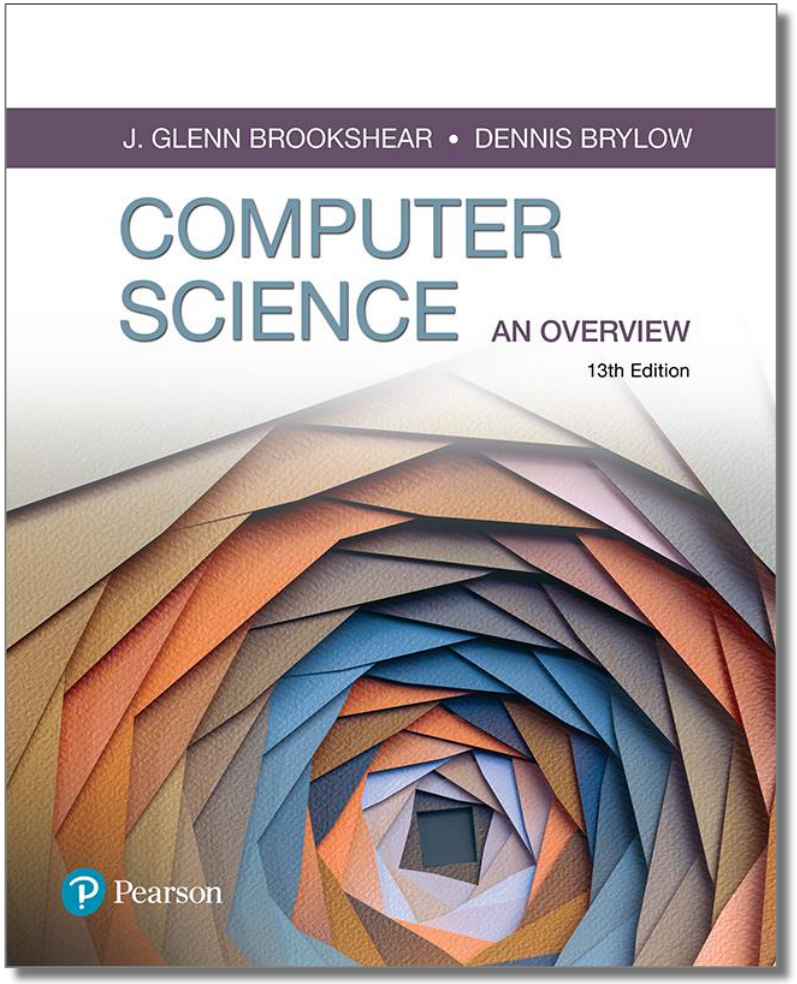
Computer Science An Overview

13th Edition



Chapter 5

Algorithms

Il concetto di algoritmo

- Esempi di algoritmi
  - Convertire da una base ad un’altra
  - Ricetta

- Molti ricercatori credono che ogni attività della mente umana sia il risultato di un algoritmo

Definizione formale di Algoritmo (I)

- Un algoritmo è un insieme ordinato di passi non ambigui ed eseguibili che definisce un processo terminante
- Gli step di un algoritmo possono essere sequenziati in differenti modi
  - Lineare (1, 2, 3, …)
  - Parallela (multiple processors)
  - Causa ed Effetto (circuits)

Definizione formale di algoritmo (II)

- Un processo terminante
  - Culmina con un risultato
  - Può includere sistemi che sono continuamente in esecuzione
    - Hospital systems
    - Long Division Algorithm
- Un processo non terminante
  - Non produce una riposta
  - E’ un Algoritmo non deterministico

Natura degli algoritmi

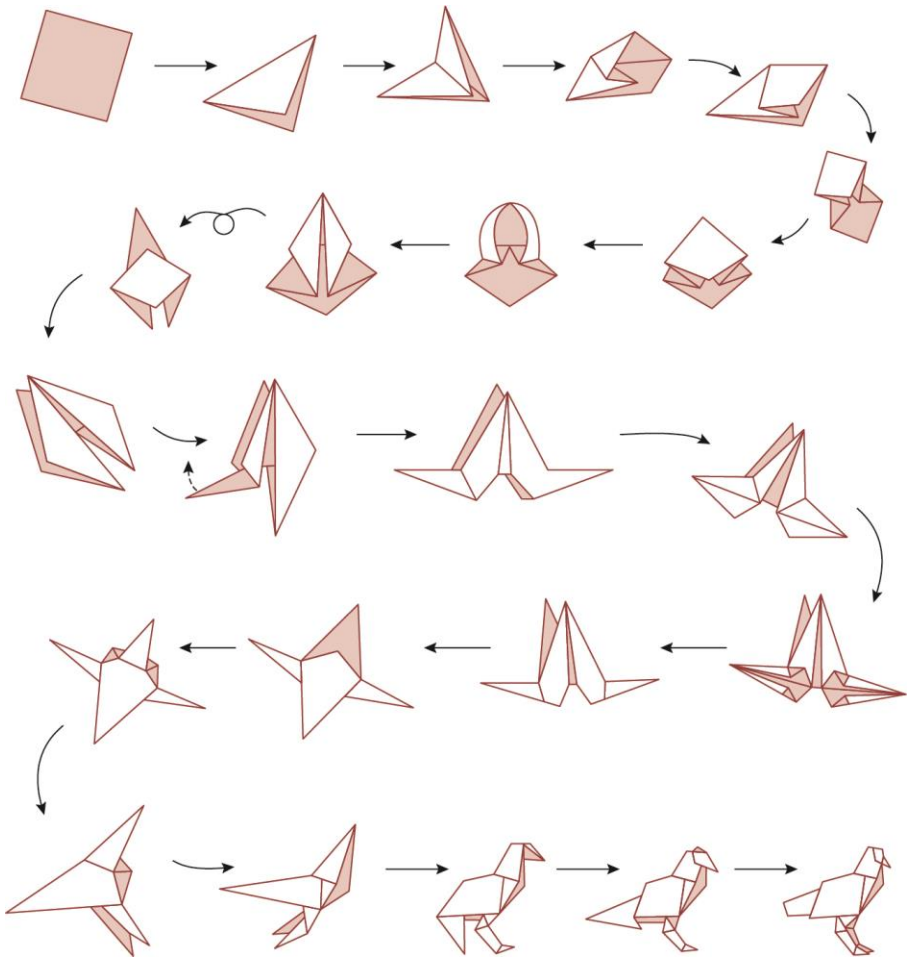
- C’è una differenza tra un algoritmo e la sua rappresentazione.
  - Analogia: differenza tra una storia e un libro

- Un programma è una rappresentazione di un algoritmo.
- Un processo è l’attività di eseguire un algoritmo.

