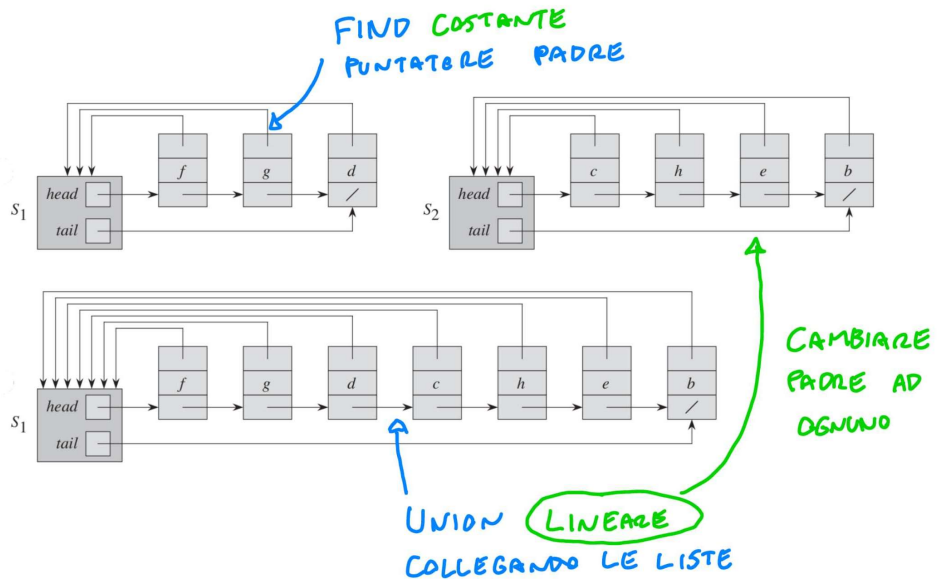
Insiemi disgiunti: dati

Ogni elemento fa parte di uno e un solo insieme

- Dobbiamo memorizzare un insieme di n elementi $S = \{s_1, s_2, \ldots, s_n\}$ suddivisi in k sottoinsiemi $S_1, S_2, \ldots, S_k$ disgiunti
- $S = S_1 \cup S_2 \cup \cdots \cup S_k$
- $S_i \cap S_j = \emptyset$

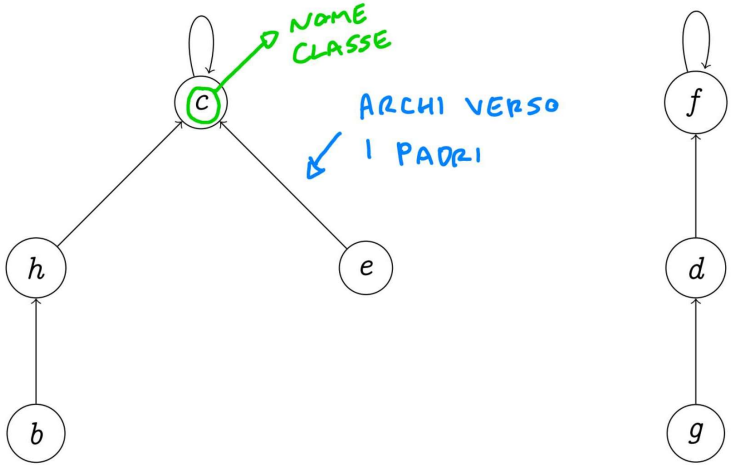
Insiemi disgiunti implementati con linked list



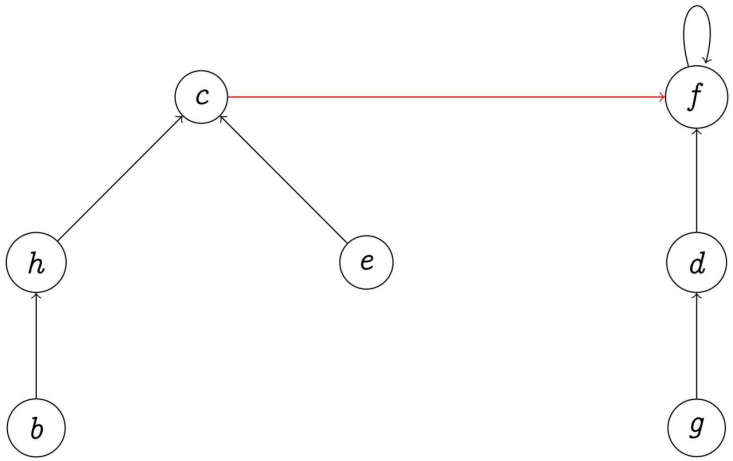
Insiemi disgiunti: operazioni

- Make-Set: creare un sottoinsieme con dentro un solo elemento
- Find: trovare il sottoinsieme S_i a cui appartiene un dato elemento s_j
- Union: dati S_i e S_j creare $S_i \cup S_j$
- Viste le operazioni supportate questa struttura dati viene spesso chiamata Find-Union

