Introduzione agli algoritmi e le strutture dati

Annalisa De Bonis

Docente

- Annalisa De Bonis
- Studio 44, quarto piano, stecca 7
- Email: debonis@dia.unisa.it
- URL: http://www.di.unisa.it/~debonis/

Orario lezioni

- Martedì: 14-16, F8
- Giovedì: 11-13, F8
- Venerdì: 11-13, F8

Orario ricevimento

- Martedì 16-17
- Giovedì 14:30-16:30
- Lezione o altri impegni improrogabili durante l'orario di ricevimento?
    - Contattatemi via email per fissare un appuntamento
- Variazioni all'orario di ricevimento saranno comunicate con un avviso sulla pagina del corso.

Libro di testo

P. Crescenzi, G. Gambosi, R. Grossi, G. Rossi

Strutture di dati e algoritmi

Progettazione, analisi e programmazione

Pearson

http://algoritmica.org

Obiettivi del corso

- Impareremo a
  - progettare e ad analizzare gli algoritmi
  - organizzare i dati da elaborare in modo che possano essere usati in modo agevole dagli algoritmi

Algoritmi

- Sequenza di passi per risolvere un determinato problema
  - Comunemente
    - Riceve in input uno o più valori
    - Produce in output o più valori

Algoritmo: cosa è

- Strumento per risolvere un **problema computazionale**
  - Il problema computazionale definisce una relazione tra input e l'output
  - Un algoritmo è quindi una procedura per ottenere la desiderata relazione input/output

Esempio: Il problema dell'ordinamento

- **Input:** sequenza di numeri $\langle a_1, \ldots, a_n \rangle$
- **Output:** una permutazione $\langle a'_1, \ldots, a'_n \rangle$ della sequenza input tale che $a'_1 \leq \ldots \leq a'_n$
- **Algoritmo di ordinamento:** procedura che prende in input $\langle a_1, \ldots, a_n \rangle$ e produce in output $a'_1 \leq \ldots \leq a'_n$
- Esempio:
  - **Sequenza input:** $\langle 9,4,5,20,31,2 \rangle$
  - **Sequenza output:** $\langle 2,4,5,9,20,31 \rangle$

Algoritmi nella storia

- Algoritmi di tipo numerico furono studiati da matematici babilonesi ed indiani più di 3000 anni fa.
- Algoritmi in uso fino a tempi recenti furono studiati dai matematici greci 500 anni a.C.
  - Esempio: Algoritmo di Euclide per il Massimo Comune Divisore (~300 a.C.)
  - Esempio: Algoritmi geometrici (calcolo di tangenti, sezioni di angoli, ...)

Algoritmi

- Permettono di risolvere problemi importantissimi:
  - Trasmissione dati in Internet (come si gestisce in pratica il flusso di dati nei vari router della rete?)
  - Ricerca nel WEB (Google)
  - Bioinformatica (come il DNA determina le nostre caratteristiche?)
  - Transazioni economiche
    - Gestione delle aste on-line su Ebay?
    - Compravendita di azioni su Internet?

Algoritmo corretto

- Un algoritmo è **corretto** se per ogni istanza input del problema si ferma producendo l'output corretto
- Un algoritmo **non** corretto potrebbe per certi input
  - non fermarsi affatto
  - produrre un valore output errato

Algoritmi efficienti

- Algoritmi Efficienti = Algoritmi che usano "poche" risorse
- Risorse: Tempo e Spazio richiesto dall'algoritmo
- Studieremo:
  - Analisi degli algoritmi: come misurare le risorse usate da un algoritmo
  - Tecniche generali per il progetto di algoritmi: come progettare algoritmi che usano "poche" risorse

- **Algoritmi efficienti permettono di scrivere programmi efficienti**

Esempio: ricerca in un array ordinato

- Dato un elemento x e un array ordinato a di n elementi, vogliamo trovare l'indice della cella dell'array contenente x oppure riportare che x non e` nell'array.

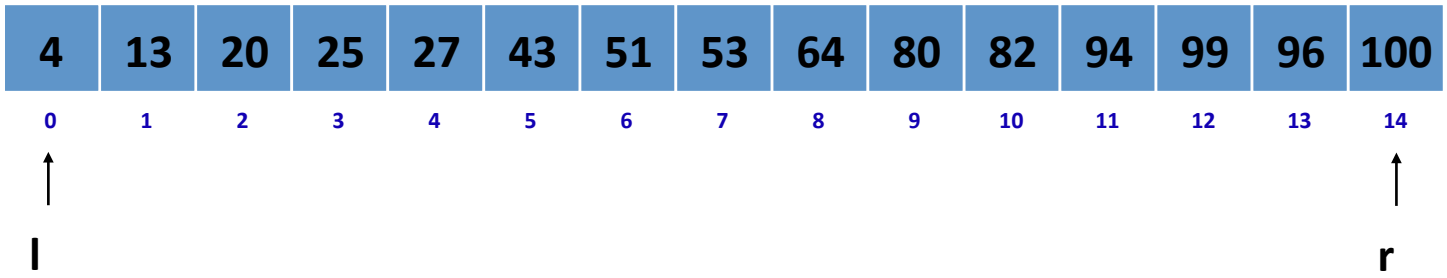
- Ricerca sequenziale: scandisco l'array dall'inizio alla fine
  - n confronti nel caso in cui gli elementi sono tutti minori di x

4	13	20	25	27	43	51	53	64	80	82	94	99	96	100
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Esempio: ricerca in un array ordinato

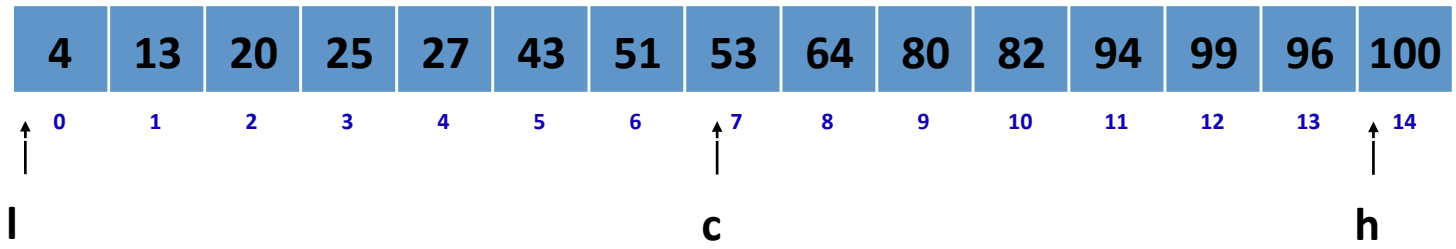
- Dato un valore x e un array ordinato a, vogliamo trovare l'indice della cella dell'array contenente x oppure riportare che x non e` nell'array.

- Idea migliore: ricerca binaria
  - Ad ogni passo
    - mantengo due indici l ed r che delimitano la zona dell'array in cui potrebbe trovarsi x.
    - confronto x con l'elemento centrale