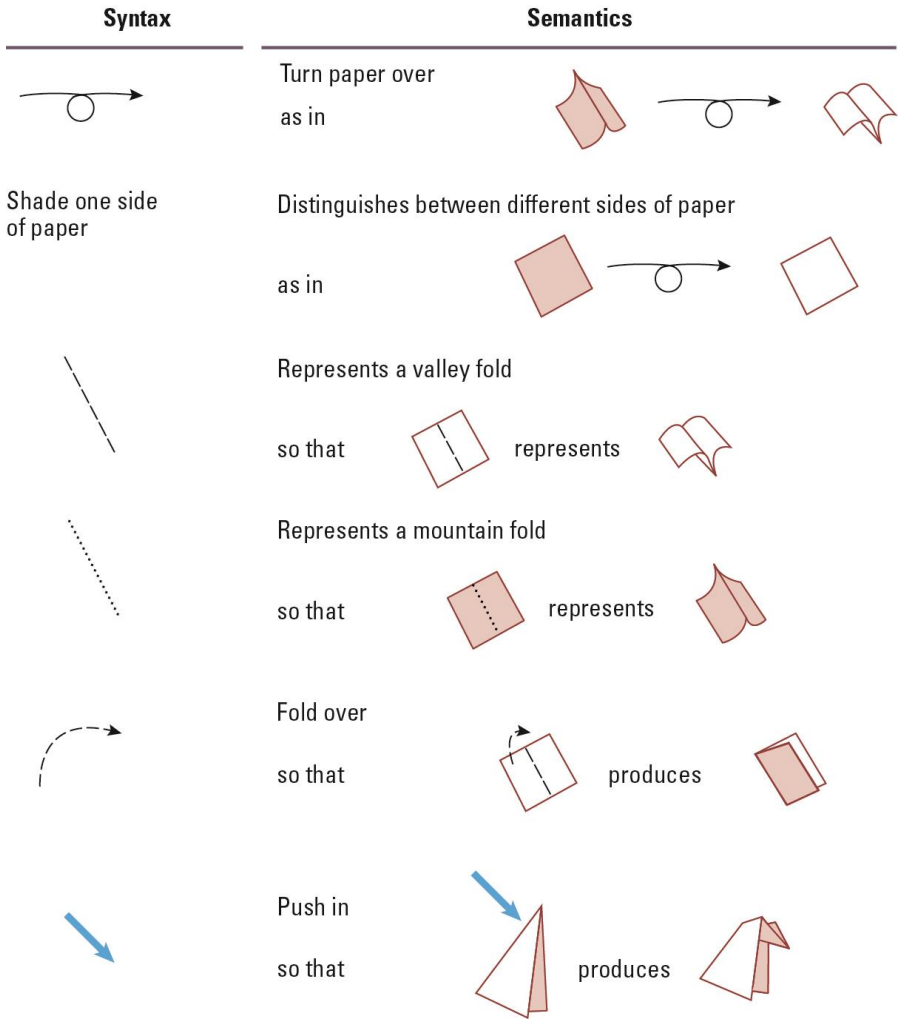
Rappresentazione degli algoritmi

- E’ fatta formalmente con Primitive ben definite
  - Una collezione di primitive costituisce un linguaggio di programmazione.
- E’ fatta informalmente con Pseudocodice
  - Lo Pseudocodice è tra il linguaggio naturale e un linguaggio di programmazione.

Origami



Primitive origami



Progettare un linguaggio di pseudocodice

- Scegliere un comune linguaggio di programmazione
- Perdere alcune delle regole sintattiche
- Permettere alcune del linguaggio naturale
- Usare notazione consistente e coincisa

Primitive di pseudocodice (I)

- Assegnazione

```
nome = espressione
```

Espressione: *combinazione di operandi e operatori che restituisce un risultato*

- Esempio

```
perimetro =  lato1+lato2+lato3
```

Primitive di pseudocodice (II)

- Selezione condizionale

```
if (condition):
    activity
```

- Esempio

```
if (il costo è diminuito):
    abbassa il prezzo del 5%
```

Primitive di pseudocodice (III)

- Selezione condizionale

```
if (condition):
    activity
else:
    activity
```

- Esempio

```
if (l'anno è bisestile):
    d = total / 366
else:
    d = total / 365
```

Primitive di pseudocodice (IV)

- Esecuzione ripetuta

```
while (condition):
    body
```

- Esempio

```
while (rimangono biglietti):
    vendi un biglietto
```

Primitive di pseudocodice (V)

- Condizioni **annidate**

```
if (not raining):
    if (temperature == hot):
        go swimming
    else:
        play golf
else:
    watch television
```

Polya's Problem Solving Steps

- 1. Capire il problema.
- 2. Escogitare un piano per risolvere il problema.
- 3. Eseguire il piano.
- 4. Valutare l'esattezza della soluzione e la possibilità che diventi uno strumento per risolvere altri problemi.

Polya's Steps nel contesto dello sviluppo dei programmi

- 1. Capire il problema.
- 2. Farsi un'idea di come una funzione algoritmica potrebbe risolvere il problema.
- 3. Formulare l'algoritmo e rappresentarlo come un programma.
- 4. Valutare l'accuratezza del programma e la possibilità che diventi uno strumento per risolvere altri problemi.

Problema dell'età dei bambini

- La persona A deve stabilire l'età dei tre figli della persona B.
  - B dice ad A che il prodotto dell'età dei bambini è 36.
  - A replica che è necessaria un'altra informazione.
  - B comunica ad A la somma delle età dei bambini.
  - A replica che un altro indizio è necessario.
  - B dice ad A che il bambino più grande suona il pianoforte.
  - A dice a B le età dei tre bambini.
- Quanti anni hanno i tre bambini?

Problema età dei bambini

a. Triples whose product is 36

(1,1,36)	(1,6,6)
(1,2,18)	(2,2,9)
(1,3,12)	(2,3,6)
(1,4,9)	(3,3,4)

b. Sums of triples from part (a)

1 + 1 + 36 = 38
1 + 2 + 18 = 21
1 + 3 + 12 = 16
1 + 4 + 9 = 14

1 + 6 + 6 = 13
2 + 2 + 9 = 13
2 + 3 + 6 = 11
3 + 3 + 4 = 10

Ispirazione misteriosa

- Potenziale risolutore di un problema: lavoro senza successo → individuazione di una soluzione mentre ci si occupa di tutt'altro

Altri approcci

- Affrontare il problema al contrario (output desiderato → input dato)
- Cercare un problema simile più semplice da risolvere o già risolto (ordinare alfabeticamente)
  - Trascurare alcune restrizioni del problema
  - Risolvere alcuni pezzi del problema (bottom up methodology: specifico→generale)
- Raffinamento a più passi: Divide il problema in problem più piccoli (top-down methodology)