Esempio di up-trees

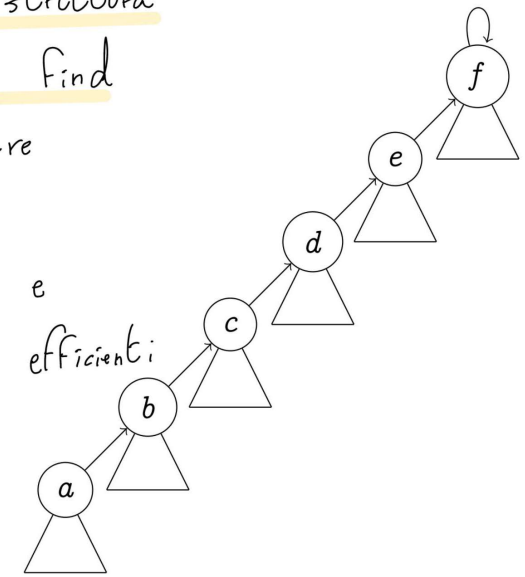


Esempio di unione di up-trees

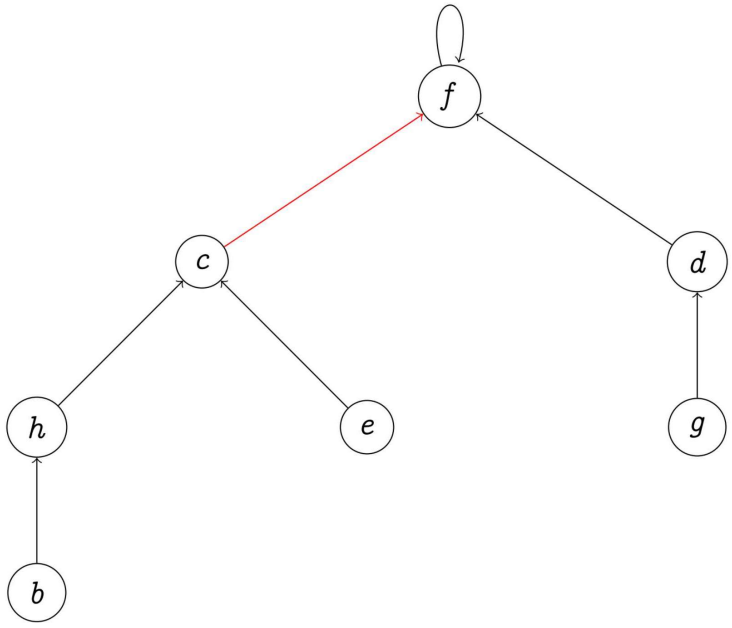


Compressione dei cammini

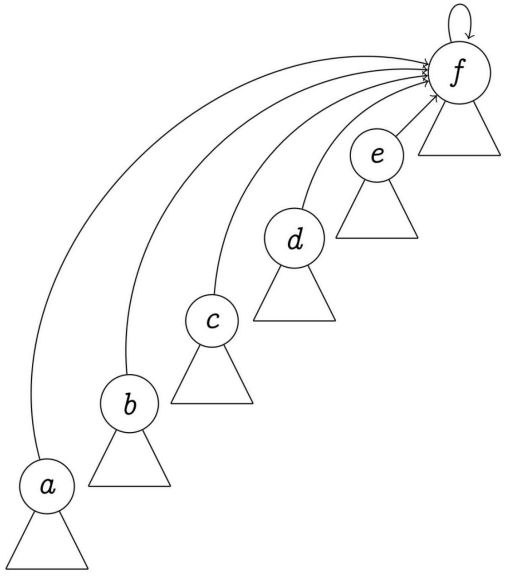
Sistemo la struttura durante la find senza variare l'ordine di grandezza e rendendo più efficienti le successive



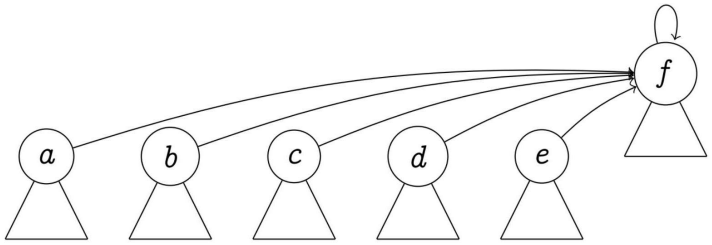
Esempio di unione di up-trees

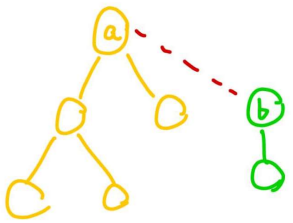


Compressione dei cammini



Compressione dei cammini





Attaccando gli insiemi con altezza (rango) minore a quelli con rango maggiore, non aumentiamo il rango totale

Make-Set e Link

```
MAKE-SET(x)
1  x.p = x
2  x.rank = 0

LINK(x, y)
1  if x.rank > y.rank
2        y.p = x
3  else x.p = y
4       if x.rank == y.rank
5            y.rank = y.rank + 1
```

UNICO CASO IN CUI AUMENTA IL RANGO

Union e Find-Set

```
UNION(x, y)
1  LINK(FIND-SET(x), FIND-SET(y))

FIND-SET(x)
1  if x ≠ x.p        x == x.p ⇐⇒ ROOT
2       x.p = FIND-SET(x.p)        COMPRESSIONE
3  return x.p
```

NON VIENE AGGIORNATO IL RANGO PER MANTENERE IL COSTO

GUADAGNO RANGO PRECISO < GUADAGNO COMPRESSIONE

Definizione della funzione $\log^*$

"numero di logaritmi da applicare per attivare a 0"

$$T(n) = \begin{cases} 1 & n < 1 \\ T(\log(n)) + 1 & \end{cases} = \Theta(\log^*(n))$$

```
LOG-STAR(n)
1  if n < 1
2       return 0
3  else return(LOG-STAR(log_2(n)) + 1)
```

ORDINE DI GRANDEZZA MIGLIORE POSSIBILE

Analisi ammortizzata del costo computazionale

CALCOLO COSTO PER RIUTILIZZO NEL TEMPO

Costo delle Find-Union

Una sequenza di m operazioni Make-Set, Union e Find-Set, di cui n sono operazioni Make-Set, può essere eseguita su una foresta di up-tree con unione per rango e compressione di cammini in tempo $O(m \log^*(n))$