Strutture Dati

Lezione 3
Analisi degli algoritmi
Le strutture dati

Oggi parleremo di …

- Analisi degli algoritmi
  - Complessità
    - calcolo
    - equazioni di ricorrenza
- Cosa sono le strutture dati
- Tipo di dato astratto
  - definizione
  - funzioni
- Tipo di dati astratto *Num_Nat*

Algoritmo 4: analisi completa

```
int Count_4( int N)
1   sum = 0                    O(1)
2   for i = 1 to N             O(N)
3       for j = 1 to N         O(N²)
4         if i <= j then       O(N²)
5           sum = sum+1        O(N(N+1)/2) = O(N²)
6   return sum                 O(1)
```

Il tempo di esecuzione è $O(N^2)$

Tempi di esecuzione asintotici

Algoritmo	Tempo di Esecuzione	Limite asintotico
Algoritmo 1	$\frac{3}{2}N^2+\frac{7}{2}N+2$	$O(N^2)$
Algoritmo 2	$6N+2$	$O(N)$
Algoritmo 3	5	$O(1)$

Stima del limite asintotico superiore

- Come definire un semplice metodo per *stimare il limite asintotico superiore* ***O*** del tempo di esecuzione di *algoritmo iterativi*
  - stabilire il limite superiore per le operazioni elementari
  - stabilire il limite superiore per le strutture di controllo
- Ci dà un limite superiore che funge da stima, *non garantisce* di trovare la *funzione precisa* del *tempo di esecuzione*

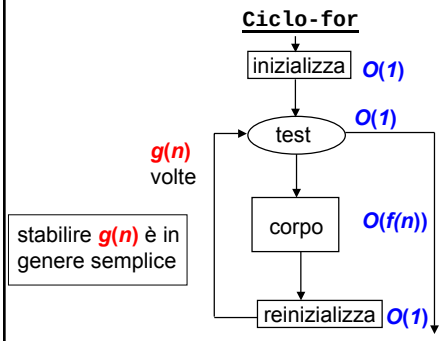
Tempo di esecuzione

Operazioni Semplici

- *operazioni aritmetiche* (+, *, …)
- *operazioni logiche* (&&, ||, ….)
- *confronti* (≤, ≥, =, …)
- *assegnamenti* (`a = b`) senza chiamate di funzione
- *operazioni di lettura* (`read`)
- *operazioni di controllo* (`break, continue, return`)

$$T(n) = \Theta(1) \Rightarrow T(n) = O(1)$$

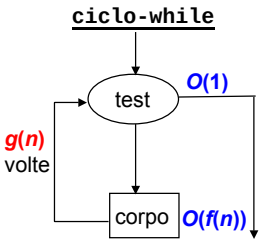
Tempo di esecuzione: ciclo for



stabilire $g(n)$ è in genere semplice

$T(n) = O(g(n) \times f(n))$

Tempo di esecuzione: ciclo while



Bisogna stabilire un limite per il numero di iterazioni del ciclo, $g(n)$

$T(n) = O(g(n) \times f(n))$

Ciclo while: esempio

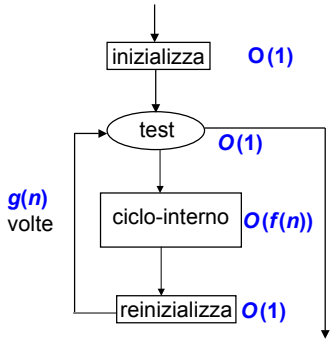
Ricerca dell'elemento x all'interno di un array $A[1 \ldots n]$:

```
i = 0                  (1)
while x ≠ A[i]         (2)
    i = i+1            (3)
```

(1)	$O(1)$
test in (2)	$O(1)$
(3)	$O(1)$
iterazioni	massimo n

$O(\text{ciclo-while}) = O(1) + n\,O(1) = O(n)$

Tempo di esecuzione: cicli innestati



$T(n) = O(g(n) \times f(n))$

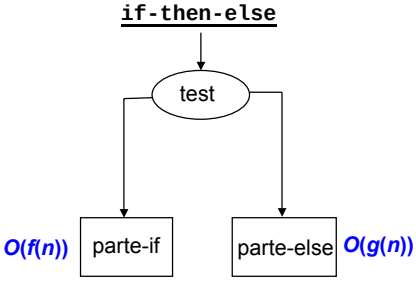
Cicli innestati: esempio

```
for i = 1 to n
    for j = 1 to n
        k = i + j
```

$\left.\begin{matrix} \\ \\ \end{matrix}\right\} = O(n) \Bigg\} = O(n^2)$

$T(n) = O(n \times n) = O(n^2)$

Tempo di esecuzione: If-Then-Else



$O(max(f(n), g(n)))$

If-Then-Else: esempio

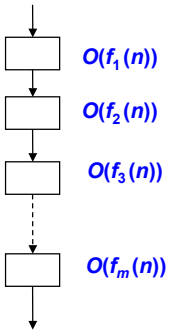
```
if A[1][i] = 0 then
   for i = 1 to n
      for j = 1 to n
         A[i][j] = 0
else
   for i = 1 to n
      A[i][i] = 1
```

