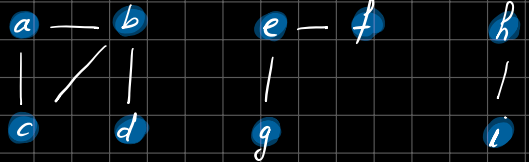
→ STRUTTURE DATI PER INSIEMI DISGIUNTI:

* Abbiamo una collezione di insiemi disgiunti, su cui abbiamo una serie di istruzioni disponibili:

1. MAKESET (x)
2. UNION (x, y) → x e y sono elementi di 2 insiemi disgiunti.
3. FINDSET (x) → restituisce l'insieme a cui appartiene x.

APPLICAZIONE DI QUESTE STRUTTURE:



* Vogliamo scrivere un algoritmo che "conti" il numero di componenti connesse.

connected_components (G):

// ogni v della mia collezione é un insieme disgiunto a sé

```
FOR v ∈ V:
    make_set(v);
// creo le componenti connesse
FOR (v_i, v_j) ∈ E                    // Per ogni arco del grafo
    IF FINDSET(v_i) ≠ FINDSET(v_j)    // se 2 vertici fanno parte di 2 insiemi disgiunti
        union(v_i, v_j)               // li unisco in uno stesso insieme
```

Cosa succede nella struttura dati:

a b c d e f g h i j

{a} {b} {c} {d} {e} {f} {g} {h} {i} {j} ← MAKESET

* ESPLORO GLI ARCHI ESISTENTI:

(b, d) → {b, d} ... (a, c) → {a, c} ... (a, b) → {a, b, c, d} (b, c) → già pres.

(e, g) → {e, g} ··· (h, i) → {h, i} ··· (e, f) → {e, g, f}

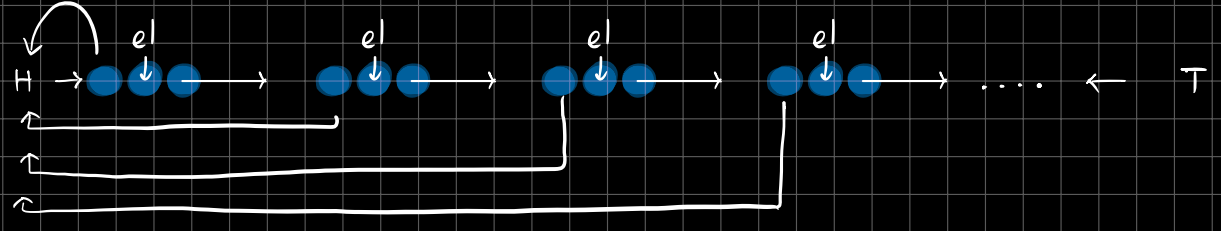
* **QUINDI RISULTA:**

{{a, c, b, d} {e, g, f} {h, i} {j}}

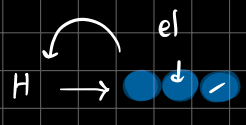
* **STRUTTURA DATI:**

Liste concatenate:

* evolvono nel tempo → ciascuna è un insieme disgiunto
- Sicuramente ogni elemento → è un nodo della linked-list.
- il rappresentante dell'insieme disgiunto → corrisponde alla head, perché è facilmente accessibile
- Per conoscere rapidamente il rappresentante → il "previous" di ogni nodo deve essere H.
- Per velocizzare la UNION → utilizzo un puntatore a TAIL



- MAKESET() → per ogni elemento del mio insieme disgiunto costruisco:



- FINDSET() → per ogni elemento della linkedlist → è una consulta al nodo "previous".

- UNION() → è una unione tra 2 linkedlist() utilizzando HEAD e PREVIOUS.

In questo modo: MAKESET $\Theta(1)$, FINDSET(x) $\Theta(1)$,

UNION() fa eccezione → Devo aggiornare il previous di ogni elemento della 2ª lista.

↳ $O(n)$
↳ cardinalità 2ª linkedlist.

* QUANTO TEMPO PUÓ RICHIEDERE AL MASSIMO L'IMPOSTAZIONE DI UN INS. DISG?

n = "numero totale di operazioni MAKE-SET";

m = " " MAKE-SET + UNION + FINDSET;

MAX. numero di union ⇒ $(n-1)$;

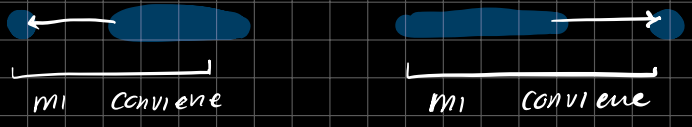
* massimo numero di operazioni (ragionevoli) che si possono fare: $\Theta(n^2)$

```
for i=1 to n:
    makeset(u)

for i=1 to n:
    union(x_i, x_{i+1});  → Θ(u)
```
} $O(n^2)$

* POSSO OTTIMIZZARLA?

Sì → posso guardare la cardinalità delle linkedlist.



* Per conoscere la cardinalità delle liste (?)

→ EURISTICA DELL'UNIONE PESATA:

* ogni lista ha un attributo lunghezza

* Union: la lista più corta viene aggiunta a quella più lunga.

→ Con euristica dell'unione pesata:

m operazioni (makeset + union + findset) possono essere fatte in $O(m + n \log n)$.

ESEMPIO:

* Consideriamo x → quante volte aggiorno $R(x)$? (nel caso peggiore) — $R(x)$ → rappresentante

1° VOLTA) Insieme risultante ha almeno 2 el.

2° VOLTA) Insieme risultante ha almeno 4 el.

3ª VOLTA) " " 8 el.

.....

i-esima) volta → " " 2^i el.

In totale, per n elementi → $n \cdot \log(n)$ aggiornamenti R.

* Tempo totale alg. $O(m + n \log n)$