Scoperta di nuovi algoritmi: arte

- La scoperta di nuovi algoritmi resta un’arte più che una scienza

- Non bisogna applicare alla lettera certi approcci, altrimenti si annullerebbero le competenze tecniche creative

Strutture iterative

- Collezione di istruzioni ripetute in modo ciclico
- Esempi:
  - Sequential Search Algorithm
  - Insertion Sort Algorithm

Sequential search algorithm in pseudocode

```
def Search (Lista, ValoreCercato):
    if (Lista è vuota):
        Dichiara fallita la ricerca
    else:
        Scegli la prima voce di Lista come VoceTest
        while (ValoreCercato > VoceTest e
restano voci da considerare):
           Scegli la successive voce di Lista come VoceTest
        if (ValoreCercato == VoceTest):
            Dichiara riuscita la ricerca
        else:
            Dichiara fallita la ricerca
```

Componenti del controllo iterativo

Initialize: Establish an initial state that will be modified toward the termination condition

Test: Compare the current state to the termination condition and terminate the repetition if equal

Modify: Change the state in such a way that it moves toward the termination condition

Strutture iterative

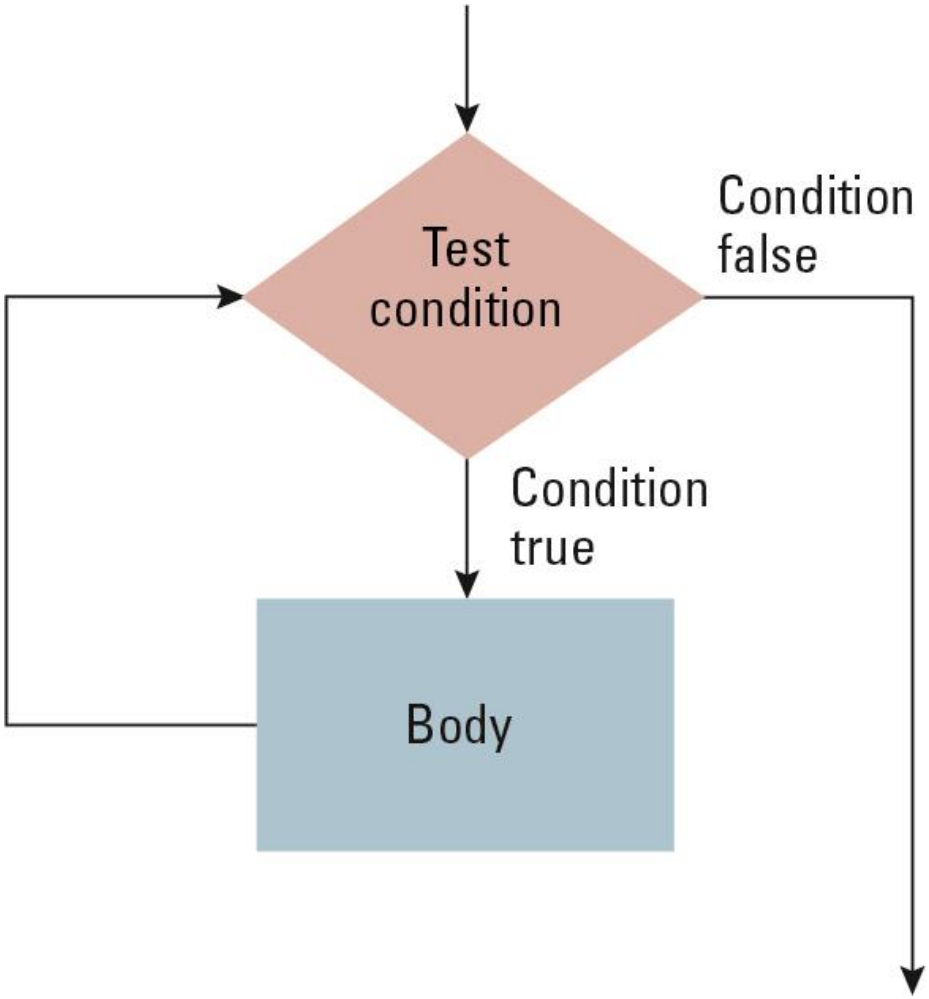
- Pre-test loop:

```
while (condition):
   body
```

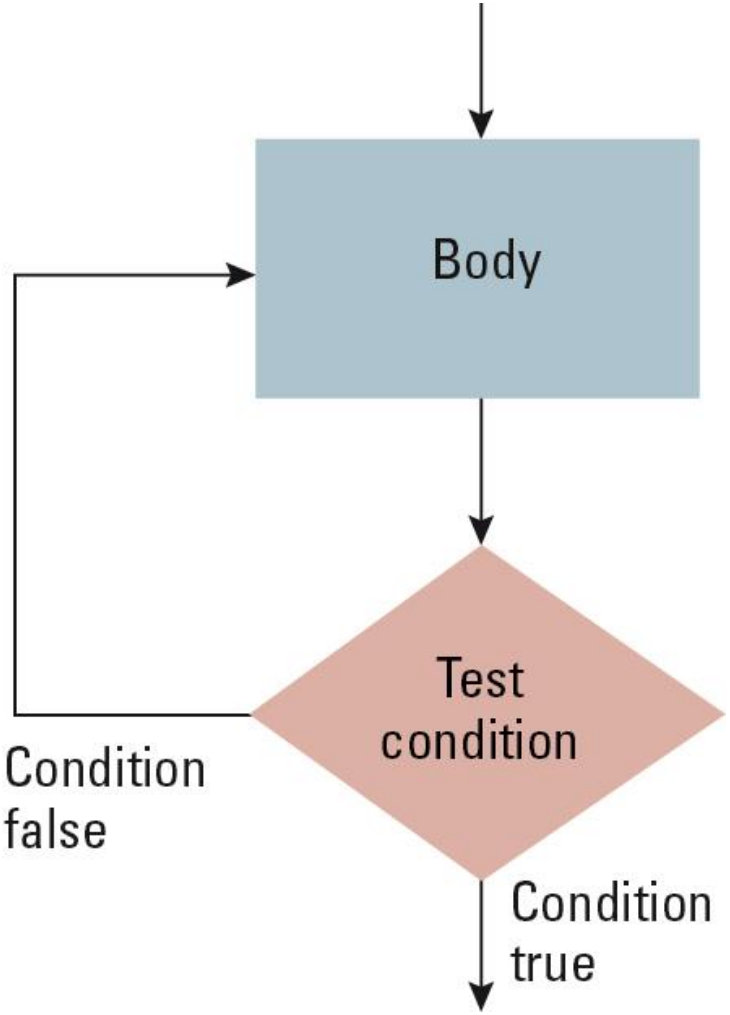
- Post-test loop:

```
repeat:
   body
   until(condition)
```

