Problemi di Ricerca

Carla Binucci e Walter Didimo

Il problema della Ricerca

In generale, il Problema della Ricerca è definito come segue:

- data una collezione *C* di elementi e una proprietà *P*, determinare se *C* contiene almeno un elemento che verifica la proprietà *P*

Il problema della Ricerca - varianti

In pratica, diverse sono le varianti con cui si può presentare questo importante problema:

- varianti sul tipo di collezione
  - la collezione può essere un array, una sequenza, un insieme di elementi, oppure un dizionario (cioè, un insieme di coppie di elementi, di tipo termine-definizione)
- varianti sulla proprietà da verificare
  - ad esempio, la proprietà da verificare può essere l'uguaglianza tra l'elemento dato ed un elemento della collezione
- varianti sul risultato della ricerca
  - può essere l'esito booleano della verifica, oppure uno o tutti gli elementi trovati, oppure le loro posizioni

Il problema della Ricerca in un array

Il problema della ricerca in un array può essere definito in questo modo:

- dato un array *dati* di elementi di tipo *T* e un valore *chiave* di tipo *T* determinare se *dati* contiene un elemento il cui valore è uguale a *chiave*

- *chiave* è chiamato la chiave della ricerca

Studieremo il caso in cui *T* è il tipo *int*, mostrando diversi algoritmi e la relativa complessità

Dimensione del problema

I dati di ingresso per il problema della ricerca in un array sono gli *N* elementi dell'array *dati* più il valore *chiave*

- se *dati* ha *N* elementi, la dimensione del problema è *N+1*
- possiamo trascurare la presenza di *chiave* ed assumere che la dimensione del problema sia *N*

Complessità del problema

La complessità asintotica del problema della ricerca in un array è lineare nel numero di elementi dell'array

- non esiste alcun algoritmo asintoticamente migliore di quello che confronta la chiave della ricerca con ciascun elemento dell'array
- infatti, nel caso peggiore (ovvero *dati* non contiene alcun elemento uguale a *chiave*), dobbiamo comunque confrontare la chiave della ricerca con ciascun elemento dell'array

Algoritmo di ricerca sequenziale esaustiva

```
/* ricerca in un array di N interi */
public static boolean ricerca(int[] dati, int chiave) {
    int i;                  // per la scansione di dati
    boolean trovato;        // esito della ricerca
    trovato = false;
    for (i=0; i<dati.length; i++)
        if (dati[i]==chiave)
            trovato = true;
    return trovato;
}
```

In questo algoritmo:

- la ricerca è sequenziale, poiché gli elementi dell'array vengono scanditi uno dopo l'altro
- la ricerca è esaustiva, poiché tutti gli elementi dell'array vengono comunque confrontati con la *chiave*

Questo algoritmo ha complessità asintotica *O(N)*

Algoritmo di ricerca sequenziale

```
/* ricerca in un array di N interi */
public static boolean ricerca(int[] dati, int chiave) {
    int i;                  // per la scansione di dati
    boolean trovato;        // esito della ricerca
    trovato = false;
    i = 0;
    while (!trovato && i<dati.length) {
        if (dati[i]==chiave)
            trovato = true;
        i++;
    }
    return trovato;
}
```

In questo algoritmo:

- la ricerca è sequenziale, ma se viene trovato un elemento uguale a *chiave* allora la ricerca viene interrotta (la ricerca non è esaustiva)

Questo algoritmo ha comunque complessità asintotica *O(N)*

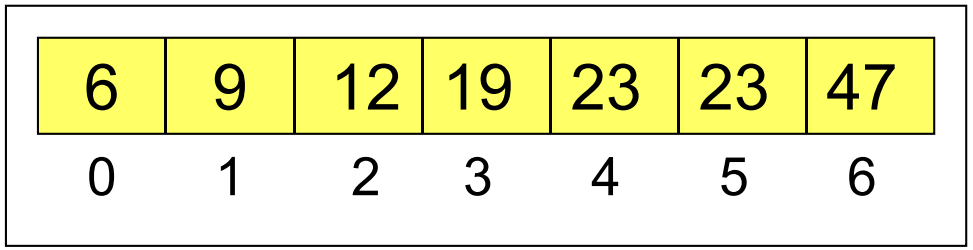
La Ricerca in un array ordinato

Il problema della ricerca può essere risolto in modo più efficiente se sono verificate delle ipotesi sull'organizzazione degli elementi nell'array:

- studiamo il problema della ricerca in un array nell'ipotesi che gli elementi dell'array siano disposti in modo ordinato

Array ordinati

Un array è ordinato in modo non decrescente se, per ogni indice *i*, l'elemento di indice *i* è maggiore o uguale di tutti gli elementi di indice *j*, con *j*<*i*



Ricerca in un array ordinato

Il problema della ricerca in un array ordinato è definito come segue:

- dato un array *dati* di elementi interi ordinati in modo non decrescente ed un intero *chiave*, determinare se *chiave* è un elemento di *dati*

Per risolvere il problema potremmo ovviamente utilizzare i metodi di ricerca in un array non ordinato

- possiamo allora certamente dire che la complessità asintotica di questo problema è al massimo lineare
- possiamo trarre vantaggio dal fatto che l'array è ordinato?

