# Esercitazione

Corso di **Algoritmi e strutture dati**
Corso di Laurea in **Informatica**
Docenti: Ugo de'Liguoro, András Horváth



## Complessità asintotica

- **Definizione del** $O$ (limite superiore):

$$f(n) \in O(g(n)) \iff \exists c > 0, \exists n_0, \forall n > n_0 . f(n) \leq cg(n)$$

cioè, a parte un fattore moltiplicativo e un numero finito di casi (quindi asintoticamente), $f(n)$ cresce al massimo come $g(n)$



## Complessità asintotica

- **Definizione del** $\Omega$ (limite inferiori):

$$f(n) \in \Omega(g(n)) \iff \exists c > 0, \exists n_0, \forall n > n_0 . f(n) \geq cg(n)$$

cioè, a parte un fattore moltiplicativo e un numero finito di casi (quindi asintoticamente), $f(n)$ cresce almeno come $g(n)$



## Complessità asintotica

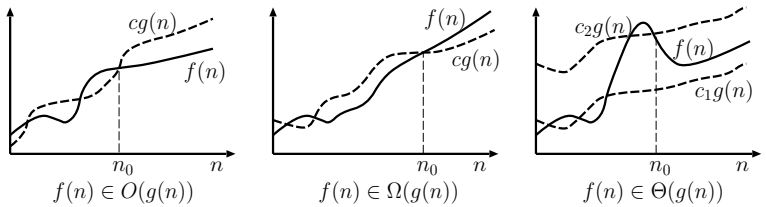
- **Definizione del** $\Theta$ (limiti stretti):

$$f(n) \in \Theta(g(n)) \iff \exists c_1 > 0, \exists c_2 > 0, \exists n_0, \forall n > n_0 . c_1 g(n) \leq f(n) \leq c_2 g(n)$$

cioè, a parte fattori moltiplicativi e un numero finito di casi (quindi asintoticamente), $f(n)$ cresce come $g(n)$

## Complessità asintotica

graficamente $O$, $\Omega$ e $\Theta$:






## Complessità asintotica

**Teorema:** Se il limite

$$a = \lim_{n\to\infty} \frac{f(n)}{g(n)}$$

esiste allora

$$0 \leq a < \infty \iff f(n) \in O(g(n))$$
$$0 < a < \infty \iff f(n) \in \Theta(g(n))$$
$$0 < a \leq \infty \iff f(n) \in \Omega(g(n))$$



## Complessità asintotica, esercizio 1

Si stabilisca, dimostrandolo formalmente, se le funzioni $2^{n-100}, 2^{\frac{1}{2}n}$ e $\left(\frac{3}{2}\right)^n$ siano $\Omega(2^n)$.

$2^{n-100}$ :

- $2^{n-100} = 2^n/2^{100}$ dove $1/2^{100}$ è "solo" un fattore moltiplicativo e dunque $2^{n-100} \in \Omega(2^n)$
- lo stesso si può dimostrare anche usando il teorema precedente

$$\lim_{n\to\infty} \frac{2^{n-100}}{2^n} = \frac{1}{2^{100}}$$

e dunque $2^{n-100} \in \Omega(2^n)$



## Complessità asintotica, esercizio 1

Si stabilisca, dimostrandolo formalmente, se le funzioni $2^{n-100}, 2^{\frac{1}{2}n}$ e $\left(\frac{3}{2}\right)^n$ siano $\Omega(2^n)$.

$2^{\frac{1}{2}n}$:

- usando il teorema precedente

$$\lim_{n\to\infty} \frac{2^{\frac{1}{2}n}}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2}^n}{2^n} = \lim_{n\to\infty} \left(\frac{\sqrt{2}}{2}\right)^n = 0$$

perché $\sqrt{2} < 2$ e dunque $2^{\frac{1}{2}n} \notin \Omega(2^n)$

## Complessità asintotica, esercizio 1

Si stabilisca, dimostrandolo formalmente, se le funzioni $2^{n-100}, 2^{\frac{1}{2}n}$ e $\left(\frac{3}{2}\right)^n$ siano $\Omega(2^n)$.

