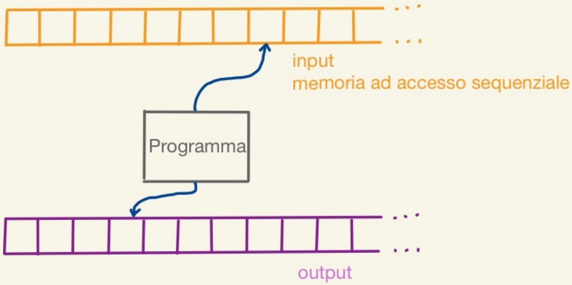
Algoritmi e Strutture Dati

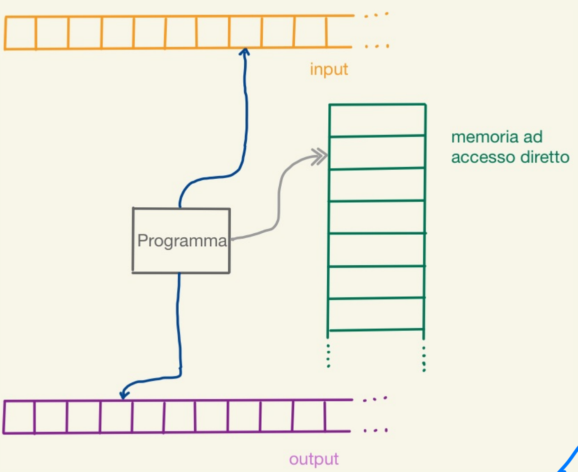
Lezione 5

7 ottobre 2022

Macchina di Turing



Macchina ad accesso diretto (RAM)



Accesso a memoria
LOAD/STORE

Aritmetico/logica
+ , * , / , -

Operazioni di cfr
<= < > >= != ==

Operazioni tempo O(1)

su tipi primitivi int float

Minimo di una sequenza

ALGORITMO minimo (Sequenza seq) → elemento

min ← primo elemento di seq

WHILE seq non è finita DO

x ← prossimo elemento di seq

IF $x < min$ THEN $min \leftarrow x$

RETURN min

$n = \#$ elementi di seq

n° di cfr $= n-1$

n° di assegnamenti $\leq 2n-1$

Se costo operazioni è $O(1) \rightarrow$ tempo totale $O(n)$

Accavallavacca

Arnese impiegato per caricare mucche su cavalli oppure una mucca sull'altra allo scopo di risparmiare spazio nel trasporto

`A C C A V A L L A V A C C A`

`A C C A V A L L A R E`

```
ALGORITMO minimo (Sequenza seq) → elemento
   min ← primo elemento di seq
   WHILE seq non è finita DO
      x ← prossimo elemento di seq
      IF x < min THEN min ← x
   RETURN min
```

$n = \#$ elementi di seq

n° di cfr $= n-1$

n° di assegnamenti $\leq 2n-1$

Sequenza di stringhe di lunghezza M

Costo di un cfr $O(M)$

Costo assegnamento (copiando la stringa) $O(M)$

$\#$ operazioni $O(n)$

$\downarrow$ $\times$

costo di ogni operazione $O(M)$

$\Rightarrow$ costo totale $O(n \cdot M)$

Calcolo x^x $\quad x \geq 0$ $\quad x^x = \underbrace{x \cdot x \cdots x}_{x \text{ volte}}$

ALGORITMO xx (intero x) → intero

```
p ← 1
FOR i ← 1 TO x DO
    p ← p * x
RETURN p
```

math/big

operazioni

- prodotti e assegnamenti (p, x) $\quad O(x)$
- confronti e incrementi (i) $\quad O(x)$

<u>Tempo</u> se le operazioni costano $O(1)$
tempo totale $O(x)$

CRITERI DI COSTO

- UNIFORME
  - operazioni elementari 1 unità di tempo
  - dati elementari → spazio $O(1)$
- LOGARITMICO
  - tempo proporzionale alla grandezza della rappresentazione dei dati
  - intero x → $\underline{O(\log x)}$
  - → spazio $O(\log x)$

