Esame scritto 22 giugno 2021

Esercizio 1 (10 punti): Si consideri la seguente funzione:

```
funzione exam(n, k:  interi)
   prod ← 1
   i ← k
   while i ≥ 1 do
      prod ← prod *i
      if i pari then i ← i/2
      else i ← (i − 1)/2
   if n = 1 return prod
   return prod* exam(n − 1, k)
```

$\Theta(1)$

Si imposti la relazione di ricorrenza che ne definisce il tempo di esecuzione facendo particolare attenzione al caso base; si giustifichi l'equazione ottenuta, e la si risolva usando il metodo iterativo ed il metodo di sostituzione, commentando opportunamente i passaggi del calcolo.

metodo iterativo

$$T(n,k) = T(n-1,k) + \Theta(\log_2(k)) \qquad T(1) = \Theta(\log_2(k))$$

$$T(n,k) = \left[T(n-2,k) + \Theta(\log_2(k))\right] + \Theta(\log_2(k))$$

$$T(n,k) = T(n-J,k) + \Theta\left(\sum_{i=0}^{J-1} \log_2(k)\right) \qquad \text{FINCHE } J = n-1$$

$$T(n,k) = \Theta(\log_2(k)) \cdot [n-1] + \Theta\left(\sum_{i=0}^{n-2} \log_2(k)\right)$$

$$T(n,k) = \Theta(n\log(k)) + \Theta([n-2]\log(k)) = \Theta(n \cdot \log(k))$$

metodo di sostituzione

IPOTIZZIAMO CHE $T(n,k) = O(n\log(k))$

$* \; T(n) = T(n-1) + ck$

$T(1) = dk$

QUINDI CHE $T(n,k) \leq J n \log(k)$

CASO BASE $T(1,k) = \log(k)$

$$dk \leq J\log(k) \quad \checkmark \quad \forall J > 1$$

IPOTESI INDUTTIVA

$$\forall m < n \rightarrow T(m,k) \leq J m \log(k)$$

PASSO INDUTTIVO

$$T(n,k) \leq J n \log(k)$$

$$\underline{T(n-1,k)} + ck \leq J n \log(k) \quad \leftarrow \text{IPOTESI INDUTTIVA}$$

$$J(n-1)\log(k) + cnk \leq J n \log(k)$$

$$J n \cancel{\log(k)} - J\log(k) + cnk \leq J n \cancel{\log}(k)$$

$$cnk \leq J\log(k) \quad \checkmark \ \forall J > 1$$

ADESSO ANALOGAMENTE, DIMOSTRO $T(n,k) = \Omega(n\log(k))$

$T(n,k) \geq J n \log(k)$

CASO BASE

$$dk \geq J\log(k) \quad \forall J < 1 \ \checkmark$$

I. INDUT. $\forall m < n \rightarrow T(m,k) \geq J m \log(k)$

P. IND.

$$T(n,k) \geq J n \log(k) \quad \text{ma} \quad T(n,k) = T(n-1,k) + cnk$$

$$J(n-1)\log(k) + cnk \geq J n \log(k)$$

$$J n \log(k) - J\log(k) + cnk \geq J n \log(k)$$

$$cnk \geq J\log(k) \quad \checkmark$$

DATO CHE

$$T(n) = O(n\log(k)) \text{ e } T(n) = \Omega(n\log(k))$$

allora $T(n) = \Theta(n\log(k))$

Esercizio 2 (10 punti): Data una sequenza S di n bit memorizzata in un array A, progettare un algoritmo che ordina S in tempo $\Theta(n)$. L'algoritmo deve ordinare *in loco*.

Dell'algoritmo proposto si dia la descrizione a parole, si scriva lo pseudocodice e si calcoli il costo computazionale.

Per quale motivo è possibile ordinare S in tempo lineare?

