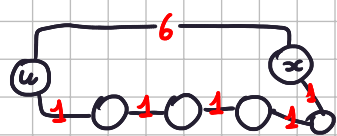


Esercizio 1. (max 5) Per un grafo G non diretto e connesso con pesi $w(e)$ sugli archi, abbiamo calcolato il albero di copertura di peso minimo (MST) T_1 e l'albero dei cammini minimi T_2 con radice u tramite l'algoritmo di Dijkstra. Si consideri ora il grafo con pesi nuovi incrementando il costo di ciascun arco di G per 1 - quindi $w'(e) = w(e) + 1$ per ogni arco e . Dimostrare o confutare ciascuna delle seguenti affermazioni:

- a) T_1 è un albero di copertura minimo anche per il grafo G con pesi $w'(e)$.
- b) T_2 è un albero dei cammini minimi con sorgente il nodo u anche per il grafo G con pesi $w'(e)$.

a) Supponiamo che T_1 sia cambiato, cio' significa che un arco ne ha sostituito 1, sia e_0 l'arco sostituito e e_n l'arco aggiunto. Se li considerassimo entrambi, si creerebbe un ciclo $C = \{e_0, e_1, e_2, \dots, e_n, e_n\}$, sappiamo che $w(e_0) < w(e_n) \Rightarrow w'(e_0) < w'(e_n)$, ma quindi $w'(C \setminus \{e_0\}) > w'(C \setminus \{e_n\})$, allora nel MST non ha senso considerare $e_n \Rightarrow T_1$ non cambia.

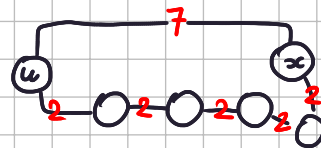
b) Falso, e' smentito dal seguente controesempio



$$\text{dist}(u, x) = 5$$

$$(u, x) \notin T_2$$

Nuovi pesi \Rightarrow



$$\text{dist}(u, x) = 7$$

$$(u, x) \in T_2$$

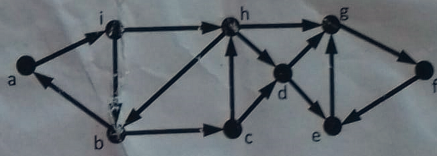
Esercizio 2. (max 5) Dare lo pseudo-codice di un algoritmo che preso in input il vettore dei padri P di un albero T con radice x e due vertici u e v , trova la distanza tra u e v in T . L'algoritmo deve avere complessità $O(n)$.

La distanza fra u e v e' uguale alla somma delle loro distanze dal primo antenato comune.

```

Es2( P: array lungo n, u: nodo, v: nodo ) {
    ca: int // common ancestor
    uA[n] = {1, 1, ..., 1} // antenati di u
    i = u
    d = 0
    while(P[i] != i) { // trovo gli antenati di u
        uA[i] = d // distanza fra u e l'antenato
        d++
        i = P[i]
    }
    d = 0
    i = v
    while(P[i] != i) { // visalgo antenati di v
        if(uA[i] != -1) {
            ca = i // trovo il primo ant. di v in comune con u
            break
        }
    }
    d++
    i = P[i]
}
return uA[ca] + d // d = dist(v, ca) e uA[ca] = dist(u, ca)
    
```

Esercizio 4. (max 5) Applicando l'algoritmo di Tarjan (l'algoritmo visto in lezione per trovare i componenti fortemente connessi in un grafo diretto) al grafo nella figura qui sotto, partendo dal nodo (a), determinare le componenti fortemente connesse numerandole nell'ordine in cui sono trovate dall'algoritmo e per ognuna indicare la sua c-root.



Esercizio 5 (max 8) Dato monete con valore 1, 5, 10, 25, dare il pseudocodice per un algoritmo greedy che da il resto per un valore N preso in input usando il minimo numero di monete. Dare la dimostrazione della correttezza dell'algoritmo.

```

Es5( $N: \text{int} > 0$ ) {
  Sol = { }
  M = {1, 5, 10, 25}
  while( $N \neq 0$ ) {
    K = {m ∈ M | m ≤ N}
    i = max(K)
    N = N - i
    Sol.add(i)
  }
  return Sol
}

```

Prop: $\text{Sol}^k \subseteq \text{Sol}^*$ dove $\text{Sol}^* :=$ qualsiasi soluzione ottimale

1 Dim: • $\text{Sol}^0 = \emptyset \subseteq \text{Sol}^*$ • ipotesi: $\text{Sol}^k \subseteq \text{Sol}^*$

• p.i. $\text{Sol}^{k+1} = \text{Sol}^k \cup \{m\}$ dove $m \leq N - \sum_{i \in \text{Sol}^k} i$. Se m non è considerato in Sol^*

2 Sia Sol^* una soluzione ottimale che ad un k -esimo passo non ha rispettato l'algo

$\Rightarrow \exists \{m_1, m_2, \dots, m_k\} \in \text{Sol}^* \mid \sum_{i=1}^k m_i = m$
 \uparrow valore da considerare nell'algoritmo

3 L'algo non viene rispettato se ad un certo punto:

• al posto di $m=25$ scelgo $M=\{m_1, \dots, m_k\} \mid \sum_{m_i \in M} m_i = 25 \Rightarrow$ la Sol non è minima

• al posto di $m=10$ scelgo $M=\{m_1, \dots, m_k\} \mid \sum_{m_i \in M} m_i = 10 \Rightarrow$ la Sol non è minima

• al posto di $m=5$ scelgo $M=\{m_1, \dots, m_k\} \mid \sum_{m_i \in M} m_i = 5 \Rightarrow$ la Sol non è minima