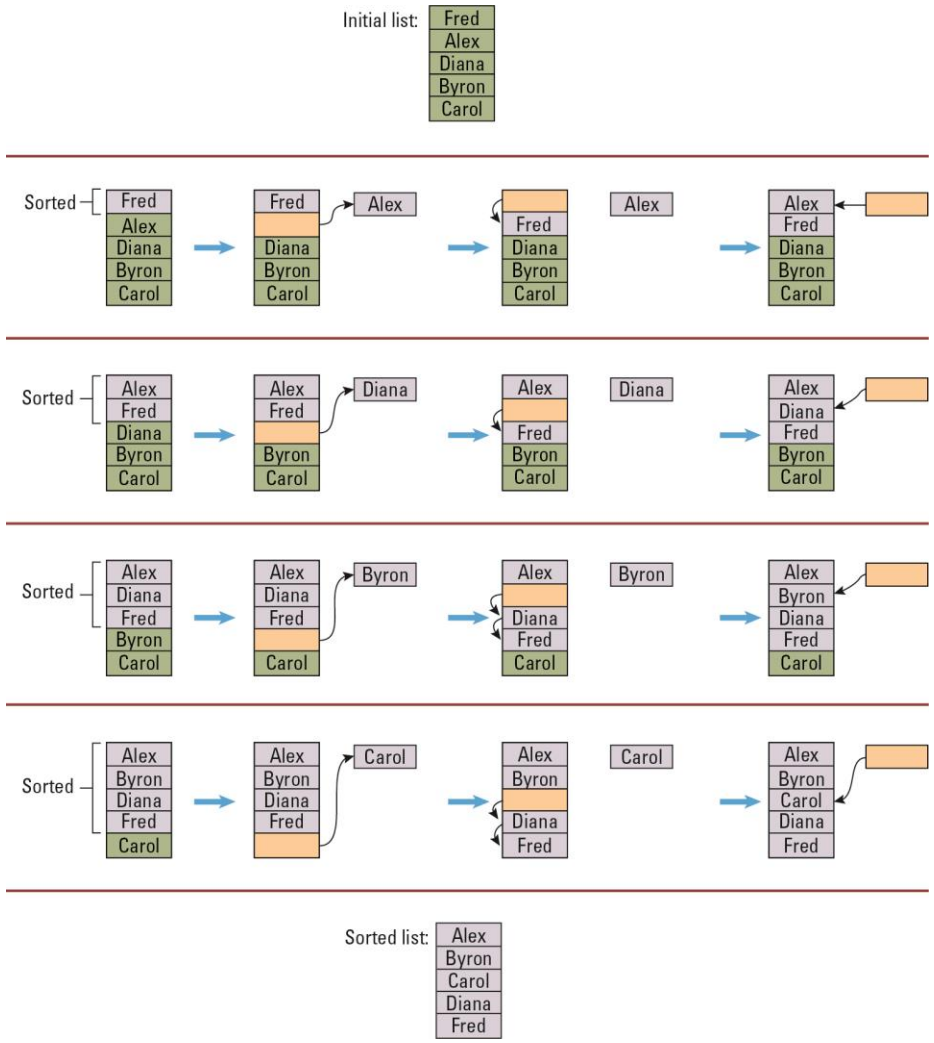
Struttura ciclo while



Struttura del ciclo repeat



Ordinamento alfabetico della lista Fred, Alex, Diana, Byron, e Carol



Insertion sort algorithm in pseudocode

```
def Sort(List):
    N = 2
    while (N <= lunghezza della lista):
      Pivot = Nth voce della lista
      Sposta voce Pivot lasciando un vuoto in List
      while (c'è un nome sopra il vuoto and
       Entry > Pivot):
          Sposta il nome nello spazio vuoto
          sottostante lasciando un vuoto sopra
          il nome
      Sposta la voce Pivot nello spazio vuoto in
          Lista
       N = N + 1
```

Efficienza degli algoritmi

- Misurato come il numero di istruzioni eseguite
- Usa la notazione big theta:
  - Esempio: Insertion sort è in $\Theta(n^2)$
- Incorpora l'analisi del caso migliore, peggiore e medio

Ricerca di un valore

C’è un dato autore in una sequenza ordinata di n libri?