ALGORITMO xx (intero x) → intero

```
p ← 1
FOR i ← 1 TO x DO
    p ← p * x
RETURN p
```

iterazione i $\quad p \leftarrow p * x$

- inizio iterazione p contiene x^{i-1}
- stop " " x^i

- Costo di $p * x$

$$\underbrace{\lg x^{i-1}}_{p} + \underbrace{\lg x}_{x} = (i-1)\lg x + \lg x = \underline{i \lg x}$$

- Costo assegnamento $p \leftarrow \dots$

$$\underbrace{\lg x^i}_{\text{n° di bit da copiare}} = \underline{i \lg x}$$

$c \cdot i \lg x + d$ $\quad c, d$ costanti

costo iterazione i

costo totale

$$e + \sum_{i=1}^{x} (c \cdot i \lg x + d) = e + c \cdot \lg x \underbrace{\sum_{i=1}^{x} i}_{\frac{x(x+1)}{2}} + \sum_{i=1}^{x} d$$

$$= e + c \cdot \lg x \cdot \frac{x(x+1)}{2} + dx$$

$$= \Theta(x^2 \lg x)$$

Costo logaritmico $T(x) = \Theta(x^2 \lg x)$ tempo

$S(x) = \Theta(x \lg x)$ spazio

Criterio di costo uniforme

Tempo Ogni istruzione elementare utilizza un'unità di tempo *indipendentemente* dalla grandezza degli operandi

Spazio Ogni variabile elementare utilizza un'unità di spazio *indipendentemente* dal valore contenuto

Criterio ragionevole quando i valori trattati dall'algoritmo sono di grandezza limitata (es. variabili intere o in virgola mobile nel range di tipi primitivi `int`, `long`, `float`, `double`)

Criterio inadeguato quando si manipolano quantità arbitrariamente grandi: è necessario tenere conto della loro grandezza, in particolare della lunghezza delle loro rappresentazioni

Criterio di costo logaritmico

Tempo Il tempo di calcolo di ciascuna operazione è *proporzionale alla lunghezza dei valori coinvolti*

somma
$x+y$

$O(\log x + \log y)$

Spazio Lunghezza della rappresentazione del dato

Esempi:

- interi: numero di bit
- stringhe: numero di caratteri

Complessità di algoritmi (Tempo)

- $\mathcal{A}$ algoritmo
- I istanza
- $tempo(I) =$ tempo impiegato da $\mathcal{A}$ su input I

La complessità in tempo viene definita come funzione $T : \mathbb{N} \to \mathbb{N}$ della *lunghezza dell'input*

- Caso peggiore
- Caso migliore
- Caso medio

Analoghe definizioni possono essere date per altre risorse (es. spazio)

3 istanze di lunghezza n $\quad i_1 \quad i_2 \quad i_3$

tempi $\quad 12 \quad 24 \quad 9$

caso peggiore $\quad T_{worst} = 24$

caso migliore $\quad T_{best} = 9$

caso medio $\quad T_{avg} = \dfrac{12 + 24 + 9}{3} = 15$

se equiprobabili

probabilità diverse

$$\begin{array}{ccc} \frac{1}{2} & \frac{1}{6} & \frac{1}{3} \\ i_1 & i_2 & i_3 \end{array}$$

$$T_{avg} = \frac{1}{2} \cdot 12 + \frac{1}{6} 24 + \frac{1}{3} 9 =$$

$$= 6 + 4 + 3 = 13$$

Complessità di algoritmi (Tempo)

- $\mathcal{A}$ algoritmo
- I istanza
- $tempo(I)$ = tempo impiegato da $\mathcal{A}$ su input I

Caso peggiore $T_{worst} : \mathbb{N} \rightarrow \mathbb{N}$
Tempo massimo utilizzato su input di lunghezza n:

$$T_{worst}(n) = \max\{tempo(I) \mid |I| = n\}$$

Complessità di algoritmi (Tempo)

- $\mathcal{A}$ algoritmo
- I istanza
- $tempo(I) =$ tempo impiegato da $\mathcal{A}$ su input I

Caso migliore $T_{best} : \mathbb{N} \to \mathbb{N}$ (in genere poco significativo)
Tempo minimo utilizzato su input di lunghezza n:

$$T_{best}(n) = \min\{tempo(I) \mid |I| = n\}$$

Complessità di algoritmi (Tempo)

