Lezione 14 maggio

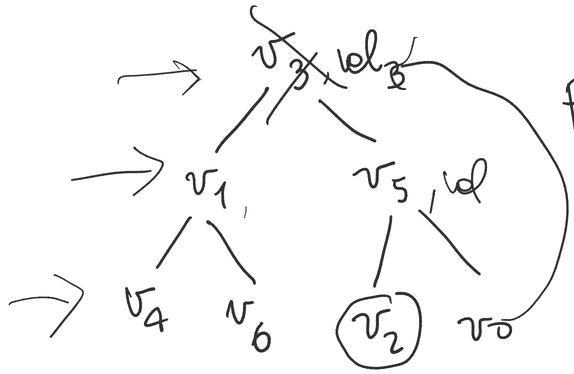
Thursday, May 14, 2020 9:09 AM

$\Theta(n \log n)$ n inserzioni in PQ nella inizializzazione

ciclo <u>while</u> è eseguito

m volte $\rightarrow$ DecreaseKey(PQ, u, dist[u])

in PQ $\Rightarrow O(n)$ ricerca

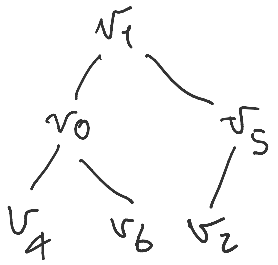


PQ

0	1	2	3	4	5	6
v_3	v_1	v_5	v_4	v_6	v_2	v_0

0	1	2	3	4	5	6
6	1	5	0	3	2	4

PosHeap



accesso al nodo i nella PQ

$PQ[PosHeap[i]] = \Theta(1)$

$\Theta(n)$ per costruirlo all'in

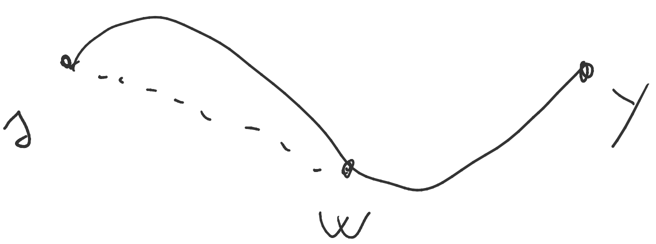
Dequeue
DecreaseKey } aggiornamento di PosHeap $\Rightarrow O(\log n)$

$$\Theta(n \log n + m \log n)$$

$$\Downarrow$$

$$\Theta(m \log n)$$

$$\Theta(n \log n + m) \to \text{Fibonacci Heap}$$



$s \sim w$

$w \sim y$

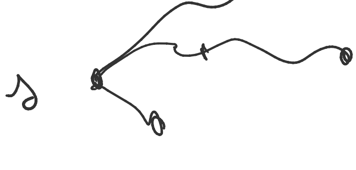
Bellman - Ford caso $W : E \to \mathbb{R}$

MINIMAL SPANNING TREE

MST

$G = (V, E, W)$ $\quad W : E \to \mathbb{R}^+$

$T = (V, E')$ $\quad E' \subseteq E : \sum_{e \in E'} w(e)$ min



Non c'è una sorgente selezionata

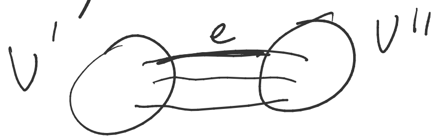
insieme di archi di peso minimo che connette i vertici del grafo e che <u>connette</u> i vertici del grafo e non abbia <u>cicli</u>

Se il grafo non è pesato?

Def. Taglio (Cut) è un sottoinsieme di archi $C \subseteq E$ la cui rimozione sconnette G.

Teo: Dato $G = (V, E, w)$ e $T = (V, E')$:

T è uno MST per G, $\forall e \in E$:



1) <u>condizione di taglio</u>:

<u>e</u> $\in E'$ sse $\exists$ un taglio in G che comprende <u>e</u> e <u>e</u> è l'arco di peso minimo.

2) <u>condizione di ciclo</u>

$e \notin E'$ sse $\exists$ un ciclo in G che comprende <u>e</u> e <u>e</u> è l'arco di peso max del ciclo.

- KRUSKAL
- <u>PRIM</u> - YARNIK

} greedy

. Esamina gli archi di G in ordine crescente di peso, $\forall (u, v)$

1) se u e v sono già collegati in E',

1) se u e v sono già collegati in E', (u, v) non viene scelto (chiuderebbe un ciclo.

2) se u e v non sono già collegati ($(u,v) \in C$ di cui è l'arco di peso minimo) (u,v) viene selezionato.

Si comincia da n nodi isolati (componenti connesse), poi ogni volta che si aggiunge un arco si uniscono componenti connesse. Finché si arriva a averne una sola.

1) PQ Minheap degli archi sul loro peso. Un elemento
arco, peso

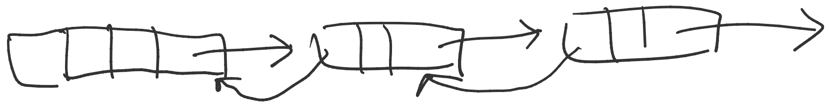
2) Struttura dati per le componenti.