$= O(n) \} = O(n^2)$ (then) ; $= O(n)$ (else)

if: $T(n) = O(n^2)$
else : $T(n) = O(n)$

$T(n) = max\,(O(n^2), O(n)) = O(n^2)$

Tempo di esecuzione: blocchi sequenziali



$O(f_1(n) + f_2(n) + \ldots + f_m(n))$

$O(max(f_i(n)))\quad i = 1, \ldots, m$

Blocchi sequenziali: esempio

```
for i = 1 to n
   A[1] = 0
for i = 1 to n
   for j = 1 to n
      A[i] = A[i] + A[i]
```

$= O(n)$; $= O(n)$ $\} = O(n^2)$

$T(n) = O(max(f(ciclo\text{-}1), f(ciclo\text{-}2)))$
$= O(max(n, n^2))$
$= O(n^2)$

Ordinamento per selezione

```
void sort(int lista[], int n)
{
 int i,j,min;

 for(i=0; i<n-1; i++)
  {
   min = i;
   for(j=i+1; j<n; j++)
      if(lista[j] < lista[min])
            min = j;
   SWAP(&lista[i], &lista[min]);
  }
}
```

$= O(n) \} = O(n^2)$

$T(n)=O(n^2)$

E per gli algoritmi ricorsivi?

- Il tempo di esecuzione è espresso tramite una equazione di ricorrenza
- Un'equazione di ricorrenza è una funzione definita in termini di se stessa (Es. fattoriale)
- Sono necessarie tecniche specifiche per risolvere le equazioni di ricorrenza

Complessità degli algoritmi ricorsivi

- Esempio: Fattoriale

```
int fact(int n)
{
  if (n≤1) return 1;   /* Caso Base */
  return n*fact(n-1);  /* Passo Induttivo */
}
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(1) + T(n-1) & \text{se } n > 1 \end{cases}$$

Soluzione di equazioni di ricorrenza

- Il metodo è iterativo
  - Si itera la regola induttiva di $T(n)$ in termini di n e del caso base
  - Richiede manipolazione delle somme

Soluzione di equazioni di ricorrenza

Base: $T(1) = a$
Induzione: $T(m) = b + T(m-1)$

I. Sostituire *m* per *n*, *n*-1, *n*-2, … finché si ottiene il caso base

1) $T(n) = b + T(n-1)$ sostituire ***m*** con ***n***
2) $T(n-1) = b + T(n-2)$ sostituire ***m*** con ***n***-1
3) $T(n-2) = b + T(n-3)$ sostituire ***m*** con ***n***-2

…..

n-1) $T(2) = b + T(1)$ sostituire ***m*** con 2

$T(1) = a$ noto

Soluzione di equazioni di ricorrenza

II. Sostituire $T(n-1)$, $T(n-2)$... fino al caso base e sostituirlo

$T(n) = b + T(n-1) =$
$b + b + T(n-2) = 2*b + T(n-2) =$
$2*b + b + T(n-3) = 3*b + T(n-3) =$
$3*b + b + T(n-4) = 4*b + T(n-4) =$
……
$T(n) = (n-1) * b + T(1)$

Inserire il caso base

$T(n) = (n-1) * b + a$

III. *Valutare l'espressione O-grande associata*

$T(n) = b*n - b + a = O(n)$

Soluzione di equazioni di ricorrenza

Esempio 1: Fattoriale

```
int fact(int n)
{
  if (n≤1) return 1;   /* Caso Base
  return n*fact(n-1);  /* Passo Induttivo
}
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(1) + T(n-1) & \text{se } n > 1 \end{cases}$$

Equazione di ricorrenza

Base: $T(1) = a$
Induzione: $T(m) = b + T(m-1)$

Esempio 1: Analisi del fattoriale

Caso Base: $T(0) = O(1)$
$T(1) = O(1)$

Passo Induttivo: $O(1) + max(O(1), T(n-1))$
$O(1) + T(n-1)$ per $n>1$

Per il fattorale, l'analisi risulta $T(n) = O(n)$

Esempio 2

$T(n) = T(n-1) + n$ per $n \geq 2$ con $T(1) = 1$

$$\begin{aligned} T(n) &= T(n-1) + n \\ &= T(n-2) + (n-1) + n \\ &= T(n-3) + (n-2) + (n-1) + n \\ &\bullet \\ &\bullet \\ &\bullet \\ &= T(1) + 2 + \ldots + (n-2) + (n-1) + n \\ &= 1 + 2 + \ldots + (n-2) + (n-1) + n \\ &= \frac{n(n+1)}{2} \end{aligned}$$