- Ricerca lineare

- Ricerca binaria

Ricerca lineare

Caso fortunato

Asimov 1 confronto

Caso sfortunato

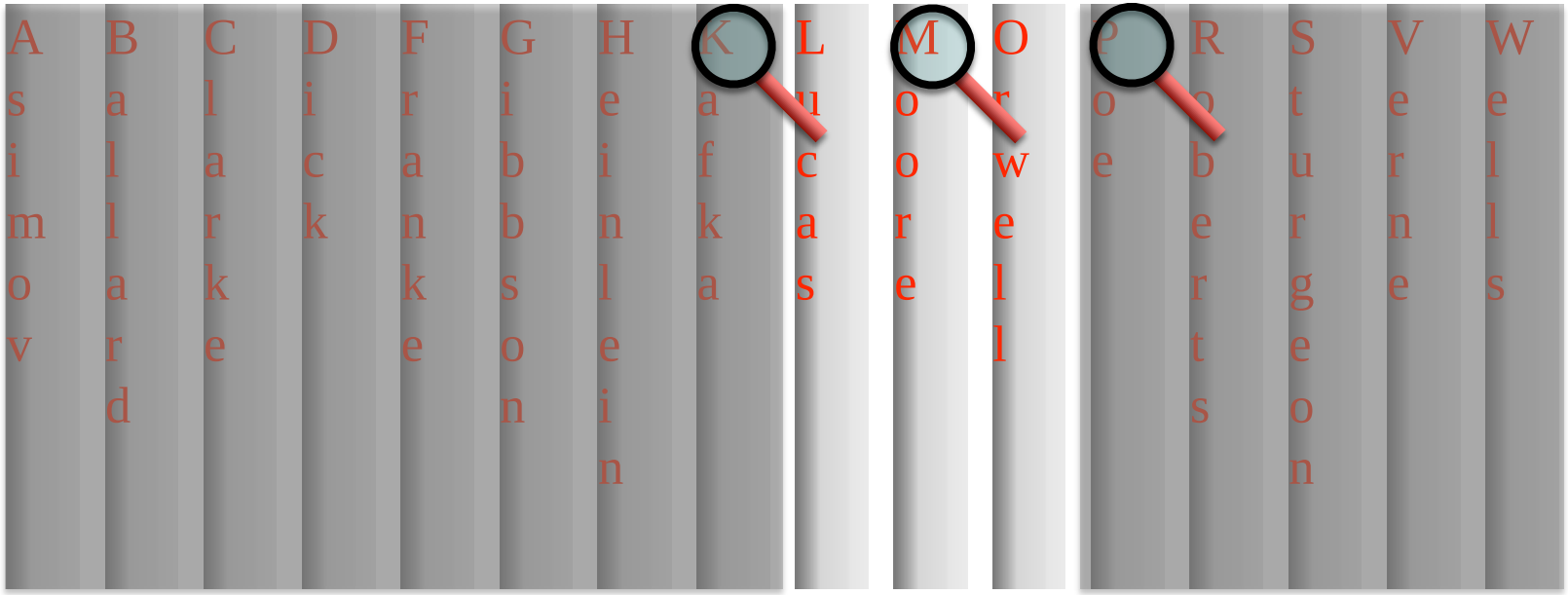
Wells 16 confronti

Caso intermedio

Kafka 8 confronti in media

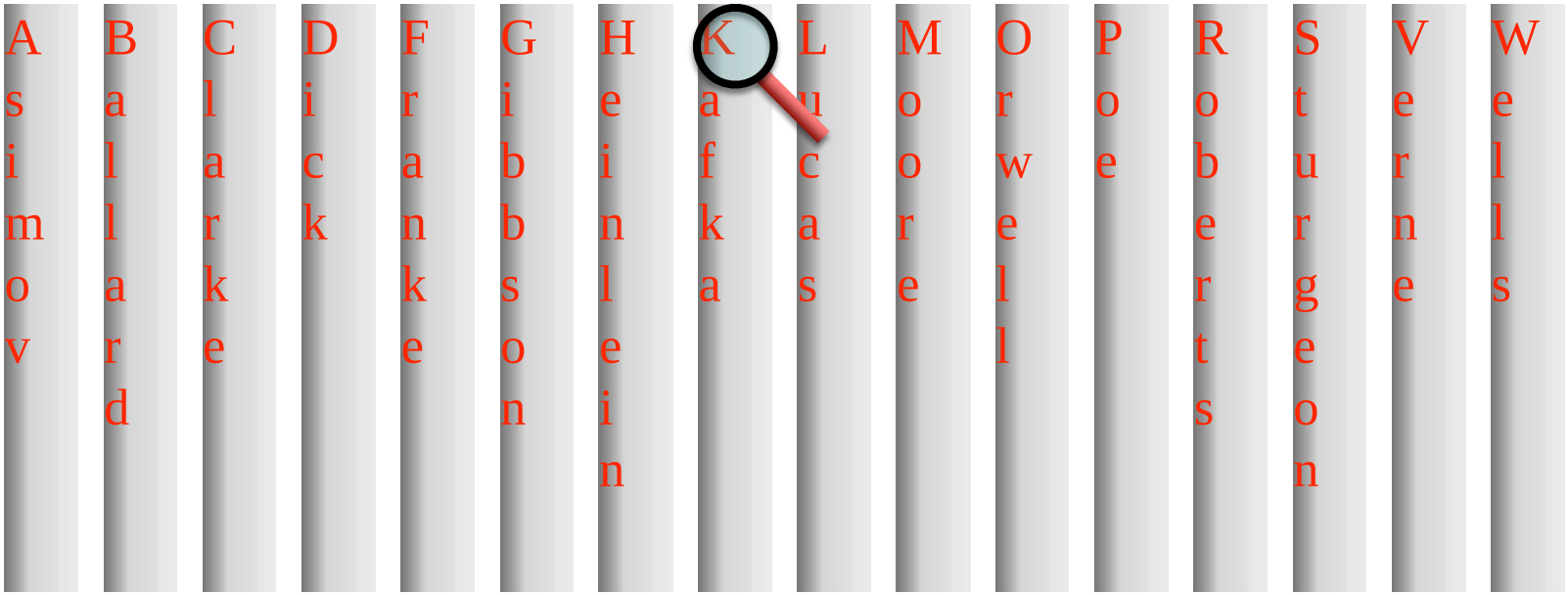
Ricerca dicotomica

Moore 3 confronti



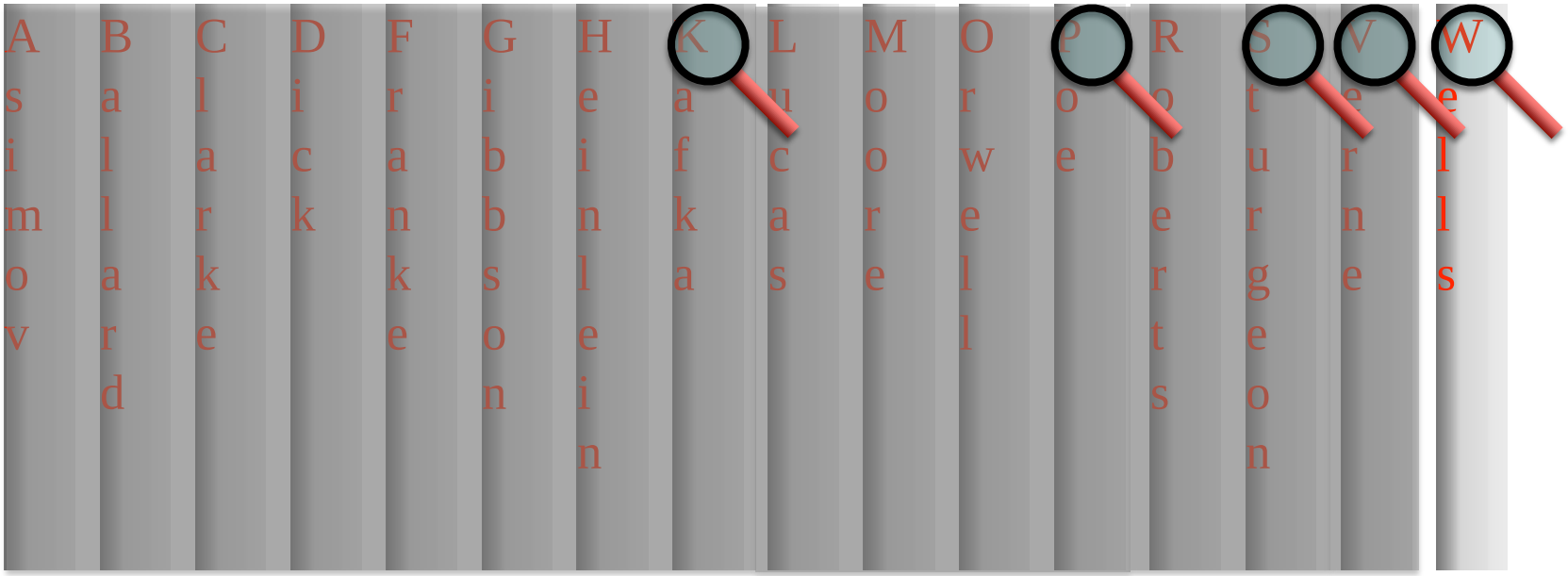
Caso fortunato

Kafka 1 confronto



Caso sfortunato

Wells 5 confronti



Quanto costa la binaria?