Operazioni: SET $\Rightarrow$ dato un nodo x dice a quale comp. conn. appartiene
; UNION $\Rightarrow$ unisce 2 componenti connesse (quella di u e quella di v)

3) array SET: $\forall$ nodo dice a quale SET appartiene

SET appartiene

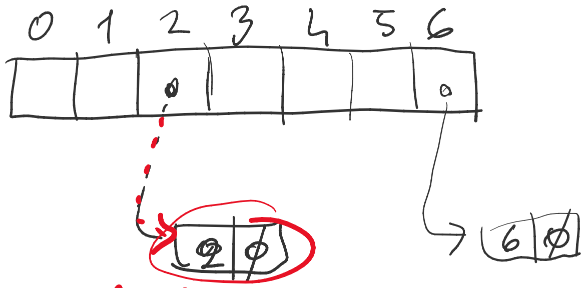
4) lista mst che memorizza gli archi della soluzione. (lista doppia)



Kruskal ( ):

```
for (u=0; u<n; u++){
    for (x=Adj[u]; x!=NULL; x=x.succ) {
        v= x.dato;
        elemento.dato = <u,v> ;        O(m logm)
        elemento.peso = x.peso;
        Enqueue (PQ, elemento)
    }
```

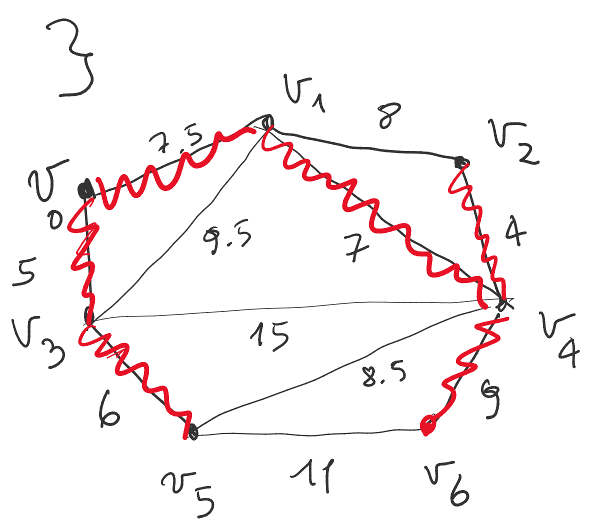
SET




```
    Set[u] = NuovoNodo()
    Crea(Set[u]);      Creo una lista di punt. Set[u]
} while (PQ ≠ ∅) {
    elemento = Dequeue(PQ); O(log m)
    <u,v> = elemento.dato
Θ(m)  if (set[u] ≠ set[v]) {   Θ(1)
Θ(n)      Unisci(set[u] e set[v]); O(log n)
```

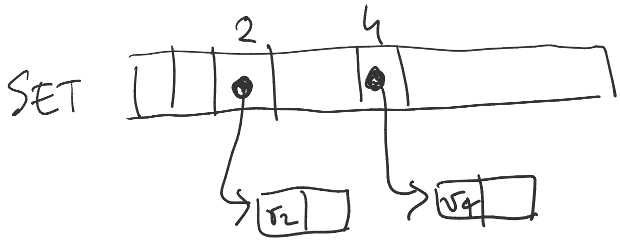
$\Theta(n)$ $\Big\{$ Unisci (set[u] e set[v]); $O(\log n)$
Inserisci (mst, $\langle u,v \rangle$); $\Theta(1)$
$\Big\}$

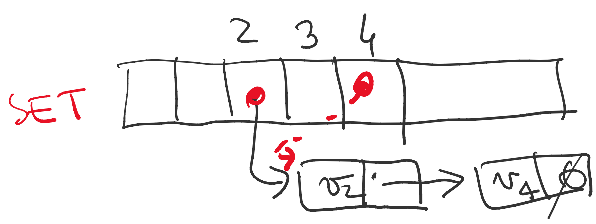
}



(v_2, v_4), 4 ✓
(v_0, v_3), 5 ✓
(v_3, v_5), 6 ✓
(v_1, v_4), 7 ✓
(v_0, v_1), 7.5 ✓
(v_1, v_2), 8 NO
(v_5, v_4), 8.5 NO
(v_4, v_6), 9 NO

(v_3, v_1), 9.5
(v_5, v_6), 11
(v_3, v_4), 15





UNIONE

$O(n)$ caso pessimo

ipotesi: ad ogni lista (associata a una componente connessa) sia associata anche la sua lunghezza.

Unione: tra 2 liste avviene copiando la più corta nella più lunga

$\Downarrow$

se considero un el. x qualsiasi

se considero un el. z raggiunge
in una lista che è almeno 2 volte
la lista iniziale
Poiché la lista finale ha al
più n elementi, quante volte
un el. z può cambiare lista.

$O(n)$ caso pessimo
$O(\log n)$ ammortizzato

totale $O(m \log m) + O(n \log n)$
$$\Downarrow$$
$$O((m+n)\log m) = O((m+n)\log n)$$