

Introduzione agli Algoritmi

Esame Scritto a canali unificati con spunti per la soluzione

docenti: T. Calamoneri, A. Monti

Sapienza Università di Roma

3 Novembre 2025

Esercizio 1 (10 punti): Nella funzione riportata qui sotto, i valori delle costanti x, y e z sono determinati dalla propria *matricola* e corrispondono rispettivamente **alla prima, seconda e quinta cifra** che compaiono nella propria matricola.

```
def es1(n):
    if n <= 1:
        return 0
    s = 1
    if x%2==1: x = x+1
    else: x = x+2
    for i in range(x):
        s = s*n
    tot = 0
    while tot*tot < s :
        tot += 1
    for i in range (y+1):
        tot += (z+1)*es1(n//x)
    return tot
```

Si sostituiscano i parametri x , y e z con i valori corrispondenti dati dalla propria matricola e si risponda alle seguenti domande:

1. Qual è l'equazione di ricorrenza che descrive il costo computazionale della funzione $es1(n)$? Perché?
2. Risolvere l'equazione di ricorrenza con il metodo dell'albero, dandone la rappresentazione grafica e dettagliando i calcoli.
3. Verificare la correttezza della soluzione con un altro metodo.

L'esercizio chiede di sostituire al posto dei parametri x , y , z i valori relativi alla propria matricola. Qui si fanno considerazioni generali affinché ciascuno possa dedurre la propria soluzione.

- **La funzione trasforma x nel primo pari che lo segue; sia x_p il valore trasformato ($x+1$ se x è inizialmente dispari ed $x+2$ se x è inizialmente pari).**
- **Il primo ciclo for viene eseguito $\Theta(x_p)$ volte ed, alla fine, s vale n^{x_p} .**
- **Il ciclo while viene eseguito $\Theta(\sqrt{s}) = \Theta(n^{\frac{x_p}{2}})$ volte; si osservi che la divisione per 2 all'esponente non crea problemi perché x_p è pari.**
- **il secondo ciclo for viene eseguito $y+1$ volte e richiama ricorsivamente la funzione su un input di taglia $\frac{n}{x_p}$.**

Ne segue la seguente relazione di ricorrenza:

$$T(n) = (y+1)T\left(\frac{n}{x_p}\right) + \Theta(n^{\frac{x_p}{2}}), \text{ con caso base } T(0) = T(1) = \Theta(1).$$

Si omette qui la soluzione tramite metodo dell'albero, la cui raffigurazione sarà un albero $(y + 1)$ -ario. Si può infine scegliere uno degli altri metodi per verificare la correttezza della soluzione trovata.

Esercizio 2 (10 punti): Si consideri la struttura dati Heap. Come prima cosa, si scelga se lavorare con un **min-heap** oppure con un **max-heap**. La scelta deve essere chiarita all'**inizio** della soluzione.

1. Inserimento:

A partire da un heap vuoto, si inseriscano **una alla volta** le prime **5 cifre della propria matricola**, seguendo l'ordine in cui compaiono.

Dopo **ogni inserimento**, si rappresenti graficamente lo stato dell'heap sotto forma di **albero binario** e si spieghi brevemente la ragione della configurazione ottenuta.

2. Estrazione:

Una volta completati tutti gli inserimenti, si svuoti l'heap effettuando **5 estrazioni successive** di massimo (se si è scelto di lavorare su un max-heap) o di minimo (se si è scelto di lavorare su un min-heap), una per volta.

Dopo **ogni estrazione**, si rappresenti nuovamente lo stato dell'heap come **albero binario** e si spieghi brevemente la ragione della configurazione ottenuta.

Come esempio di soluzione consideriamo di costruire un min-heap e di avere la matricola 2736415. I casi di max-heap e di altre matricole si risolvono in modo analogo.

In figura 1 si riporta da sinistra a destra la crescita dell'heap con l'inserimento delle chiavi 2, 7, 3, 6 e 4. La giustificazione

delle figure discende direttamente dall'algoritmo di inserimento in un heap e pertanto non viene qui dettagliata. In figura 2 si riporta la riduzione dell'heap a seguito dell'estrazione dei minimi, e cioè di 2, 3, 4 e 6. Anche in questo caso non viene dettagliata la giustificazione delle figure, che discende dall'algoritmo di estrazione del minimo.

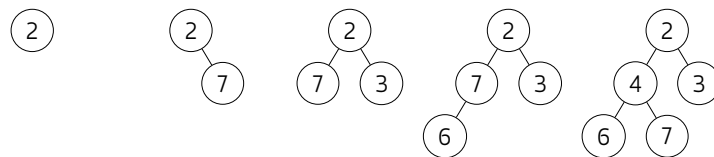


Figura 1: Heap dopo ciascun inserimento (prime 5 cifre)

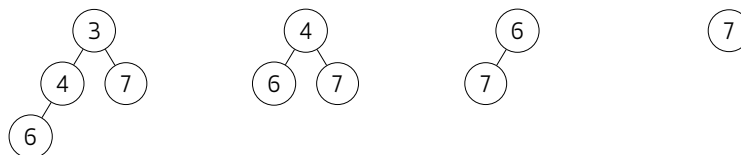


Figura 2: Heap dopo ciascuna estrazione

Esercizio 3 (10 punti): Sia data una lista concatenata memorizzata tramite record e puntatori; ogni nodo contiene un campo *val*, che memorizza una delle 10 cifre decimali, ed un puntatore *next* al nodo successivo nella lista.

Si scriva un algoritmo **ricorsivo** che, data la testa della lista *p* ed un array *mat* contenente le 7 cifre della vostra matricola, restituisca la nuova lista ottenuta eliminando tutti i nodi il cui valore coincide con una qualsiasi delle cifre presenti in *mat*. L'algoritmo deve avere costo computazionale $O(n)$ dove n è il numero di nodi della lista.

Dell'algoritmo proposto:

- a) si descriva a parole il funzionamento dell'algoritmo.
- b) si scriva lo pseudocodice,
- b) si giustifichi il costo computazionale richiesto, risolvendo formalmente la ricorrenza associata.

NOTA BENE: La funzione proposta **non** deve far uso di variabili globali.

- a. **L'idea dell'algoritmo è semplice: se la lista è vuota, si restituisce *None*. Altrimenti, si consideri *p*, il puntatore al nodo di testa. Si applica ricorsivamente la funzione alla lista *p.next*, ottenendo come risultato la lista *q* già filtrata. Se il valore *p.val* è una delle cifre presenti nella matricola (*mat*), il nodo viene escluso e si restituisce direttamente *q*. Altrimenti, si collega il nodo *p* alla lista *q* (impostando *p.next* = *q*) e si restituisce *p*.**
- b. **Ecco un possibile pseudocodice:**

```
def es3(p, mat)
    if p is None:
        return None
    q = es3(p.next, mat)
    if p.val in mat:
        return q
    p.next = q
    return p
```

- c. **Il costo computazionale è quello di una visita in cui ogni nodo richiede lavoro costante. L'operazione *in* per controllare se *p.val* appartiene a *mat* ha, in questo caso particolare, costo costante poiché *mat* contiene sempre**

7 elementi.

Il costo di una visita di questo tipo è

$$T(n) = T(n - 1) + O(1)$$

con $T(0) = \Theta(1)$ la cui soluzione è $O(n)$; l'equazione di ricorrenza va risolta con uno dei metodi studiati.