Migliore dei casi

valore centrale ⇒ 1 confronto

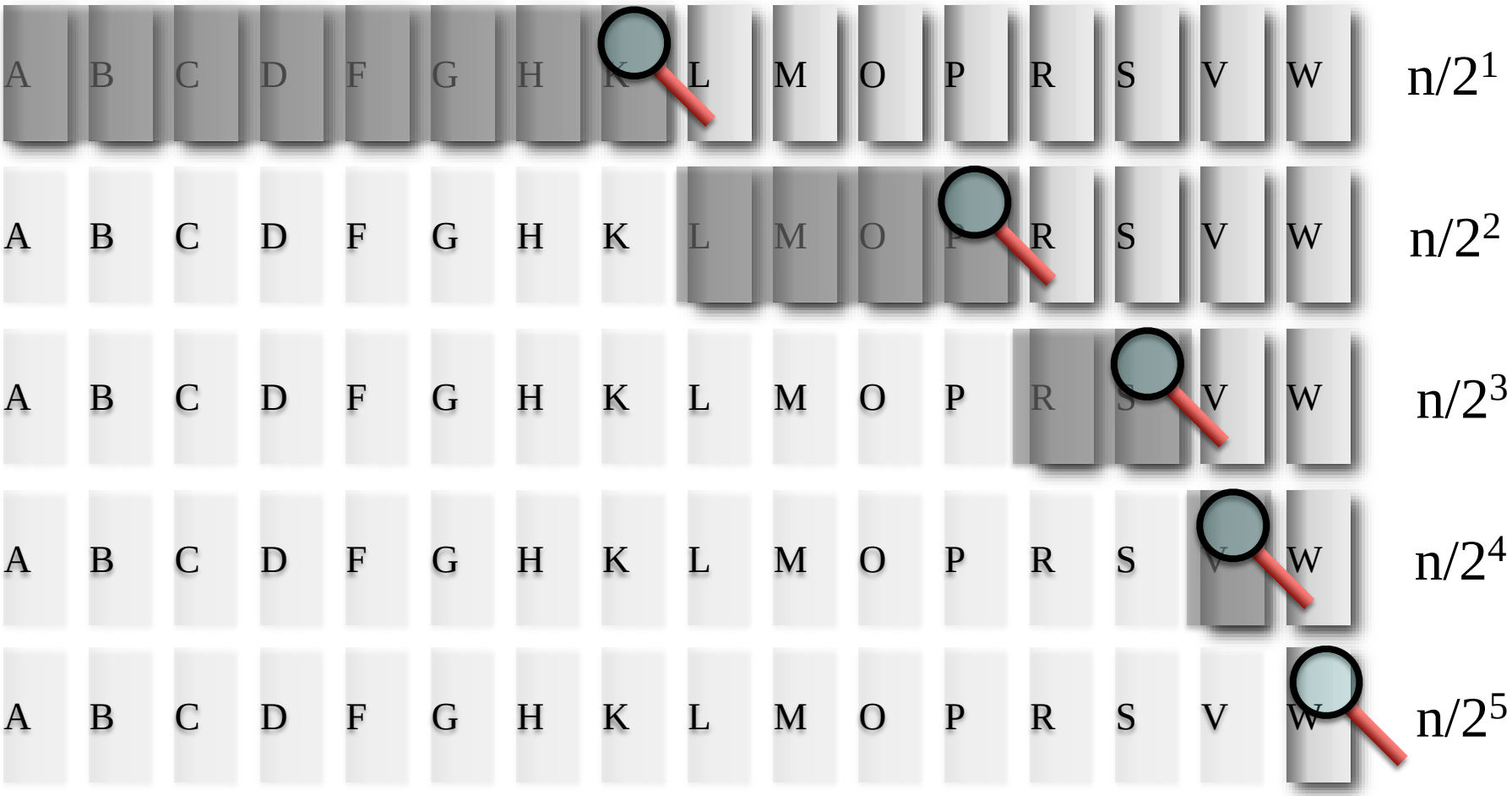
Peggiore dei casi

p.e. valore agli estremi ⇒ ? confronti

Caso medio

quando accade? ⇒ ? confronti

Divide et Impera



Confronto fra ricerche

caso	ricerca lineare	ricerca binaria
migliore	1	1
peggiore	n	$\log_2(n)$
medio	n/2	$\log_2(n)$

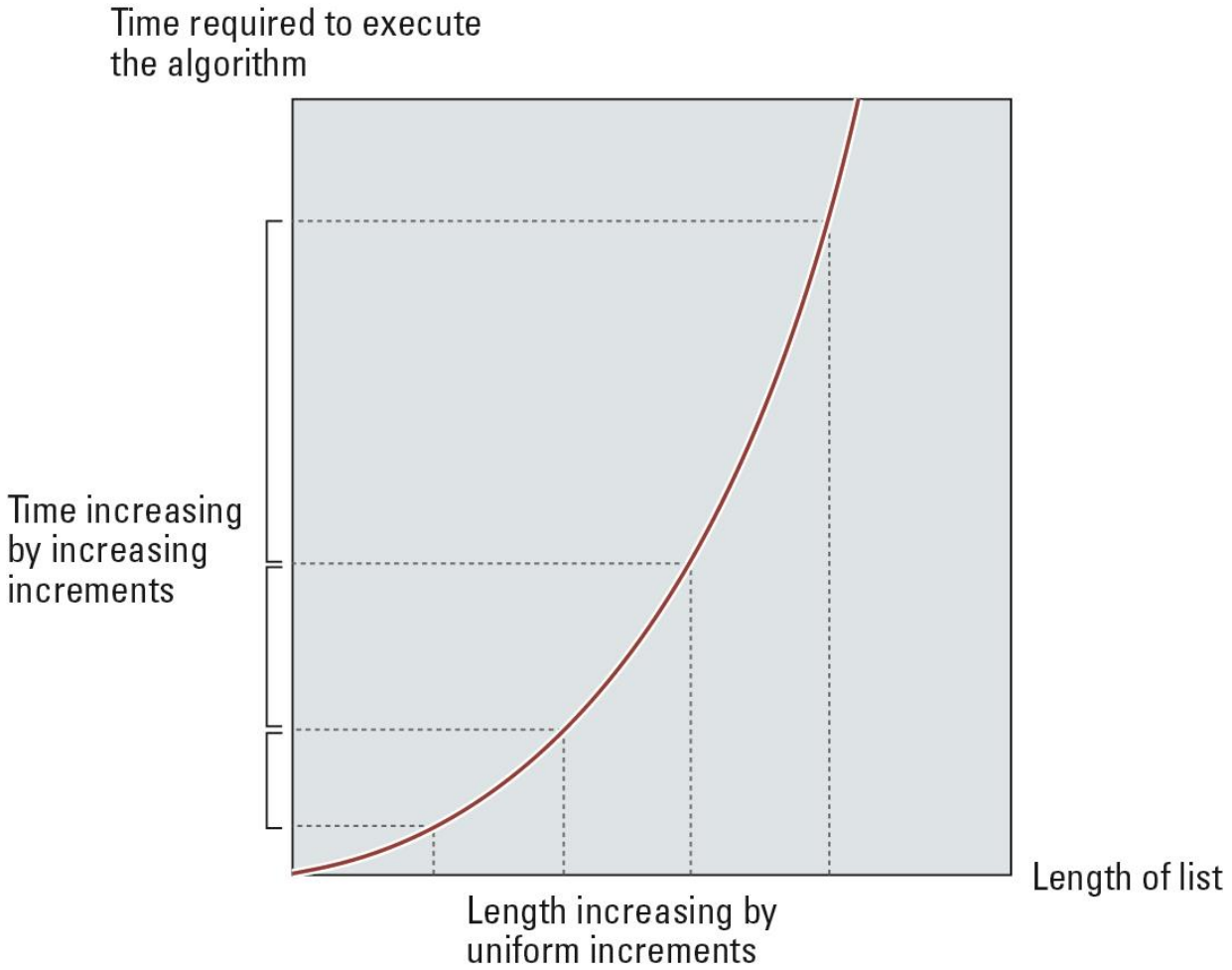
secondi	tempo	Log_2(secondi)
10^1	0.17 min	3.32
10^3	16.6 min	9.97
10^6	1.6 sett.	19.93
10^9	3.2 anni	29.9
10^{12}	31.7 millenni	
10^{15}	31.7 milioni di a	
10^{18}	2.3 età dell'universo	56.47

circa 4.3×10^{17} secondi

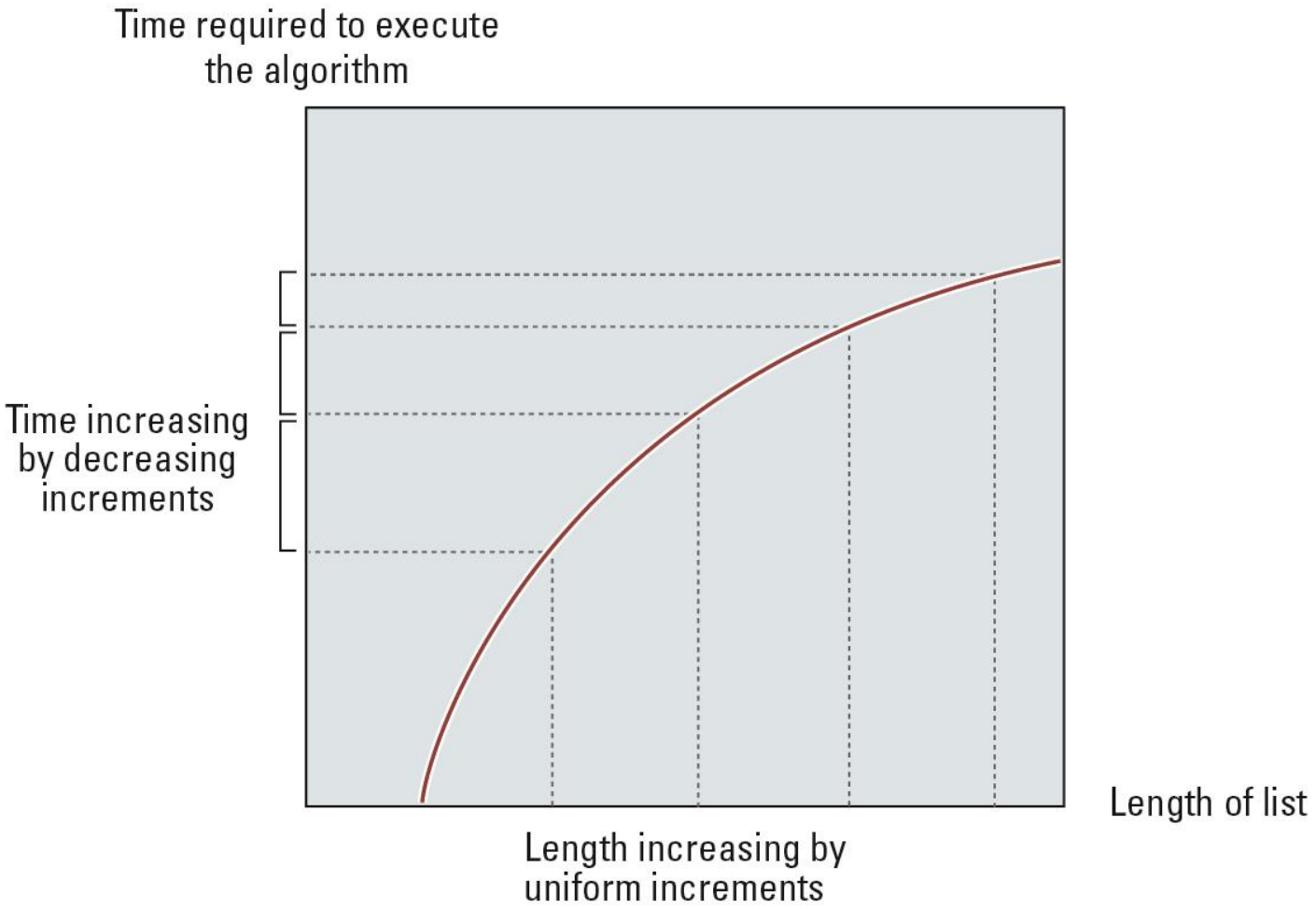
Applicazione dell'ordinamento per inserimento in una situazione di caso peggiore

Initial list	Comparisons made for each pivot				Sorted list
	1st pivot	2nd pivot	3rd pivot	4th pivot	
Elaine David Carol Barbara Alfred	1 Elaine David Carol Barbara Alfred	3 David 2 Elaine Carol Barbara Alfred	6 Carol 5 David 4 Elaine Barbara Alfred	10 Barbara 9 Carol 8 David 7 Elaine Alfred	Alfred Barbara Carol David Elaine