$T(n)=O(n^2)$

Esempio 3

$T(n) = T\left(\frac{n}{2}\right) + 1$ per $n \geq 2$ con $T(1) = 0$

Ipotesi: $n=2^h$ $h \geq 0$

$$
\begin{aligned}
T(2^h) &= T(\tfrac{2^h}{2}) + 1 \\
&= T(2^{h-1}) + 1 \\
&= T(2^{h-2}) + 1 + 1 \\
&= T(2^{h-3}) + 3 \\
&\bullet \\
&\bullet \\
&\bullet \\
&= T(2^{h-h}) + h \\
&= T(1) + h \\
&= \log n
\end{aligned}
$$

E' la complessità dell'algoritmo di ricerca binaria!

T(n)=O(logn)

Esempio 3

$T(n) \in O(\log n)$ per $n=2^h$ $h \geq 0$

$T(n) = ?$ per $n \neq 2^h$

Si consideri l'intervallo $n_1 < n < n_2$ tale che $n_1 = 2^h, n_2 = 2^{h+1}$ cioè $n_2 = 2n_1$

Poiché $T(n)$ è una funzione crescente di n, $T(n) \in O(\log n_2)$

Inoltre, $\log n_2 = \log 2n_1 < \log 2n = 1 + \log n$

T(n)∈ O(logn) per qualsiasi valore di n

Le strutture dati

Cosa sono?

Cosa sono le strutture dati

- E' importante definire una struttura svincolandola dalla concreta rappresentazione all'interno del calcolatore
- Una struttura dati ammette più implementazioni
  - con costi diversi in termini di spazio e in termini di tempo
- Nell'analisi di un algoritmo si considerano le strutture svincolate dalla loro implementazione
  - mette in evidenza il metodo risolutivo, tralasciando gli aspetti implementativi
  - facilita la comprensione
  - facilita l'analisi dei suoi costi

Cosa sono le strutture dati

- Un **Dato** è un valore che una variabile può assumere
- Il **Tipo di Dati** (di una variabile) è
  - l'insieme dei valori che la variabile può assumere
  - le operazioni che possono essere effettuate su tali valori
- Una **Struttura Dati** è un agglomerato di variabili, eventualmente di tipo diverso, connesse in vario modo
- Le strutture dati vengono costruite usando i meccanismi di astrazione del linguaggio stesso

Tipo di dati astratto

- Un **tipo di dati astratto** (Abstract Data Type o **ADT**) è un tipo di dati in cui la specifica degli oggetti e la specifica delle operazioni sugli oggetti sono distinte dalla rappresentazione degli oggetti e dall'implementazione delle operazioni
- I dettagli dell'implementazione sono nascosti all'utente dell' ADT

Tipo di dati astratto

- Si ha la separazione del significato logico delle operazioni in un ADT dai dettagli dell'implementazione
- La definizione dei dati da parte di un ADT è nella loro forma logica
- La realizzazione dei dati dentro una struttura dati è nella loro forma fisica

31

Tipi di Dati Astratti e Strutture Dati

- Struttura Dati è la realizzazione fisica di un Tipo di Dati Astratto
- Generalmente, ci sono tre categorie di operazioni su ADT
  - Costruzioni: creano una nuova istanza dell'insieme
  - Interrogazioni: restituiscono informazioni sull'insieme
  - Operazioni di modifica: modificano l'insieme

32

Il tipo di dati astratto *Num_Nat*

```
Struttura Num_Nat
   oggetti: un sottoinsieme ordinato di numeri interi che
        parte da 0 e termina con Int_Max
   funzioni: per ogni x,y ∈ Num_Nat; TRUE e FALSE ∈ Booleano;
        +,-,< e == usuali operazioni su numeri interi

   Num_Nat Zero()          ::=   0
   Booleano E_Zero(x)      ::=   if(x) return FALSE
                                 else return TRUE
   Booleano Uguale(x,y)    ::=   if(x==y) return TRUE
                                 else return FALSE
   Num_Nat Somma(x,y)      ::=   if(x+y)<=Int_Max return x+y
                                 else return Int_Max
   Num_Nat Successivo(x)   ::=   if(x==Int_Max) return x
                                 else return x+1
   Num_Nat Sottrai(x,y)    ::=   if(x<y) return 0
                                 else return x-y
end Num_Nat
```

Laboratorio: Esercitazione 2

- In un sistema di numerazione in base BASE le cifre assumono i valori compresi tra 0 e BASE–1 (cosa nota!). Si vogliano generare tutti i numeri costituiti da NDATI cifre.
- BASE e NDATI siano forniti da tastiera.
- Per BASE = 2 e NDATI = 3 il programma dovrà restituire la sequenza

  0 0 0
  0 0 1
  0 1 0
  0 1 1
  1 0 0
  1 0 1
  1 1 0
  1 1 1

34