- $\mathcal{A}$ algoritmo
- I istanza
- $tempo(I) =$ tempo impiegato da $\mathcal{A}$ su input I

Caso medio $T_{avg} : \mathbb{N} \to \mathbb{N}$
Media dei tempi utilizzati su input di lunghezza n, pesata con le probabilità:

$$T_{avg}(n) = \sum_{|I|=n} Prob(I) \cdot tempo(I)$$

Complessità di algoritmi

- $\mathcal{A}$ algoritmo
- $\mathcal{R}$ risorsa (es. tempo)
- $f : \mathbb{N} \to \mathbb{N}$ funzione

Limitazione superiore (upper bound)

$\mathcal{A}$ ha costo di esecuzione $O(f(n))$ rispetto a $\mathcal{R}$
se la quantità di $\mathcal{R}$ *sufficiente* per eseguire $\mathcal{A}$ istanze di lunghezza n è $O(f(n))$

$\forall$ istanza I con $|I| = n$

$T(I) \leq O(f(n))$

Complessità di algoritmi

- $\mathcal{A}$ algoritmo
- $\mathcal{R}$ risorsa (es. tempo)
- $f : \mathbb{N} \to \mathbb{N}$ funzione

caso peggiore

Limitazione superiore (upper bound)
$\mathcal{A}$ ha costo di esecuzione $O(f(n))$ rispetto a $\mathcal{R}$
se la quantità di $\mathcal{R}$ *sufficiente* per eseguire $\mathcal{A}$ istanze di lunghezza n è $O(f(n))$

Limitazione inferiore (lower bound)
$\mathcal{A}$ ha costo di esecuzione $\Omega(f(n))$ rispetto a $\mathcal{R}$
se la quantità di $\mathcal{R}$ *necessaria* per eseguire $\mathcal{A}$ su istanze di lunghezza n è $\Omega(f(n))$

$\forall n \; \exists I$ con $|I| = n$

t.c. $T(I) \geq \Omega(f(n))$

Complessità di problemi

- $\mathcal{P}$ problema
- risorsa tempo

Quanto tempo si impiega per risolvere $\mathcal{P}$?

Complessità di problemi

- $\mathcal{P}$ problema
- risorsa tempo

Quanto tempo si impiega per risolvere $\mathcal{P}$?

- Trovo un algoritmo $\mathcal{A}$ che risolve $\mathcal{P}$ in tempo $O(T(n))$

  $\Longrightarrow$ tempo $O(T(n))$ è *sufficiente* per risolvere $\mathcal{P}$

limitazione superiore

Complessità di problemi

- $\mathcal{P}$ problema
- risorsa tempo

Quanto tempo si impiega per risolvere $\mathcal{P}$?

- Trovo un algoritmo $\mathcal{A}$ che risolve $\mathcal{P}$ in tempo $O(T(n))$

  $\Longrightarrow$ tempo $O(T(n))$ è *sufficiente* per risolvere $\mathcal{P}$

$\mathcal{A}$ è l'algoritmo che risolve $\mathcal{P}$ più velocemente?

NO

trovo un algoritmo più veloce

sì

- Dimostro che *ogni algoritmo* che risolve $\mathcal{P}$ utilizza tempo $\Omega(T(n))$

  $\Longrightarrow$ tempo $\Omega(T(n))$ è *necessario* per risolvere $\mathcal{P}$

limiti inferiori

Complessità di problemi

- $\mathcal{P}$ problema
- $\mathcal{R}$ risorsa (es. tempo)
- $f, g : \mathbb{N} \to \mathbb{N}$ funzioni

Limitazione superiore (upper bound)
$\mathcal{P}$ ha complessità $O(f(n))$ rispetto a $\mathcal{R}$
quando *esiste un algoritmo* che risolve $\mathcal{P}$
il cui costo di esecuzione è $O(f(n))$ rispetto a $\mathcal{R}$

Limitazione inferiore (lower bound)
$\mathcal{P}$ ha complessità $\Omega(g(n))$ rispetto a $\mathcal{R}$
quando *ogni algoritmo* che risolve $\mathcal{P}$
ha costo di esecuzione $\Omega(g(n))$ rispetto a $\mathcal{R}$