STRUTTURE DATI PER INSIEMI DISGIUNTI

- E' SPESSO UTILE RAPPRESENTARE IN MANIERA EFFICIENTE FAMIGLIE DINAMICHE DI INSIEMI DISGIUNTI SOGGETTI ALLE SEGUENTI OPERAZIONI:

  MAKE_SET(x) - INTRODUZIONE DEL SINGOLETTO $\{x\}$

  UNION(x,y) - UNIONE DEGLI INSIEMI DISGIUNTI CHE CONTENGONO RISPETTIVAMENTE x E y

  FIND_SET(z) - RICERCA DELL'INSIEME CHE CONTIENE x

- TALI OPERAZIONI CONSENTONO LA GESTIONE DI RELAZIONI DI EQUIVALENZA DINAMICHE MONOTONE NON-DECRESCENTI

- ANALIZZEREMO SEQUENZE DI m OPERAZIONI MAKE_SET, UNION E FIND_SET COMPRENDENTI n OPERAZIONI MAKE_SET

- OVVIAMENTE:
  - $n \leq m$
  - # UNION $\leq n-1$

ESEMPIO DI UNA SEMPLICE APPLICAZIONE: MANTENIMENTO DELLE COMPONENTI CONNESSE DI UN GRAFO NON ORIENTATO CRESCENTE

DEFINIZIONE SIA $G=(V,E)$ UN GRAFO NON ORIENTATO. PONIAMO $u \sim v$, PER $u, v \in V$, SE ESISTE UN CAMMINO DA u A v IN G.

- LA RELAZIONE $\sim$ E' UNA RELAZIONE DI EQUIVALENZA SU V

LE COMPONENTI CONNESSE DI G SONO LE CLASSI DI EQUIVALENZA DI $\sim$, ■

ES. SIA G IL SEGUENTE GRAFO:

- $\{a,b,c,d\}$, $\{e,f,g\}$, $\{h,i\}$, $\{j\}$ SONO LE COMPONENTI CONNESSE DEL GRAFO G

```
CONNECTED_COMPONENTS (G)
 for  v ∈ V[G] do
     Make_Set(v)
 for (u,v) ∈ E[G] do
     if  Find_Set(u) ≠ Find_Set(v) then
         Union (u,v)

SAME_COMPONENT (u,v)
 if  Find_Set(u) = Find_Set(v) then
     return  true
 else
     return  false
```

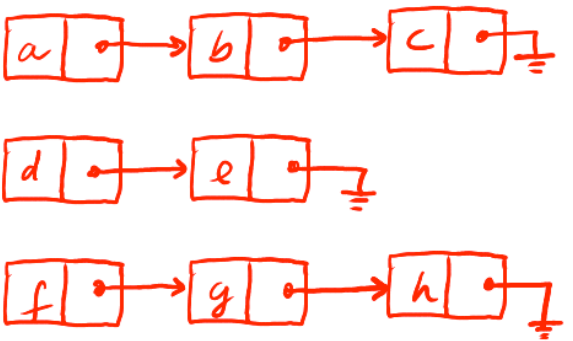
ESERCIZIO

- SIA $G=(V,E)$ UN GRAFO CON k COMPONENTI CONNESSE.
- QUANTE CHIAMATE A FIND_SET CI SONO NEL CORSO DELL'ESECUZIONE DI CONNECTED_COMPONENTS (G) ?
- E QUANTE CHIAMATE A UNION ?

IMPLEMENTAZIONE DI INSIEMI DISGIUNTI CON LISTE CONCATENATE

I TENTATIVO

ES. $\{a,b,c\}$, $\{d,e\}$, $\{f,g,h\}$



IMPLEMENTAZIONE DELLE OPERAZIONI

MAKE_SET(x) : OVVIA

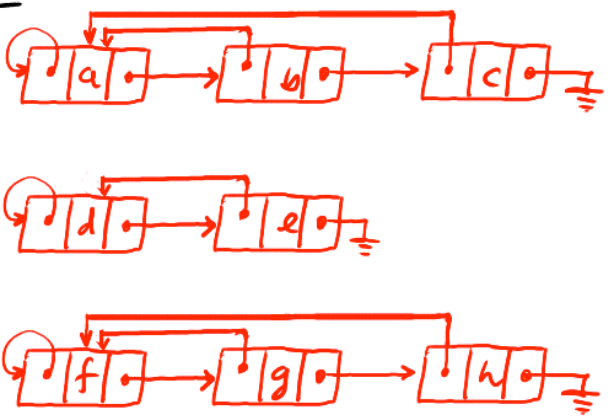
UNION(x,y) : ?