$\left(\frac{3}{2}\right)^n$:

- usando il teorema precedente

$$\lim_{n\to\infty} \frac{\left(\frac{3}{2}\right)^n}{2^n} = \lim_{n\to\infty} \left(\frac{\frac{3}{2}}{2}\right)^n = 0$$

perché $3/2 < 2$ e dunque $\left(\frac{3}{2}\right)^n \notin \Omega(2^n)$



## Complessità asintotica, esercizio 2

Si dimostri la seguente implicazione:

$$f_1(n) \in O(g_1(n)) \wedge f_2(n) \in O(g_2(n)) \implies f_1(n) + f_2(n) \in O(g_1(n) + g_2(n)).$$

Dimostrazione:

- $f_1(n) \in O(g_1(n))$ implica che $\exists c_1 > 0, \exists n_1, \forall n > n_1 . f_1(n) \leq c_1 g_1(n)$
- $f_2(n) \in O(g_2(n))$ implica che $\exists c_2 > 0, \exists n_2, \forall n > n_2 . f_2(n) \leq c_2 g_2(n)$
- con $n_0 = \max(n_1, n_2)$ e $c = \max(c_1, c_2)$

$$\forall n > n_0 . f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n) \leq c g_1(n) + c g_2(n) = c(g_1(n) + g_2(n))$$

e quindi $f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$



## Complessità asintotica, esercizio 3

Si dimostri che la seguente implicazione è falsa:

$$f_1(n) \in O(g_1(n)) \vee f_2(n) \in O(g_2(n)) \quad \implies \quad f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$$

- per dimostrare che l'implicazione sia falsa basta trovare un esempio con cui la "sinistra" è vere mente la "destra" è falsa
- con

$$f_1(n) = n^2, f_2(n) = n, g_1(n) = n, g_2(n) = n$$

è cosi



## Correttezza tramite invariante

L'**invariante di ciclo** è una proposizione sul valore delle variabili dell'algoritmo:

- *inizializzazione:* la proposizione vale immediatamente prima di entrare nel ciclo
- *mantenimento:* se la proposizione vale prima di eseguire il corpo del ciclo, allora vale anche dopo aver eseguito il corpo del ciclo
- deve essere *utile*: aiuta a dimostrare la correttezza dell'algoritmo

## Correttezza tramite invariante, esercizio

Dimostrare la correttezza del seguente algoritmo per calcolare il valore di un polinomio rappresentato dai suoi coefficienti $a_0, a_1, ..., a_n$ in un punto $x$.

HORNER($a_0, a_1, ..., a_n, x$)
  $y \leftarrow 0$
  $i \leftarrow n$
  **while** $i \geq 0$ **do**
    $y \leftarrow a_i + x \cdot y$
    $i \leftarrow i - 1$
  **return** $y$



## Correttezza tramite invariante, esercizio

| # esecuzioni del ciclo | $i$ | $y$ |
|---|---|---|
| 0 | $n$ | 0 |
| 1 | $n-1$ | $a_n$ |
| 2 | $n-2$ | $a_{n-1} + a_n x$ |
| 3 | $n-3$ | $a_{n-2} + a_{n-1}x + a_n x^2$ |
| 4 | $n-4$ | $a_{n-3} + a_{n-2}x + a_{n-1}x^2 + a_n x^3$ |
| $\vdots$ | $\vdots$ | |

La relazione fra $i$ e $y$ in ogni riga della tabella precedente, cioè l'invariante del ciclo, è

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k$$



## Correttezza tramite invariante, esercizio

All'uscita del ciclo si ha $i = -1$ e quindi

$$y = \sum_{k=0}^{n-(-1+1)} a_{k-1+1} x^k = \sum_{k=0}^{n} a_k x^k$$

cioè $y$ contiene il valore del polinomio nel punto $x$ e quindi l'algoritmo è corretto.



## Relazione di ricorrenza, esercizio

Si analizzi la complessità temporale in funzione di $n$ del seguente algoritmo, ricavando la relazione di ricorrenza e calcolandone il $\Theta$:

BUMP($n$)
  **if** $n \leq 1$ **then return** 5
  **else**
    $m \leftarrow 0$
    **for** $i \leftarrow 1$ **to** 8 **do**
      $m \leftarrow m +$ BUMP($\lfloor n/2 \rfloor$)
    **for** $i \leftarrow 1$ **to** $n$ **do**
      **for** $j \leftarrow 1$ **to** $n$ **do**
        $m \leftarrow m + 1$
    **return** $m$

## Relazione di ricorrenza, esercizio

- il corpo dei **for** annidati viene eseguito $n^2$ volte, mentre BUMP($\lfloor n/2 \rfloor$) viene eseguito 8 volte, si ottiene la ricorrenza:

$$T(n) = 8T(n/2) + n^2 \text{ se } n > 1, \quad T(1) = 1$$

- svolgendo:

$T(n) = 8T(n/2)+n^2 = 8(8T(n/2^2)+(n/2)^2)+n^2 = 8^2T(n/2^2)+2n^2+n^2 = \cdots$

da cui in generale, quando $k \leq \log_2 n$ (scritto semplicemente $\log n$ nel seguito)