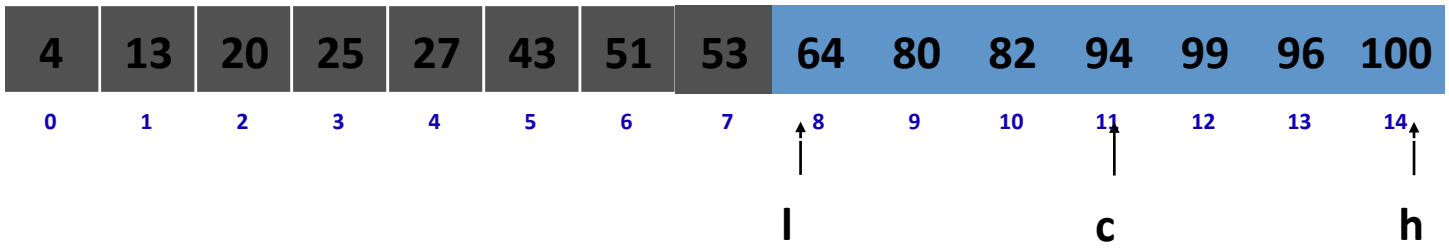
Esempio: ricerca in un array ordinato

- Ricerca binaria di 82
- Primo confronto con 53



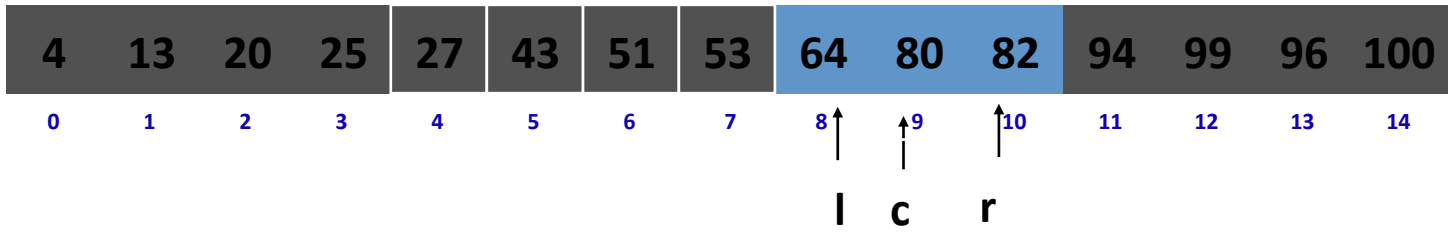
Esempio: ricerca in un array ordinato

- Ricerca binaria di 82
- Secondo confronto con 94



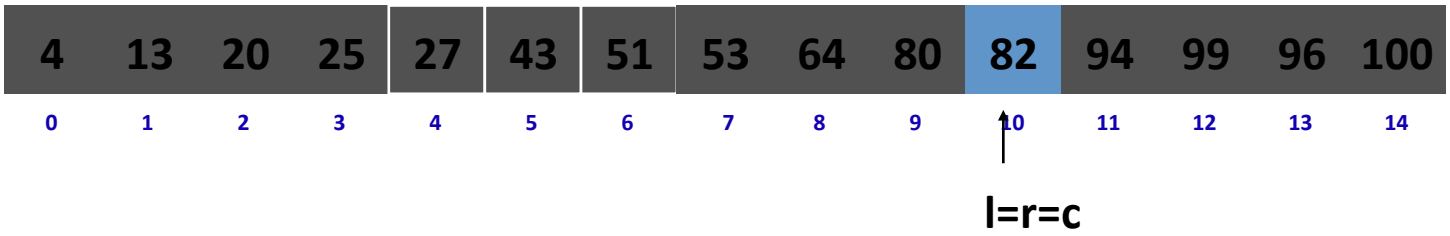
Esempio: ricerca in un array ordinato

- Ricerca binaria di 82
- Terzo confronto con 80



Esempio: ricerca in un array ordinato

- Ricerca binaria di 82
- Quarto confronto con 82



- In totale 4 confronti. Se avessi cercato un altro elemento avrei fatto non piu` di 4 confronti. Per un array di n elementi, si fanno al piu` $\lfloor \log n \rfloor + 1$ confronti.
- Con la ricerca sequenziale avrei fatto 10 confronti e nel caso in cui avessi cercato 100 o un numero piu` grande avrei fatto 15 confronti.

