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

Algoritmi e Strutture Dati - 03/05/13

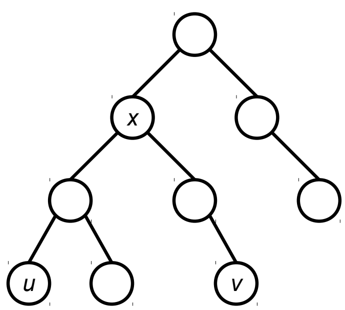
Esercizio 1 – Punti ≥ 7

Trovare un limite superiore e inferiore al costo computazionale del seguente algoritmo, dando una dimostrazione formale.

```
crazy(integer n)
---------------------------------------------
if n ≤ 1 then
    return n
else
    integer b ← 1
    for i ← 1 to n/2 do
        b ← (b · 2) mod 1007
    return b + crazy(⌊n/2⌋) + crazy(⌊n/3⌋)
```

Esercizio 2 – Punti ≥ 7

Si consideri un input formato da un albero binario T e due suoi nodi u e v. Si descriva un algoritmo che restituisca il più vicino antenato comune, ovvero il nodo che è antenato sia di u che di v e che abbia profondità massima (ovvero, sia il più vicino possibile ad entrambi i nodi). Ad esempio, nella figura x è il più vicino antenato comune di u e v. Analizzare la complessità computazionale dell'algoritmo proposto. Si descriva come il vostro algoritmo gestisce il caso in cui u è antenato di v o viceversa.



Esercizio 3 – Punti ≥ 7

Si consideri una griglia $G[0 \ldots n+1, 0 \ldots n+1]$, con $n > 1$. Ogni cella (i, j) della griglia può essere libera ($G[i, j] = 0$) oppure contenere un ostacolo ($G[i, j]$=1). Le celle sui bordi contengono ostacoli ($G[0, j]$=1, $G[n+1, j] = 1$, $G[i, 0] = 1$, $G[i, n+1] = 1$, per ogni $0 \leq i, j \leq n+1$). Un giocatore viene posto inizialmente nella casella $(1, 1)$. Ad ogni passo, il giocatore può muoversi in una delle caselle libere adiacenti. Specificamente, è possibile spostarsi dalla cella (i, j) ad una delle celle libere tra $(i-1, j)$, $(i+1, j)$, $(i, j-1)$ e $(i, j+1)$. Descrivere un algoritmo efficiente in grado di determinare il numero minimo di passi necessari per spostarsi dalla cella $(1, 1)$ alla cella (n, n) (che si assumono essere entrambe sempre libere), o $+\infty$ se non è possibile raggiungere (n, n) da $(1, 1)$. Analizzare la complessità computazionale dell'algoritmo proposto.

Esercizio 4 – Punti ≥ 12

Lungo un fiume ci sono n porti. A ciascuno di questi porti è possibile affittare una barca che può essere restituita ad un altro porto. E' praticamente impossibile andare controcorrente. Il costo dell'affitto di una barca da un punto di partenza i ad un punto di arrivo j, con $i < j$, è denotato con $C[i, j]$. E' possibile che per andare da i a j sia più economico effettuare alcune soste e cambiare la barca piuttosto che affittare un'unica barca. Se si affitta una barca in $k_1, k_2, k_3, \ldots, k_l$ (con $k_1 = 1, k_1 < k_2 < k_3 < k_l$) allora il costo totale è $C[k_1, k_2] + C[k_2, k_3] + \ldots + C[k_{l-1}, k_l] + C[k_l, n]$.
Scrivere un algoritmo che dato in input i costi $C[i, j]$, determini il costo minimo per recarsi da 1 ad n. Analizzare la complessità computazionale dell'algoritmo proposto. Per un punteggio aggiuntivo, si stampino i porti in cui devono essere noleggiate le barche.