ESSENDO CHE DEVO ORDINARE UNA SEQUENZA S DI n bit, ED ESSENDO i bit 0 O 1, QUINDI INTERI CHE RIENTRANO IN UN INTERVALLO NOTO, HO TUTTE LE IPOTESI NECESSARIE PER POTER UTILIZZARE UN COUNTING SORT, CHE SI RICORDI AVERE COSTO $\Theta(n)$. PERÒ, ESSENDO CHE n NON AVRÀ VALORI ALL'INFUORI DI 0 O 1, ED ESSENDO $1 > 0$, POSSO SEMPLICEMENTE SPOSTARE TUTTI GLI 0 A SINISTRA DEGLI 1, AVENDO COMUNQUE UNA SEQUENZA RISULTANTE ORDINATA.

pseudo codice:

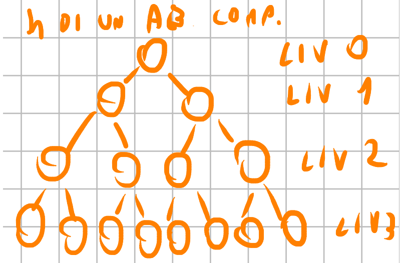
```
DEF ORDINA(S):
    INT i = 0;                      Θ(1)
    INT J = 0;                      Θ(1)
    WHILE(i <= LEN(S) - 1):         len(S)-1 VOLTE => n VOLTE
        IF(A[i] == 0):              Θ(1)
            INT TMP = A[J];         Θ(1)
            A[J] = A[i];            Θ(1)
            A[i] = TMP;             Θ(1)
            J += 1;                 Θ(1)
        i += 1;                     Θ(1)
    RETURN S;                       Θ(1)
```

TALE ALGORITMO SCORRE TUTTI I VALORI DI S, ED OGNI QUALVOLTA TROVA UNO 0, LO SCAMBIA CON IL VALORE NELL'INDICE J, RAPPRESENTANTE L'INDICE PIÙ A SINISTRA DELLA SEQUENZA NON ORDINATA.

COSTO $\Rightarrow$ $\Theta(1) + n\,\Theta(1) = \Theta(n)$

Esercizio 3 (10 punti): Dato un albero binario T, radicato nel nodo r, definiamo *altezza minimale* di T la minima distanza (cioè il minimo numero di archi) tra r e una qualsiasi foglia di T.

Progettare un algoritmo che, dato il puntatore alla radice di un albero binario memorizzato tramite record e puntatori, restituisca la sua altezza minimale.
Il costo dell'algoritmo deve essere $O(n)$, dove n è il numero di nodi dell'albero.
Dell'algoritmo proposto si dia la descrizione a parole, si scriva lo pseudocodice e si motivi il costo computazionale.
Quali sono i valori minimo e massimo che l'altezza minimale di T può assumere? Motivare la risposta.



$h = \log(n+1) - 1$

SCRIVERÒ UN ALGORITMO RICORSIVO, CHE VISITERÀ OGNI NODO DELL'ALBERO TENENDO CONTO DELLA DISTANZA, NON APPENA SI INCOMBERÀ IN UNA FOGLIA, SI RITORNERÀ TALE DISTANZA.

pseudo codice

OR INTESO COME OR LOGICO

```
DEF ES3(T, DIST);
    IF (T->PARENT == NULL):
        DIST = 0;        // SIAMO SULLA RADICE
    IF(((T->RIGHT) OR (T->LEFT)) == NULL): // SE NON HA FIGLI
        RETURN DIST;
    IF(T->RIGHT != NULL AND T->LEFT == NULL):
        RETURN ES3(T->RIGHT, DIST+1);
    IF(T->LEFT != NULL AND T->RIGHT == NULL):
        RETURN ES3(T->LEFT, DIST+1);
    RETURN MIN(ES3(T->LEFT, DIST+1), ES3(T->RIGHT, DIST+1));
```

SEMPLICEMENTE, AD OGNI PASSO RICORSIVO SI CONTROLLANO I FIGLI DEL NODO CORRENTE, È OVVIO CHE, NEL CASO PEGGIORE SI CONTROLLA OGNI NODO DELL'ALBERO, $T(n) = O(n)$.

NEL MIGLIOR CASO, UN FIGLIO DELLA RADICE È FOGLIA, L'ALTEZZA MINIMALE SARÀ 1.

NEL CASO PEGGIORE, L'ALBERO È COMPLETO, E L'ALTEZZA MINIMALE SARÀ $\log_2(n+1) - 1$