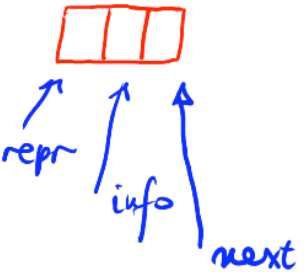
FIND_SET(z) : ?

- SERVE UN PUNTATORE DA CIASCUN ELEMENTO AL PRIMO ELEMENTO DELLA LISTA

II TENTATIVO

ES. $\{a,b,c\}$, $\{d,e\}$, $\{f,g,h\}$





IMPLEMENTAZIONE DELLE OPERAZIONI

MAKE_SET(x) : OVVIA

UNION(x,y) : MEDIANTE CONCATENAZIONE E SUCCESSIVO AGGIORNAMENTO DEI CAMPI repr

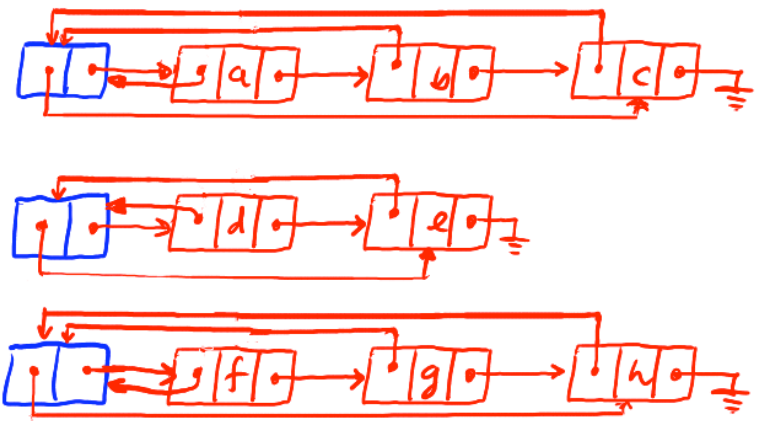
FIND_SET(x) : MEDIANTE CALCOLO DI repr[x]

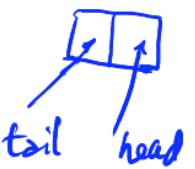
COME SEMPLIFICARE L'OPERAZIONE DI CONCATENAZIONE ?

- E' UTILE MANTENERE UN PUNTATORE ALL'ULTIMO ELEMENTO DI CIASCUNA LISTA (E CONVENIRE DI CONCATENARE LA SECONDA LISTA ALLA PRIMA, DURANTE L'ESECUZIONE DI UN'OPERAZIONE DI UNION)

III TENTATIVO

ES. $\{a,b,c\}$, $\{d,e\}$, $\{f,g,h\}$





COMPLESSITA'

SI CONSIDERI LA SEQUENZA DI $2n-1$ OPERAZIONI:

MAKE_SET(x_1)
⋮
MAKE_SET(x_n)

n → $\Theta(n)$

UNION(x_2, x_1) → 1
UNION(x_3, x_2) → 2
UNION(x_4, x_3) → 3
⋮
UNION(x_n, x_{n-1}) → $n-1$

$n-1$

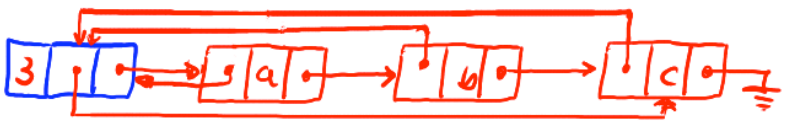
$\Theta(n^2)$

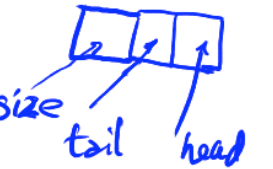
QUINDI CIASCUNA OPERAZIONE HA COSTO AMMORTIZZATO $\Theta(n)$