Array ordinati – ricerca sequenziale

Nella ricerca sequenziale, possiamo interrompere la ricerca se durante la scansione dell'array viene incontrato un elemento uguale o maggiore alla chiave della ricerca (il costo nel caso peggiore è ancora *O(N)*)

```
/* ricerca in un array ordinato */
public static boolean ricercaOrd(int[] dati, int chiave) {
    int i;                  // per la scansione di dati
    boolean trovato = false; // esito della ricerca
    boolean finito = false;  // incontrato elemento maggiore
    i = 0;
    while (!trovato && !finito && i<dati.length) {
        if (dati[i]==chiave)
            trovato = true;
        else if (dati[i]>chiave)
            finito = true;
        i++;
    }
    return trovato;
}
```

Spazio di ricerca

In un algoritmo di ricerca gli elementi dell'array vengono partizionati in due insiemi:

- un insieme è composto dagli elementi che sono sicuramente diversi dalla chiave di ricerca

- l'altro insieme, chiamato spazio di ricerca, è composto dagli elementi che potrebbero essere uguali alla chiave di ricerca

Strategia di ricerca

Ad ogni passo, un algoritmo di ricerca confronta la chiave della ricerca con un elemento dell'array tra quelli che ancora fanno parte dello *spazio di ricerca*

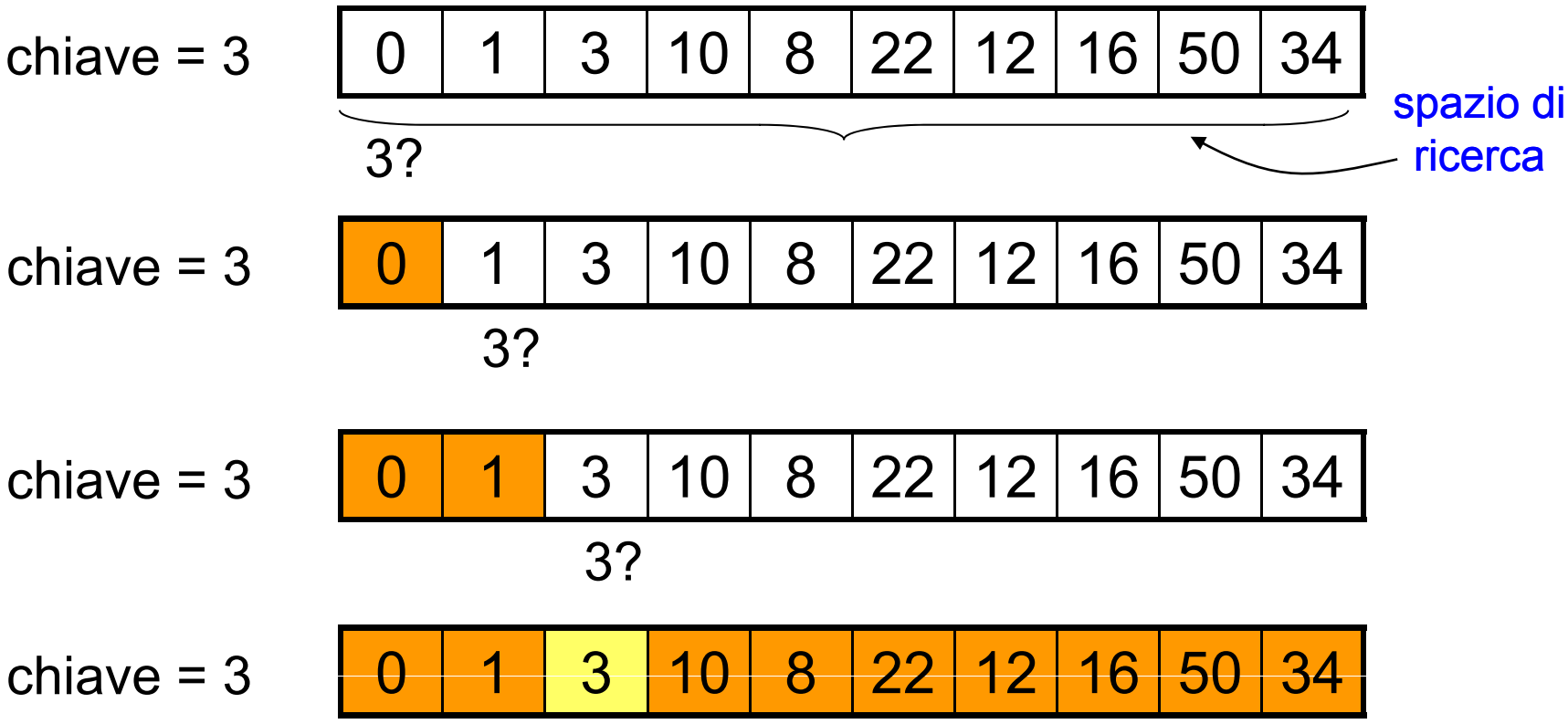
- inizialmente lo *spazio di ricerca* è l'intero array

Ogni confronto tra la chiave della ricerca e un elemento dello spazio di ricerca può:

