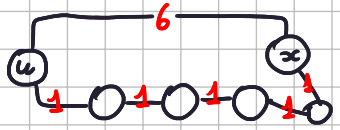
Esercizio 1. (max 5) Per un grafo G non diretto e connesso con pesi w(e) sugli archi, abbiamo calcolato il albero di copertura di peso minimo (MST) T_1, e l'albero dei cammini minimi T_2 con radice u tramite l'algoritmo di Dijkstra. Si consideri ora il grafo con pesi nuovi incrementando il costo di ciascun 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).

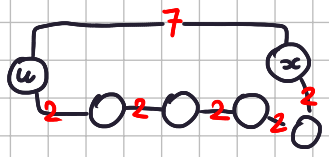
a) Supponiamo che T_1 sia cambiato, ciò significa che un arco ne ha sostituito 1, sia e_o l'arco sostituito e e_n l'arco aggiunto. Se li considerassimo entrambi, si creerebbe un ciclo $C = \{e_o, e_1, e_2 .., e_K, e_n\}$, sappiamo che $w(e_o) < w(e_n) \Rightarrow w'(e_o) < w'(e_n)$. ma quindi $w'(C /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

b) Falso, c' smentito dal seguente controesempio



dist(u,x) = 5
(u,x) ∉ T_2

Nuovi pesi ⇒



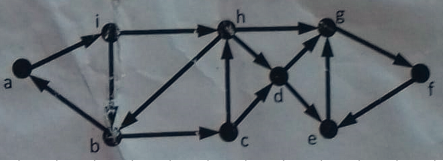
dist(u,x) = 7
(u,x) ∈ T_2

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){ // risalgo 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.