Analisi degli algoritmi

- Per poter analizzare un algoritmo abbiamo bisogno di fare riferimento ad un modello computazionale
- Nell'analisi assumeremo il modello RAM (Random Access Machine)
  - Macchina astratta
  - Basato sullo schema di von Neumann: programmi e dati codificati come sequenze binarie contenute nella memoria

Modello RAM
Caratteristiche principali

- Memoria principale infinita contenente sia i dati che il programma
  - Ogni cella di memoria può contenere una quantità di dati finita
  - Stesso tempo per accedere ad ogni cella di memoria
- Singolo processore
  - Singola unità di elaborazione
  - Due registri:
    - Contatore di programma
    - Accumulatore per effettuare le seguenti **operazioni elementari** : +, -, *, /, <, >, ..., =, &, |, ~, ..., trasferimento e accesso ai dati, salti condizionati e non

Analisi degli algoritmi basata sul modello RAM

- Assunzioni per l'analisi del tempo
  - Operazioni elementari hanno costo costante
  - Costo dell'algoritmo per un certo input = numero istruzioni elementari eseguite
- Assunzioni per l'analisi dello spazio
  - Costo dell'algoritmo per un certo input = numero celle di memoria occupate durante l'esecuzione del programma (oltre a quelle occupate dai dati input)

Caso pessimo e caso medio

- P: problema computazionale
- A: algoritmo che risolve P
- Analisi di A consiste nel trovare una funzione f(n) che esprime il costo computazionale (complessita`) di A per un'istanza di P di dimensione n:
- **caso pessimo:** f(n) = costo **massimo** tra **tutte** le istanze di P aventi dimensioni pari a n
- **caso medio:** f(n) = costo **mediato** tra tutte le **istanze** di P aventi dimensioni pari a n

Calcolo del costo nel caso pessimo

IF (condizione) { blocco1 } ELSE { blocco2 }

c = costo per valutare se la condizione e` verificata

b1= costo per eseguire blocco1

b2= costo per eseguire blocco2

Costo totale= c+ max{b1,b2}

Calcolo del costo al caso pessimo

1. FOR(i=0; i< m ; i=i+1)
2. {corpo}

- a = costo per eseguire la linea 1 una singola volta
  - possiamo assumere costo uguale ad ogni iterazione
- b_i=costo per eseguire la linea 2 all'iterazione i
  - il costo b_i puo` cambiare ad ogni iterazione

Costo totale = $a(m+1) + \sum_{i=0}^{m-1} b_i$

Calcolo del costo al caso pessimo

Esempio: caso in cui il costo del corpo e` costante ad ogni iterazione

FOR(i=0; i< m; i=i+1) //costo a per ogni iterazione

Stampa(a[i]); //costo b_i = b (b e`una costante) per ogni iterazione

Costo totale = $a(m+1) + \sum_{i=0}^{m-1} b_i = a(m+1) + bm$

Calcolo del costo al caso pessimo

Esempio: caso in cui il costo del corpo cambia ad ogni iterazione

```
FOR(i=0; i< m; i=i+1){
 s=0;                        //costo d per ogni singola esecuzione
 FOR(j=0; j < i;j++) {       //costo e per ogni singola esecuzione
   s=s+a[j]                  //costo f per ogni singola esecuzione
  Stampa(s);                 //costo g per ogni singola esecuzione
  }
}
```

Costo del corpo del for esterno all'iterazione i e` b_i= d+ e(i+1)+(f+g)i

Costo totale= $a(m+1)+\sum_{i=0}^{m-1} b_i = a(m+1)+\sum_{i=0}^{m-1}(d+e(i+1)+(f+g)i)$

$$= a+(a+d+e)m+(e+f+g)\sum_{i=0}^{m-1} i$$

$$= a+(a+d+e)m+(e+f+g)m(m-1)/2$$

Guida per il calcolo del costo al caso pessimo

- Il costo di una chiamata a funzione è il costo del suo corpo più il passaggio dei parametri (le funzioni ricorsive meritano una trattazione a parte)

- Il costo di una sequenza di istruzioni è la somma dei costi delle istruzioni nella sequenza