- $$T(n) = 8^k T(n/2^k) + \left(\sum_{i=0}^{k-1} 2^i\right) n^2 = 8^k T(n/2^k) + (2^k - 1)n^2$$

- con $k = \log n$ e supponendo $T(1) = 1$ abbiamo

$$T(n) = 8^{\log n} + (2^{\log n} - 1)n^2 = 2n^3 - n^2 = \Theta(n^3)$$



## Analisi di un algoritmo, esercizio

Si consideri il seguente algoritmo:

```
AUGUSTA(A[1..n])
  for i ← n − 1 down to 1 do
      j ← i
      while j < n and A[j] < A[j + 1] do
          scambia A[j + 1] con A[j]
          j ← j + 1
```

Si risponda succintamente seguenti domande:

1. Cosa fa AUGUSTA($A[1..n]$)?
2. Si indichino l'invariante del ciclo esterno e l'invariante del ciclo interno.
3. Quali sono il caso peggiore e la sua complessità in termini di $\Theta$?
4. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?



## Analisi di un algoritmo, esercizio

- AUGUSTA ordina in senso non crescente il vettore $A[1..n]$ con una variante dell'INSERT-SORT
- invarianti

  esterno: $A[i + 1..n]$ è ordinato in senso non crescente
  interno: per ogni $h \in i..j - 1$, $A[h] \geq A[j]$
- il caso peggiore si verifica quando $A$ sia ordinato in senso crescente, e l'algoritmo è $\Theta(n^2)$
- una soluzione migliore è un'opportuna variante di un algoritmo $\Theta(n \log n)$, come il MERGE-SORT



## Conoscenza di una struttura dati, esercizio

- si consideri una tabella di hash ad indirizzamento aperto $T[0, ..., 19]$, con 20 elementi, che utilizza la seguente funzione di hashing

$$h(k, i) = (k \ mod \ 20 + 3i + 2i^2) \ mod \ 20$$

- si riporti il suo contenuto dopo l'inserimento delle seguenti chiavi: 15,40,35,20
- si riporti l'algoritmo di ricerca di una chiave in una tabella hash che usa l'indirizzamento aperto

## Conoscenza di una struttura dati, esercizio 1

Riportiamo dapprima l'algoritmo di inserimento (non richiesto nel testo):

```
HASH-INSERT(k, T[0..m − 1])
  for i ← 0 to m − 1 do
      j ← h(k, i)
      if T[j] ≠ nil then
          T[j] ← k
          return
  error overflow
```



## Conoscenza di una struttura dati, esercizio 1

- funzione hash: $h(k,i) = (k \ mod \ 20 + 3i + 2i^2) \ mod \ 20$
- nel caso dato $m = 20$
- all'inizio $T[j] = nil$ per ogni $j \in 0..19$
- quindi 15 viene inserito in $h(15,0) = 15$
- 40 viene inserito in $h(40,0) = 0$
- a questo punto $h(35,0) = 15$, ma $T[15] = 15$
- $h(35,1) = (15+5) \mod 20 = 0$, ma $T[0] = 40$
- infine $h(35,2) = (15+6+8) \mod 20 = 9$ onde 35 viene posto in $T[9]$
- infine $h(20,0) = 0$, ma $T[0] = 40$
- $h(20,1) = (3+2) \mod 20 = 5$, onde 20 viene inserito in $T[5]$



## Conoscenza di una struttura dati, esercizio 1

Infine l'algoritmo di ricerca ritorna l'elemento di $T$ con chiave $k$ se esiste, *nil* altrimenti:

```
HASH-SEARCH(k, T[0..m − 1])
  for i ← 0 to m − 1 do
      j ← h(k, i)
      if T[j] = nil then
          return nil
      else
          if T[j].key = k then
              return T[j]
  return nil
```



## Conoscenza di una struttura dati, esercizio 2

- riportare le caratteristiche di un *heap minimo*
- il seguente array rappresenta un *heap minimo*

$$(1,2,3,10,8,9,7,14,15,16,13)$$

- effettuare l'estrazione del minimo riportando lo heap dopo ogni scambio di elementi

## Conoscenza di una struttura dati, esercizio 2

Uno heap minimo è un albero binario semi-completo sinistro, vale a dire completo sino al penultimo livello mentre ciascun vertice dell'ultimo non ha lacune alla sua sinistra; inoltre la chiave di ogni vertice padre è minore o uguale di quella dei suoi eventuali figli. (Nella rappresentazione in forma di array $H$ l'eventuale figlio sinistro di $H[i]$ è $H[2i]$ ed il suo eventuale figlio destro è $H[2i+1]$.)