EURISTICA: NELL'ESEGUIRE UN'OPERAZIONE DI UNION, SI APPENDA LA LISTA PIÙ CORTA A QUELLA PIÙ LUNGA (UNIONE PER RANGHI)

IV TENTATIVO

ES. $\{a,b,c\}$, $\{d,e\}$, $\{f,g,h\}$





COMPLESSITA'

- SIA DATA UNA SEQUENZA $\mathcal{S}$ DI OPERAZIONI MAKE_SET, UNION E FIND_SET E PONIAMO:

  n = # MAKE_SET

  m = # MAKE_SET + # UNION + # FIND_SET

  k = # AGGIORNAMENTI AI CAMPI repr

- LA COMPLESSITA' DELLA SEQUENZA $\mathcal{S}$ E' CHIARAMENTE $\Theta(m + k)$

- PROVIAMO A VALUTARE k IN FUNZIONE DI n
- SIA x UN NODO. QUANTE VOLTE PUÒ CAMBIARE IL VALORE $repr[x]$?
- PER L'EURISTICA DELL' UNIONE PER RANGHI, OGNI QUALVOLTA IL VALORE $repr[x]$ E' AGGIORNATO, LA LUNGHEZZA DELLA LISTA CONTENENTE x AUMENTA DI ALMENO 2 VOLTE.
  QUINDI IL CAMPO $repr[x]$ PUÒ CAMBIARE AL PIÙ $\lfloor \log n \rfloor$ VOLTE
- DI CONSEGUENZA IL NUMERO TOTALE DI AGGIORNAMENTI AL CAMPO $repr$ DI TUTTI I NODI E' $O(n \log n)$
- PERTANTO LA COMPLESSITA' DELLA SEQUENZA S E' $O(m + n \log n)$

ESERCIZI

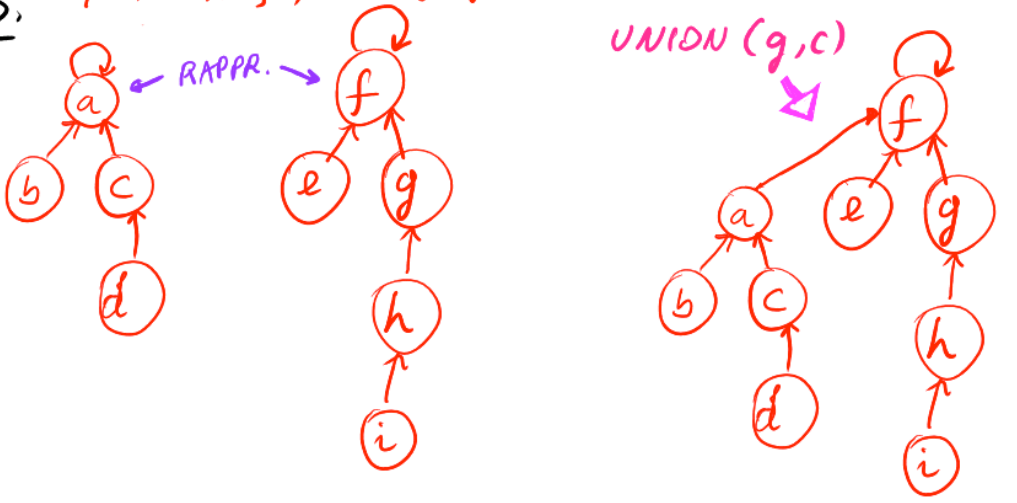
- E' POSSIBILE FARE A MENO DEL CAMPO tail ?

- SI DIMOSTRI CHE CON L'IMPLEMENTAZIONE MEDIANTE LISTE CONCATENATE ED UNIONE PER RANGHI SI HANNO I SEGUENTI COSTI AMMORTIZZATI

  MAKE_SET → $O(1)$
  FIND_SET → $O(1)$
  UNION → $O(\lg n)$

RAPPRESENTAZIONE DI INSIEMI DISGIUNI MEDIANTE FORESTE DI ALBERI DISGIUNTI

ES. $\{a,b,c,d\}, \{e,f,g,h,i\}$



- L'IMPLEMENTAZIONE CON ALBERI DISGIUNTI NON E' PIÙ EFFICIENTE DI QUELLA CON LISTE CONCATENATE, A MENO CHE NON SI UTILIZZINO LE EURISTICHE DI

  - UNIONE PER RANGHI
  - COMPRESSIONE DEI CAMMINI

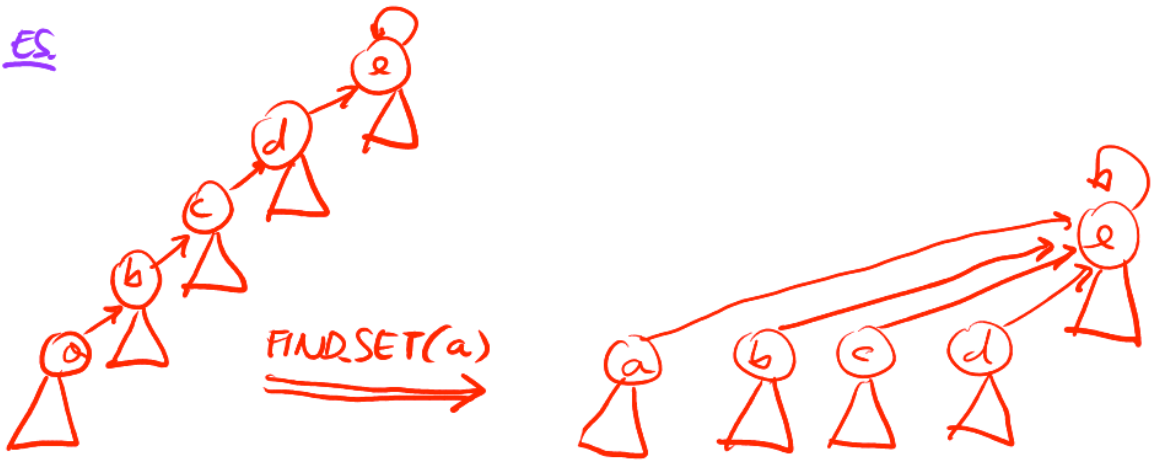
UNIONE PER RANGHI

- SI DEFINISCE OPPORTUNAMENTE LA NOZIONE DI RANGO DI UN ALBERO (LEGATA IN QUALCHE MODO ALL'ALTEZZA)
- QUINDI SI SCEGLIE DI INNESTARE L'ALBERO DI RANGO PIÙ PICCOLO NELLA RADICE DELL'ALBERO DI RANGO PIÙ ALTO
- NEL CASO DI ALBERI AVENTI LO STESSO RANGO, GLI ALBERI SI INNESTANO IN MANIERA ARBITRARIA E IL RANGO DELL'ALBERO RISULTANTE VIENE INCREMENTATO DI UN'UNITÀ.

COMPRESSIONE DEI CAMMINI

- TUTTI I NODI INCONTRATI NELL'ESECUZIONE DI UNA FIND_SET VENGONO INNESTATI NELLA RADICE

ES.



```
MAKE_SET(x)
  p[x] := x
  rank[x] := 0
```

```
LINK(x, y)
  if rank[x] > rank[y]
    then p[y] := x
    else p[x] := y
         if rank[x] = rank[y]
           then rank[y] := rank[y] + 1
```

```
UNION(x, y)
  LINK(FIND_SET(x), FIND_SET(y))
```

```
FIND_SET(x)
  if x ≠ p[x] then
    p[x] := FIND_SET(p[x])
  return p[x]
```

EFFETTI DELLE EURISTICHE SULLA COMPLESSITA'

- L'UNIONE PER RANGHI (SENZA COMPRESSIONE DEI CAMMINI) DA' LUOGO AD UNA COMPLESSITA' $O(m \lg n)$
- LA COMPRESSIONE DEI CAMMINI (SENZA UNIONE PER RANGHI) DA' LUOGO AD UNA COMPLESSITA' $\Theta(n + f \cdot (1 + \log_{2+f/n} n))$, DOVE $f = \#\,FIND\_SET$
- UTILIZZANDO ENTRAMBE LE EURISTICHE, SI OTTIENE UNA COMPLESSITA' $O(m\alpha(n))$, DOVE $\alpha(n)$ E' UN'INVERSA DELLA FUNZIONE DI ACKERMANN
- IN PRATICA $\alpha(n) \leq 4$, SEBBENE $\alpha(n) \longrightarrow +\infty$.