- trovare l'elemento e terminare la ricerca
- ridurre lo spazio di ricerca

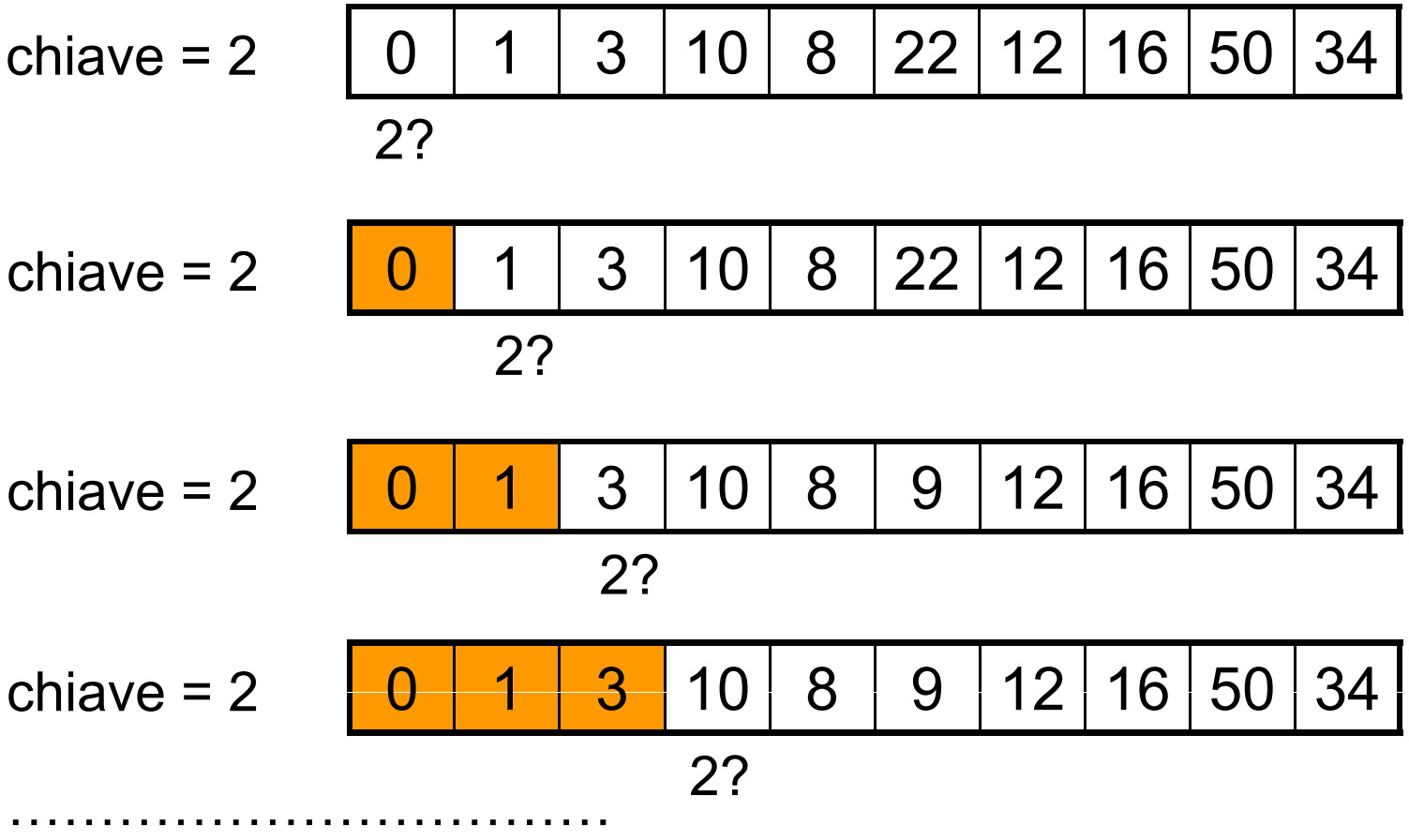
Spazio di ricerca nella ricerca sequenziale su array generici – esito positivo

Comportamento degli algoritmi di ricerca sequenziale per array generici (e per array ordinati) se la chiave appartiene all'array