Gli stati dello heap dato successivi all'estrazione del minimo in radice durante il processo di ricostruzione dello heap sono:

$$(\underline{13}, \underline{2}, 3, 10, 8, 9, 7, 14, 15, 16)$$
$$(2, \underline{13}, 3, 10, \underline{8}, 9, 7, 14, 15, 16)$$
$$(2, 8, 3, 10, 13, 9, 7, 14, 15, 16)$$

Gli elementi sottolineati sono quelli coinvolti nello scambio che produce la riga successiva.



## Grafi, visite, esercizio

Si effettui la visita in profondità, a partire dal nodo 1, del seguente grafo di sei nodi dato tramite le liste di adiacenza:

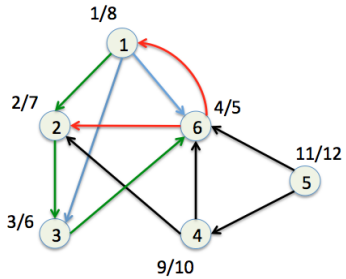
```
1: 2, 3, 6
2: 3
3: 6
4: 2, 6
5: 4, 6
6: 1, 2
```

Si disegni la foresta generata della visita, riportando per ogni nodo i tempi di inizio e fine visita (ossia i valori degli attributi $d$ ed $f$) e per ogni arco il suo tipo secondo la classificazione degli archi durante una visita in profondità.



## Grafi, visite, esercizio

La visita DFS del grafo, con i valori $d/f$ riportati in corrispondenza dei nodi, risulta:

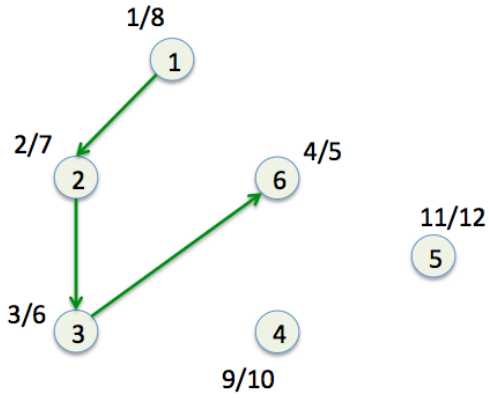



Gli archi d'albero sono: $(1,2), (2,3), (3,6)$. Archi all'indietro sono $(6,1)$ e $(6,2)$. Archi in avanti sono $(1,3)$ e $(1,6)$. Gli archi trasversali sono i rimanenti: $(4,2), (4,6), (5,4), (5,6)$.



## Grafi, visite, esercizio

Quindi la visita genera la foresta, in cui 4 e 5 sono alberi di un solo nodo:



## Grafi, Dijkstra, esercizio

Si applichi l'algoritmo di Dijkstra al grafo riportato sotto con le liste di adiacenti a partire dal nodo $A$. Riportare il contenuto della coda di priorità prima di ogni estrazione.

$A : B(3), C(6), D(7)$
$B : C(2), E(1)$
$C : D(1)$
$D :$
$E : D(4)$



## Grafi, Dijkstra, esercizio

$A : B(3), C(6), D(7)$
$B : C(2), E(1)$
$C : D(1)$
$D :$
$E : D(4)$

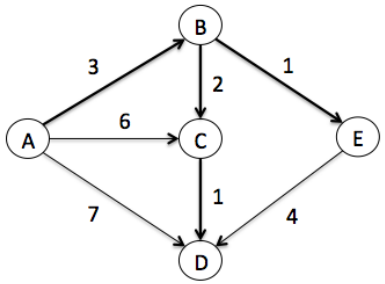
Quelli che seguono sono gli stati della coda (rappresentata come uno heap minimo in forma di array) prima dell'estrazione di ciascun vertice:

$(A, 0), (B, \infty), (D, \infty), (C, \infty), (E, \infty)$
$(B, 3), (D, 7), (C, 6), (E, \infty)$
$(E, 4), (D, 7), (C, 5)$
$(C, 5), (D, 7)$
$(D, 6)$



## Grafi, Dijkstra, esercizio

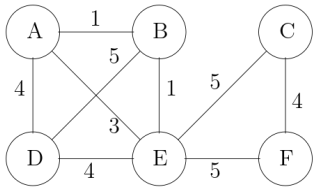
Il grafico, in cui gli archi di maggior spessore sone quelli che compongono i cammini minimi da $A$ scelti dall'algoritmo di Dijkstra, risulta:






## Grafi, alb. minimo ricoprente, esercizio

Si disegnino tutti gli alberi minimi ricoprenti del grafo seguente.



# Grafi, alb. minimo ricoprente, esercizio

Gli MST, il cui peso è 15, sono i seguenti: