(m + m \log m)$

- Alberi con UNIONBYRANK e PathCompr. $\rightarrow O(m \cdot d(m, m))$

+ campo somma aggiornato solo nella radice

$$m = \Theta(m)$$