Spazio di ricerca nella ricerca sequenziale su array generici – esito negativo

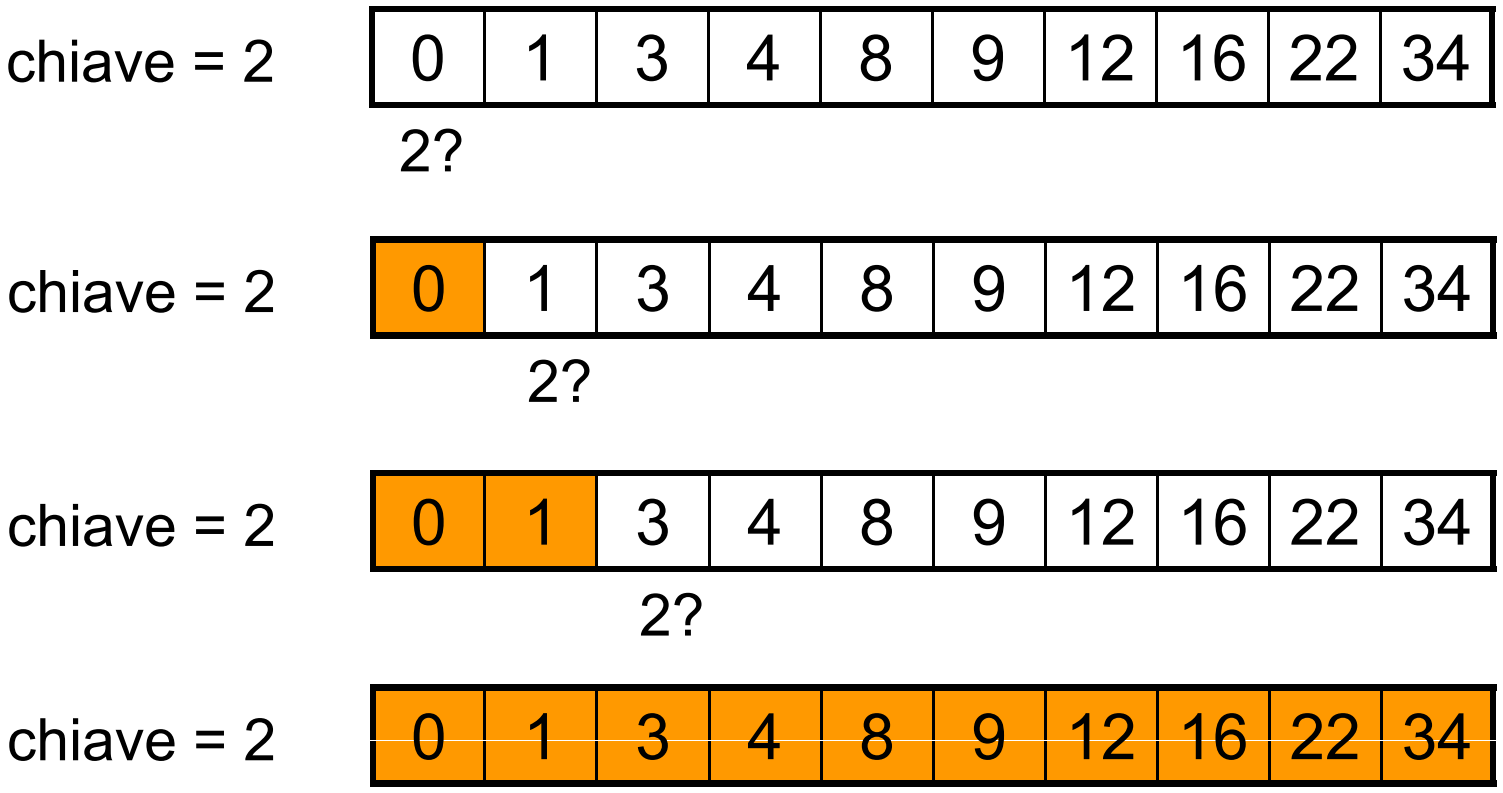
Comportamento dell'algoritmo di ricerca sequenziale per array generici se la chiave non appartiene all'array



……

Spazio di ricerca nella ricerca sequenziale su array ordinati – esito negativo

Comportamento dell'algoritmo di ricerca sequenziale per array ordinati se la chiave non appartiene all'array



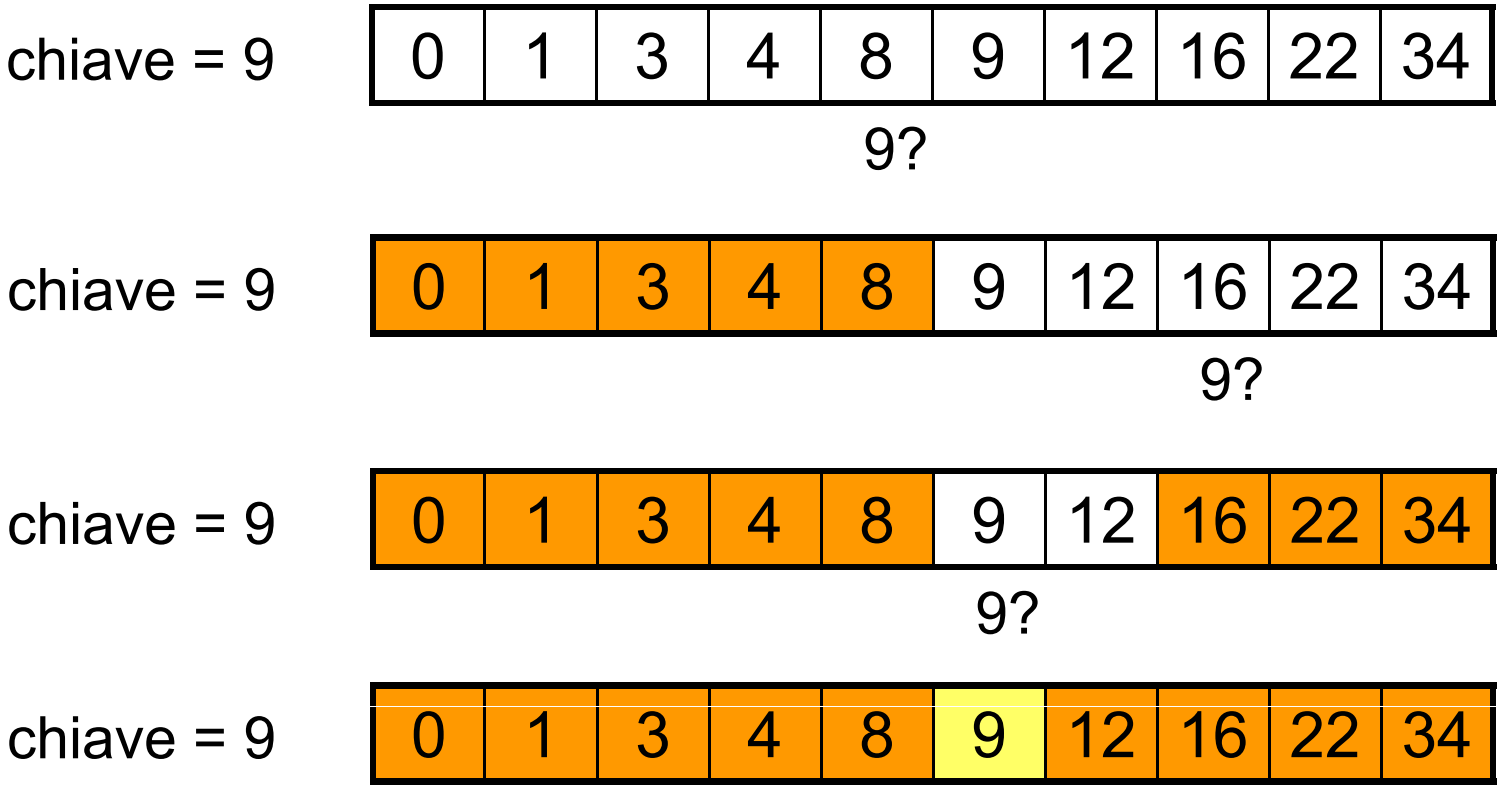
Ricerca binaria (o dicotomica)

