Cognome: **Nome**: **# Matricola**: **Riga**: **Col**:

Algoritmi e Strutture Dati – 01/09/14

Esercizio 1 – Punti ≥ 6 (Parte A)

Trovare un limite superiore alla seguente equazione di ricorrenza, utilizzando il metodo di sostituzione o per tentativi:

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 6T(n/8) + T(n/4) + 1 & n > 1 \end{cases}$$

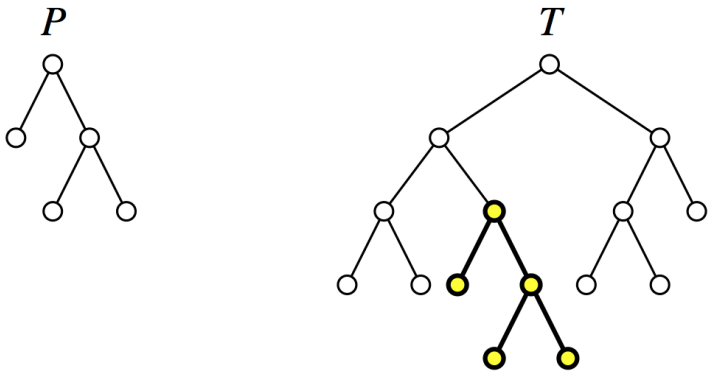
Esercizio 2 – Punti ≥ 6 (Parte B)

Disponiamo di un tubo metallico di lunghezza L. Da questo tubo vogliamo ottenere al più n segmenti, aventi rispettivamente lunghezza $S[1], S[2], \ldots, S[n]$. Il tubo viene segato sempre a partire da una delle estremità, quindi ogni taglio riduce la sua lunghezza della misura asportata. Ovviamente un tubo di lunghezza L non può dare origine a segmenti più lunghi di L.

1. Scrivere un algoritmo efficiente che, dati L e il vettore S, restituisca il numero massimo di segmenti che possono essere ottenuti dal tubo.
2. Dimostrare che tale algoritmo è corretto
3. Determinare il costo computazionale dell'algoritmo di cui al punto 1.

Esercizio 3 – Punti ≥ 9 (Parte A)

Dati due alberi binari P e T, scrivere un algoritmo che restituisca vero se P ha la stessa struttura di un sottoalbero di T. Ad esempio, nella figura seguente,



l'albero P ha la stessa struttura del sottoalbero evidenziato in T. Discutere la complessità dell'algoritmo risultante in termini di n_p (il numero di nodi contenuti in P) e n_t (il numero di nodi contenuti in T). Ricordate che è possibile aggiungere campi di supporto nei nodi dell'albero.

Esercizio 4 – Punti ≥ 12 (Parte B)

Si consideri un vettore V contenente n interi positivi. Supponete che all'inizio, un'ipotetica pedina sia posizionata nella posizione 1. Quando la pedina si trova in posizione i, $V[i]$ rappresenta il numero di massimo di passi che la pedina può fare in avanti a partire dall'elemento in cui ci si trova. Ad esempio, se la pedina si trova in posizione 1 e in $V[1]$ c'è il valore 4, la pedina può fare *una* mossa e raggiungere una qualunque delle posizioni $2 \ldots 5$. Il movimento finisce quando la pedina raggiunge esattamente la posizione n.
Scrivere un algoritmo che restituisca il numero minimo di *mosse* necessarie per raggiungere la posizione n a parte dalla posizione 1.

Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale.