Prof. L. Margara Scritto del 6/9/2023

Algoritmi e Strutture Dati

Esercizio 1.[11 punti]
Calcolare in ordine di grandezza il costo computazionale nel caso pessimo della funzione PARIDISPARI. Giustificare sempre tutte le risposte.

```
PARIDISPARI(n)
if n ≤ 2
    return 1
if n è pari
    return 2 * PARIDISPARI(n − 2)
if n è dispari
    return 3 * PARIDISPARI(n − 2)
```

Traccia Soluzione esercizio 1

Per qualunque valore di n la funzione richiama se stessa una sola volta su $n-2$ quindi l'equazione ricorsiva associata è:

$T(n) = T(n-2) + k =$
$T(n-4) + k + k =$
$T(n-6) + k + k + k =$
$\vdots$

E quindi $T(n) = \Theta(n)$

Esercizio 2.[11 punti]
Scrivere una funzione ricorsiva che prenda in input un albero binario di ricerca T (puntatore alla radice) e ritorni il numero di nodi di T (radice esclusa) che non hanno il fratello.
In ogni nodo sono memorizzati il puntatore al figlio sinistro e il puntatore al figlio destro. Non è presente (e non è possibile aggiungerlo) il puntatore al padre. Discutere la correttezza e il costo computazionale della funzione proposta.

Traccia della soluzione dell'esercizio 2.

```
CONTA-FIGLI-UNICI(p)
if p == NIL
    return 0
if p.left == NIL and p.right ≠ NIL
    return 1 + CONTA-FIGLI-UNICI(p.right)
if p.right == NIL and p.left ≠ NIL
    return 1 + CONTA-FIGLI-UNICI(p.left)
return CONTA-FIGLI-UNICI(p.left) + CONTA-FIGLI-UNICI(p.right)
```

Costo $\Theta(n)$.

Esercizio 3.[11 punti]
Sia $P = \{p_1, p_2, \ldots, p_n\}$ un insieme (disordinato) di n numeri interi positivi e T un numero intero positivo. Dare un algoritmo polinomiale (di costo minimo) per calcolare un sottoinsieme S di P di cardinalità minima tale per cui la somma dei numeri in S sia maggiore o uguale a T. Determinare se l'algoritmo proposto è ottimo oppure no. Calcolare il costo computazionale dell'algoritmo proposto.

Se la somma dei numeri in S dovesse essere uguale a T (non maggiore uguale) sareste comunque in grado di dare un algoritmo capace di fornire la soluzione ottima in tempo polinomiale ?

Traccia Soluzione esercizio.

```
F(v,T):
    w = Sort(v)
    s = 0
    n = length(w)
    sol = insieme vuoto
    while (n ≥ 1) and (s < T)
        s = s + w[n]
        sol = sol ∪ w[n]
        n = n − 1
    if s < T then return "nessuna soluzione"
    return sol
```

Costo computazionale: $T(n) = \Theta(n\ log(n))$ (dovuto all'ordinamento)

Se la somma dei numeri in S dovesse essere uguale a T il problema cambierebbe radicalmente e nessun algoritmo conosciuto riuscirebbe a risolvere il problema in tempo polinomiale.