La strategia di ricerca binaria (o dicotomica) cerca di favorirsi il più possibile se il confronto viene fatto con un elemento diverso dalla chiave

- ad ogni passo, la chiave viene confrontata con *l'elemento centrale* dello spazio di ricerca:
  - se la chiave è stata trovata, la ricerca termina
  - se la chiave è diversa dall'elemento, lo spazio di ricerca viene dimezzato:
    - se la chiave è maggiore dell'elemento, allora lo spazio di ricerca diventa l'insieme degli elementi alla sua destra, altrimenti quello degli elementi alla sua sinistra

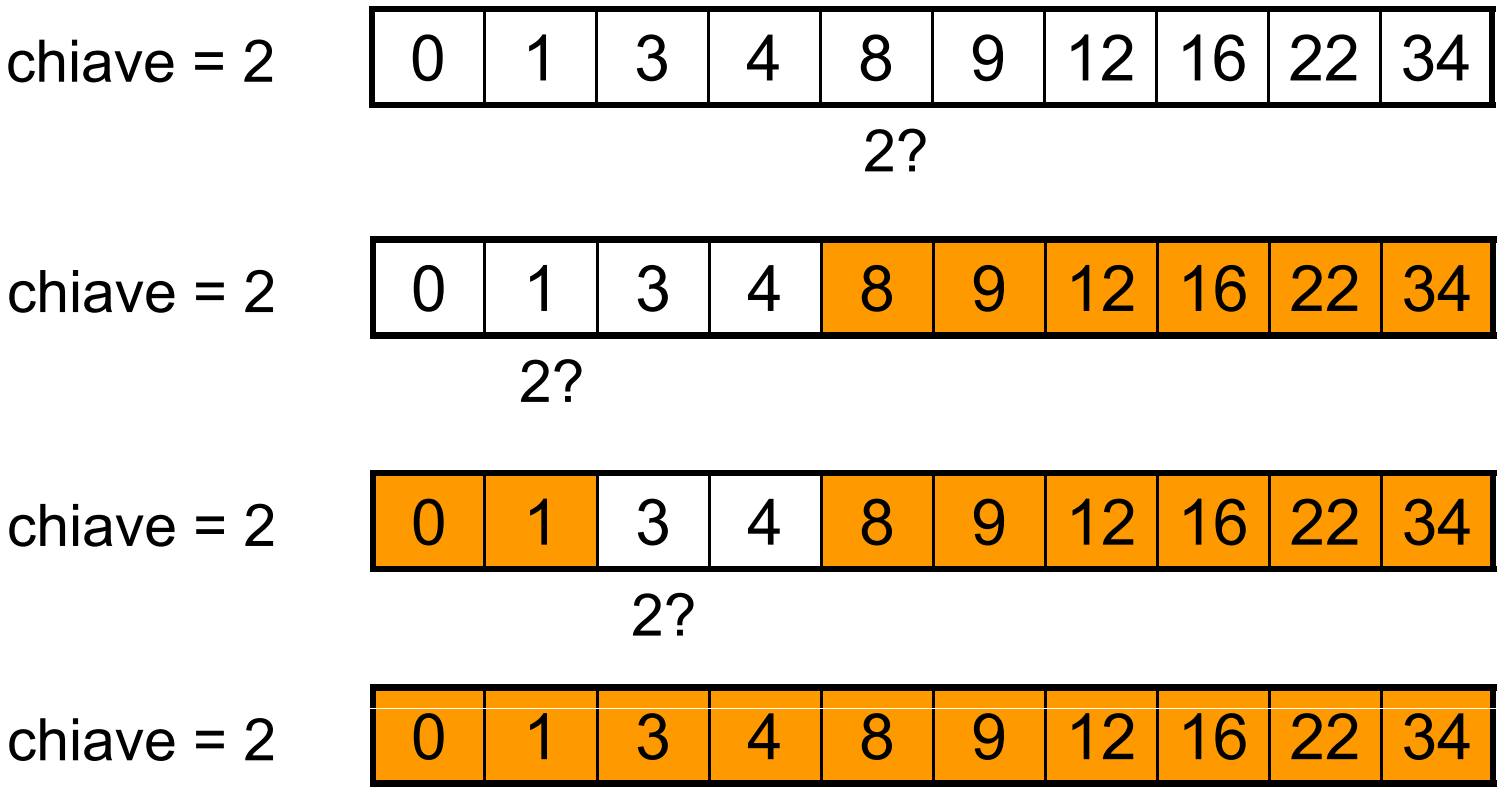
Esempio - chiave appartiene all'array

Comportamento della ricerca binaria nel caso in cui l'array contenga un elemento uguale a chiave