Algoritmo di ordinamento per inserimento: analisi del caso peggiore



Algoritmo di ricerca binaria: analisi del caso peggiore



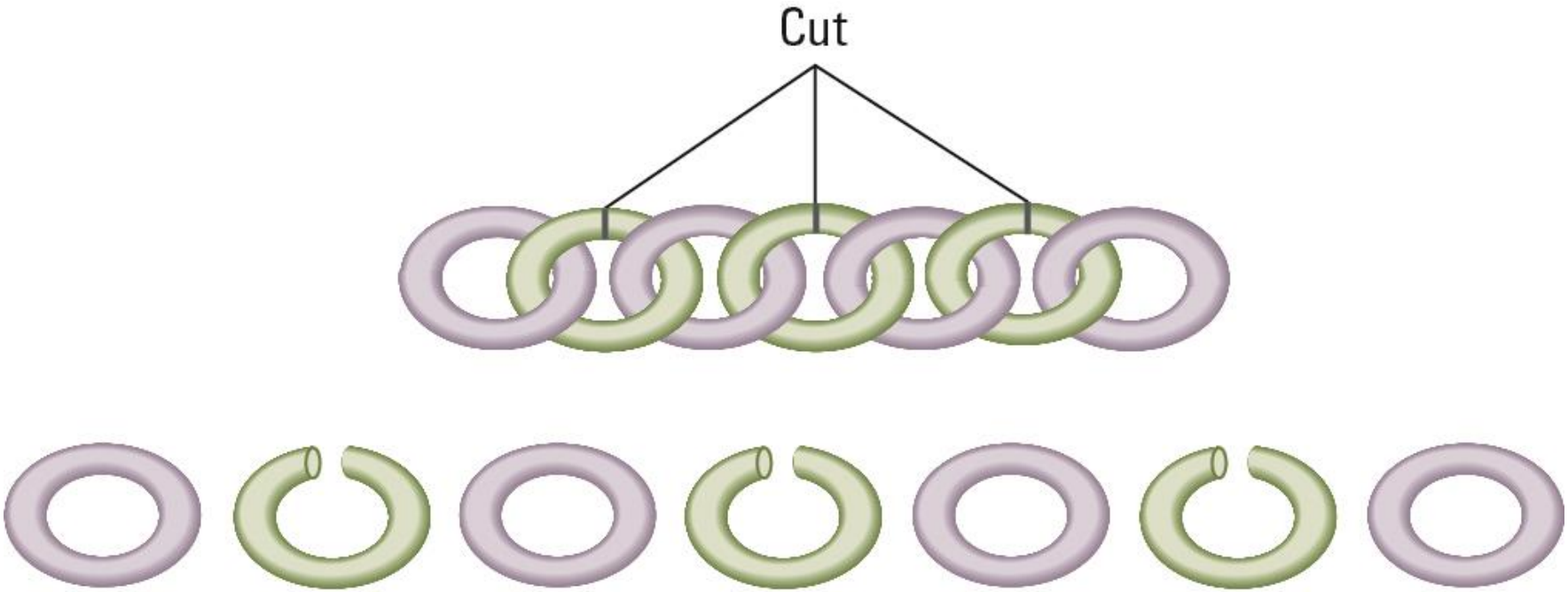
Verifica del software

- Prova di correttezza (con logica formale)
  - Asserzioni
    - Precondizioni
    - Invarianti di ciclo
- Il Testing è utilizzato per verificare il software
- Il test prova soltanto che il programma è corretto per i casi di test utilizzati.

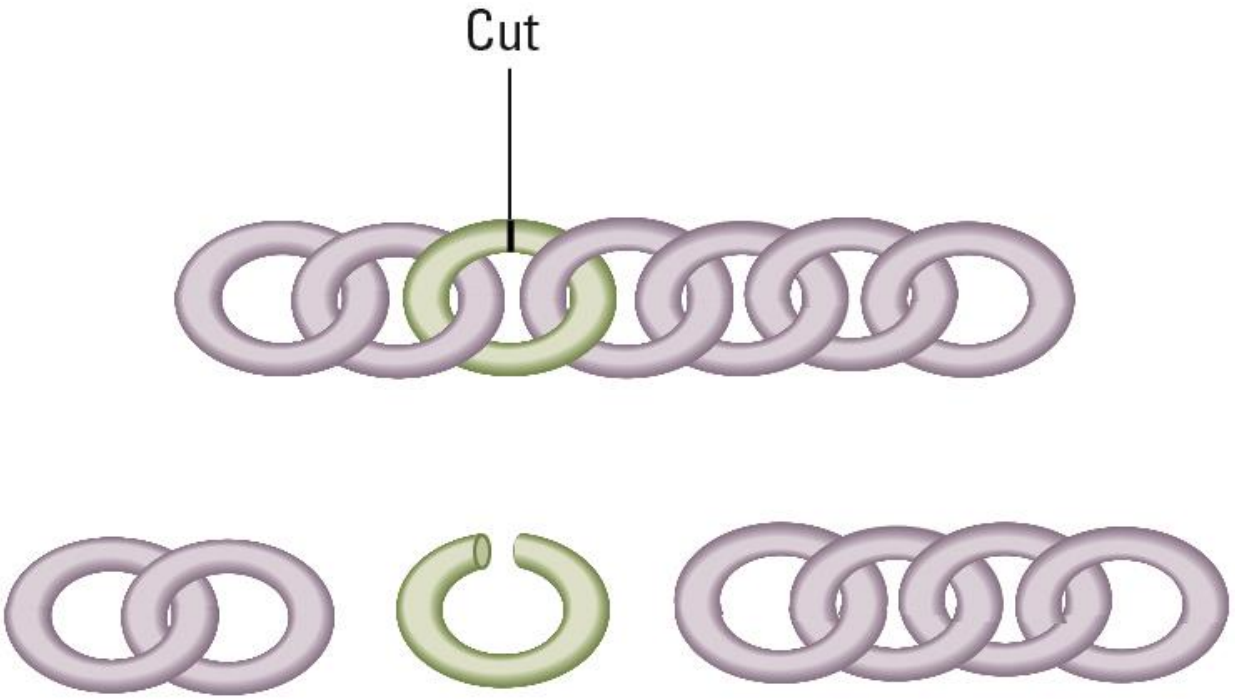
Chain Separating Problem

- Un viaggiatore con una catena d'oro di sette anelli deve alloggiare in un albergo isolato per sette notti.
- Il prezzo per ogni notte è un anello della catena.
- Qual è il numero minimo di anelli che devono essere rimossi dalla catena affinchè il viaggiatore possa pagare ogni mattina con un anello senza dover pagare in anticipo?

Separare la catena effettuando solo tre tagli



Risolvere il problema con un taglio



- 1^ mattina: paga con anello singolo

- 2^ mattina: recupera l'anello singolo e paga con pezzo a due anelli

- 3^ mattina: paga con l'anello singolo

- 4^ mattina: recupera i tre anelli e paga con il pezzo a quattro anelli

- 5^ mattina: paga con l'anello singolo

- 6^ mattina: recupera l'anello singolo e paga con il pezzo a due anelli

- 7^ mattina: paga con l'anello singolo

Asserzioni associate a una tipica struttura while