Esempio - chiave non appartiene all'array

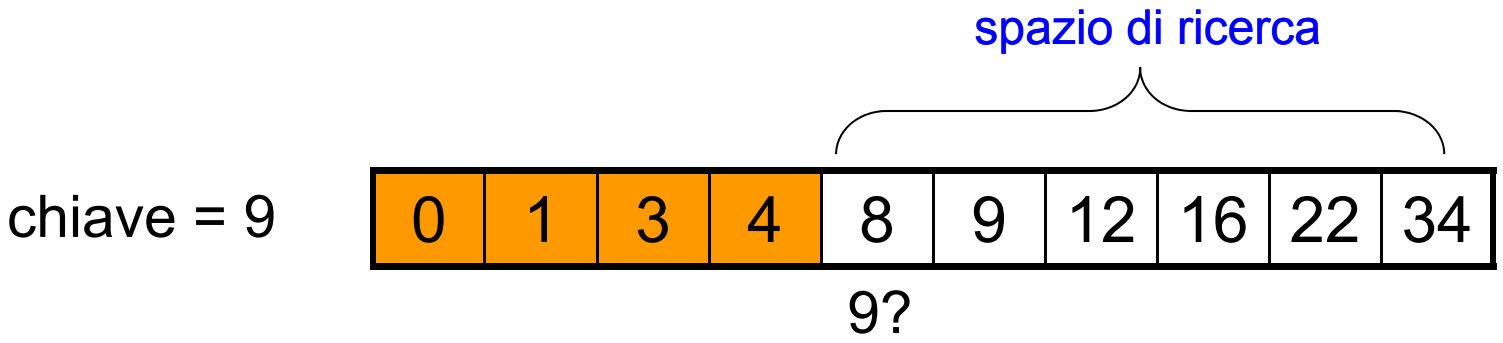
Comportamento della ricerca binaria nel caso in cui chiave non appartiene all'array



Descrizione dello spazio di ricerca

Nella ricerca esaustiva, lo spazio di ricerca è delimitato

- a sinistra dall'indice dell'elemento corrente
- a destra dalla lunghezza di dati (diminuita di uno)



Come si descrive lo spazio di ricerca nella ricerca binaria ?

Spazio di ricerca nella ricerca binaria

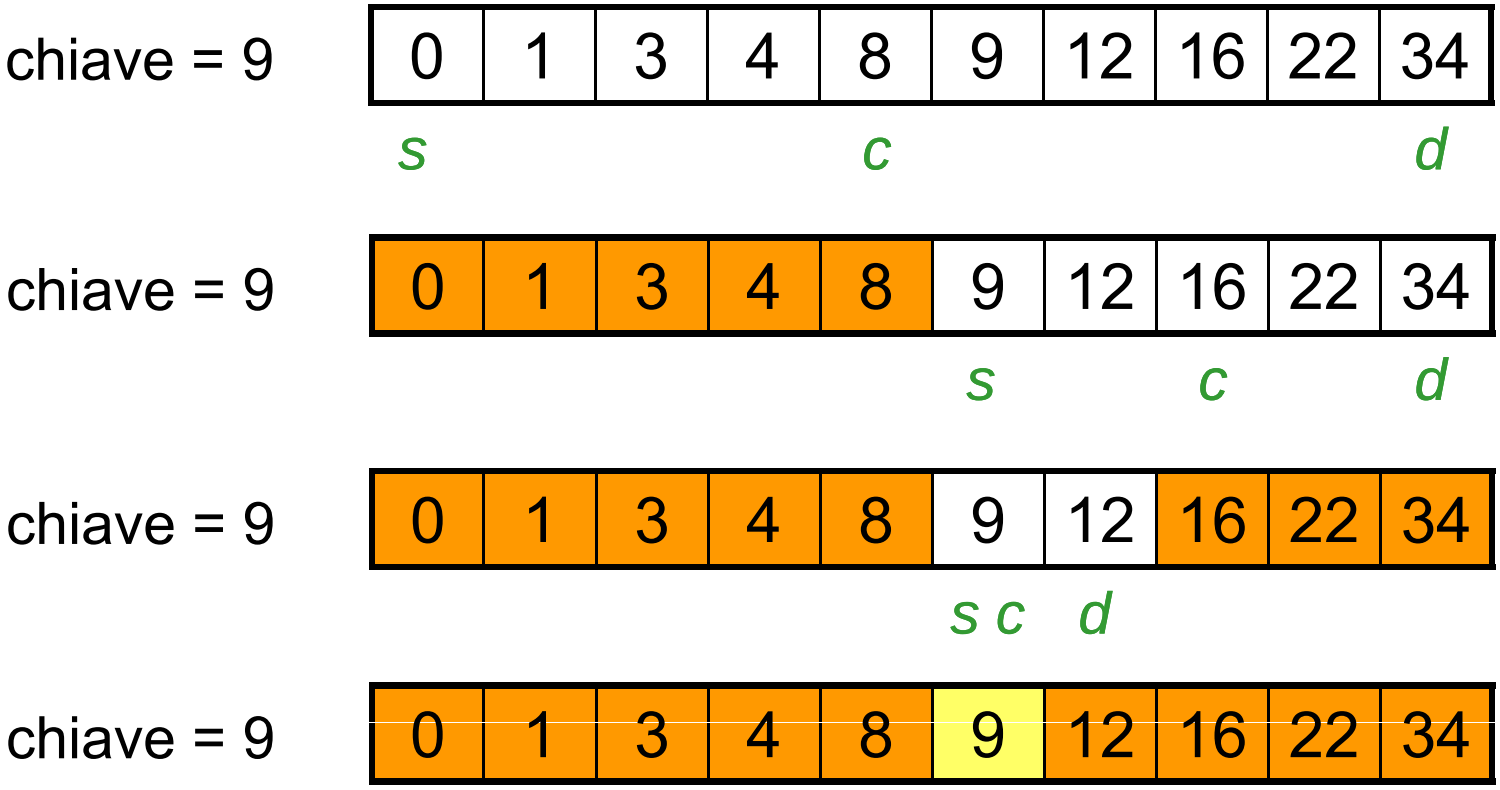
Nella ricerca binaria, lo spazio di ricerca è sempre connesso

- possiamo allora descrivere lo spazio di ricerca mediante una coppia di variabili
  - *sinistra* è l'indice del primo elemento dello spazio residuo di ricerca
  - *destra* è l'indice dell'ultimo elemento dello spazio residuo di ricerca
  - lo spazio di ricerca contiene gli elementi di indice compreso tra sinistra e destra
  - inoltre, *centro* è l'indice dell'elemento centrale pari a *(sinistra+destra)/2*

Spazio di ricerca nella ricerca binaria

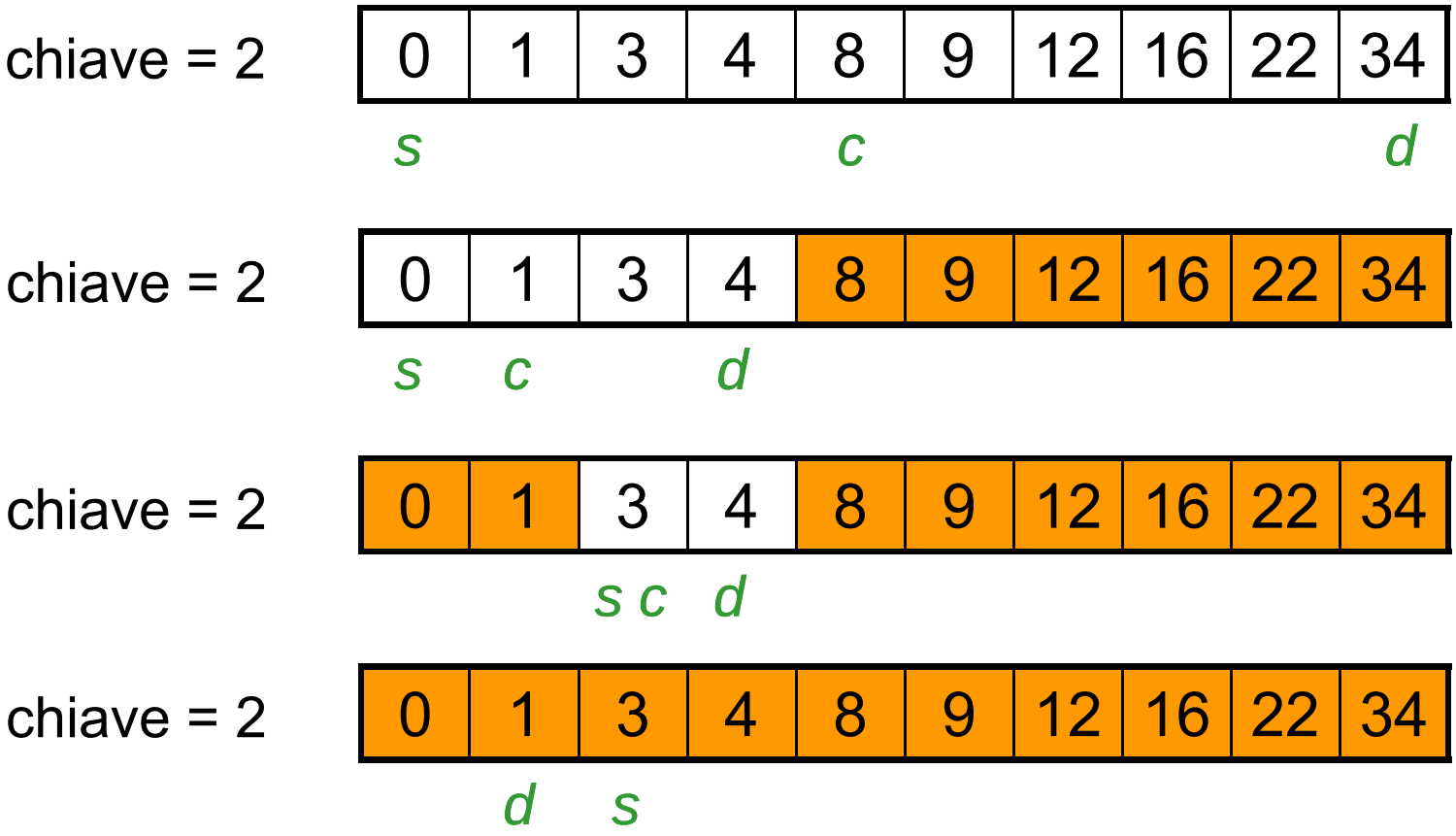
Ecco il ruolo delle variabili *destra* (*d*), *sinistra* (*s*) e *centro* (*c*)

- *chiave* appartiene all'array



Spazio di ricerca nella ricerca binaria

- *chiave* non appartiene all'array



- la ricerca fallisce se *sinistra* diventa maggiore di *destra*

Metodo per la ricerca binaria

```
/* ricerca binaria in un array ordinato di interi */
public static boolean ricBinaria(int[] dati, int chiave) {
    int sinistra = 0;
    int destra = dati.length - 1;
    int centro;
    boolean trovato = false;
    while (!trovato && (sinistra<=destra)) {
        centro = (sinistra+destra)/2;
        if (dati[centro]==chiave)
            trovato = true;
        else if (dati[centro]>chiave)
            /* devo continuare a sinistra di centro */
            destra = centro-1;
        else  // dati[centro]<chiave
            /* devo continuare a destra di centro */
            sinistra = centro+1;
    }
    return trovato;
}
```

Complessità della ricerca binaria

Quale è l'operazione dominante del metodo di ricerca binaria?

- una *operazione dominante* del metodo è il confronto dati[centro]==chiave tra la chiave e l'elemento centrale dello spazio di ricerca

Quale è il caso peggiore?

- il *caso peggiore* (dal punto di vista asintotico) si presenta quando l'array non contiene alcun elemento uguale alla chiave

Per determinare la complessità della ricerca binaria, bisogna determinare quante volte viene eseguita, nel caso peggiore, l'operazione dominante

- ovvero, bisogna determinare quante volte viene eseguito il corpo del *while*

Complessità della ricerca binaria

Quante volte viene eseguita la condizione (o il corpo) del *while* nel caso peggiore?

- il corpo del *while* viene eseguito fintanto che lo spazio di ricerca è non vuoto

- la complessità della ricerca binaria può essere quindi determinata contando il numero di esecuzioni del corpo del *while* necessarie a svuotare completamente lo spazio di ricerca nel caso peggiore

Complessità della ricerca binaria

- Inizialmente lo spazio di ricerca ha dimensione *N*
- l'esecuzione del corpo del *while* su uno spazio di ricerca di dimensione K riduce la dimensione dello spazio di ricerca almeno a K/2
- dopo H esecuzioni del corpo del *while*, lo spazio di ricerca si è ridotto da N almeno a $N/2^H$
- lo spazio di ricerca è sicuramente vuoto dopo W esecuzioni del corpo del *while*, dove W è il più piccolo intero tale che $N/2^W < 1$
- nel caso peggiore, il corpo del *while* viene eseguito $\lceil \log_2 N \rceil$ volte

La ricerca binaria ha quindi complessità asintotica *O(log N)*

Metodo ricorsivo per la ricerca binaria

L'algoritmo di ricerca binaria può essere anche implementato in modo ricorsivo

- si realizza un metodo ricorsivo che effettua la ricerca binaria entro lo spazio di ricerca delimitato dai parametri *sinistra* e *destra*, e un metodo per avviare la ricorsione

```
/* ricerca binaria in un array ordinato di interi
 * implementazione ricorsiva */
public static boolean ricBinaria(int[] dati, int chiave){
    return ricBinRic(dati, chiave, 0, dati.length-1);
}
```

Metodo ricorsivo per la ricerca binaria

```
/* ricerca binaria entro lo spazio di ricerca */
private static boolean ricBinRic(int[] dati, int chiave,
                                 int sinistra, int destra){
    int centro;
    boolean trovato;
    if (sinistra<=destra) {
        centro = (sinistra+destra)/2;
        if (dati[centro]==chiave)
            trovato = true;
        else if (dati[centro]>chiave)
            trovato =
                ricBinRic(dati, chiave, sinistra, centro-1);
        else // dati[centro]<chiave
            trovato =
                ricBinRic(dati, chiave, centro+1, destra);
    } else
        trovato = false;
    return trovato;
}
```

Glossario dei termini principali

Termine	Significato
Ricerca di un elemento in un array	Problema che consiste nel verificare se un array contiene almeno un elemento uguale ad un elemento dato chiamato chiave di ricerca
Ricerca sequenziale in un array	Strategia di ricerca nella quale la chiave di ricerca viene confrontata con gli elementi dell'array scandendoli sequenzialmente dal primo verso l'ultimo
Array ordinato (in modo non decrescente)	Array ordinato in modo tale che ogni suo elemento è maggiore o uguale agli elementi che lo precedono
Spazio di ricerca	E' la porzione di collezione (array) che è candidata a contenere la chiave della ricerca
Ricerca binaria o dicotomica	Strategia di ricerca per array ordinati nella quale la chiave viene confrontata ad ogni passo con l'elemento centrale dello spazio di ricerca: se la chiave è stata trovata la ricerca termina; se la chiave non è stata trovata, viene dimezzato lo spazio di ricerca