

Algoritmi di ricerca (*search algorithms*)

Liceo G.B. Brocchi - Bassano del Grappa (VI)
Liceo Scientifico - opzione scienze applicate
Giovanni Mazzocchin

Motivazioni

- Il nostro obiettivo è **cercare** la prima occorrenza di un elemento all'interno di una struttura dati:
 - l'unica struttura dati che conosciamo è l'**array**
 - per semplicità lavoreremo su array di interi
 - nella realtà si lavora spesso sugli array o su altre strutture dati, contenenti però dati molto più complessi dei semplici interi
- Questo problema ha innumerevoli applicazioni in Informatica. Alcuni esempi:
 - ricerca di un record in un database
 - ricerca di un file in un file system
 - ricerca di un gene in un genoma sequenziato
 - ricerca di una parola all'interno di un dizionario
 - ricerca di un'impronta digitale all'interno di un archivio di impronte di pregiudicati

Ricerca lineare (*linear search*)

6	5	9	23	5	7	6	7	78	4	56	98
---	---	---	----	---	---	---	---	----	---	----	----

- ipotizziamo di dover cercare la prima occorrenza della chiave 7 e memorizzare l'indice dell'elemento che la contiene. Qual è il modo più ovvio per farlo?

```
const int items_size = 12;
int items[items_size] = {6, 5, 9, 23, 5, 7, 6, 7, 78, 4, 56, 98};
bool found = false;
int index = -1;
int item = 7;

for (int i = 0; i < items_size && found == false; i++) {
    if (items[i] == item) {
        found = true;
        index = i;
    }
}
```

Ricerca lineare

6	5	9	23	5	0	6	-2	78	4	-56	98
---	---	---	----	---	---	---	----	----	---	-----	----

- se l'elemento cercato non è presente, la ricerca lineare restituisce:
 - index: **-1**
 - found: **false**

E se l'array fosse ordinato?

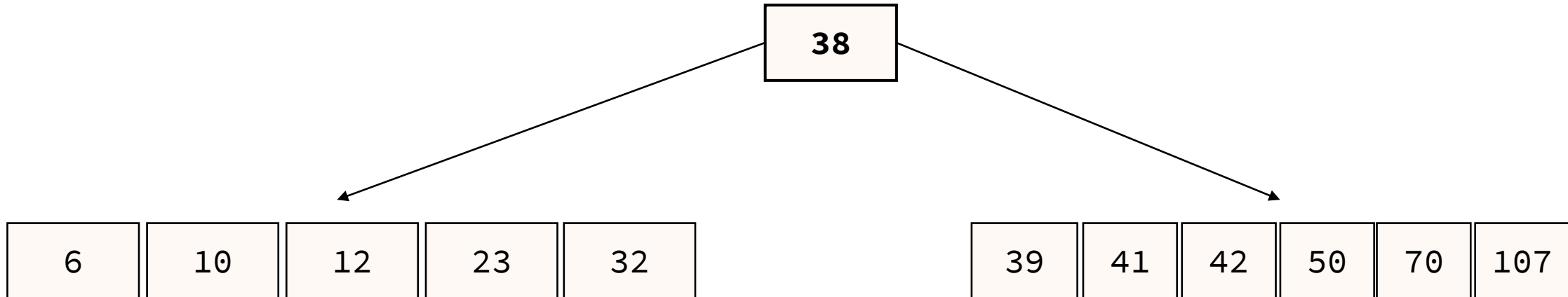
6	10	12	23	32	38	39	41	42	50	70	107
---	----	----	----	----	----	----	----	----	----	----	-----

- L'array è **ordinato** in senso crescente
- Cerchiamo di visualizzare l'array in un modo che permetta di velocizzare la ricerca, senza dover più partire dal primo elemento e scorrerlo tutto
- Questo array ha 12 elementi. L'ultimo elemento ha indice 11:
 - l'elemento **centrale** ha indice $11 / 2 = 5$
- In generale, data una *slice* di un array A di indice minimo l e indice massimo h , possiamo dire che:
 - la slice contiene $h - l + 1$ elementi
 - l'indice centrale della slice è $(l + h) / 2$

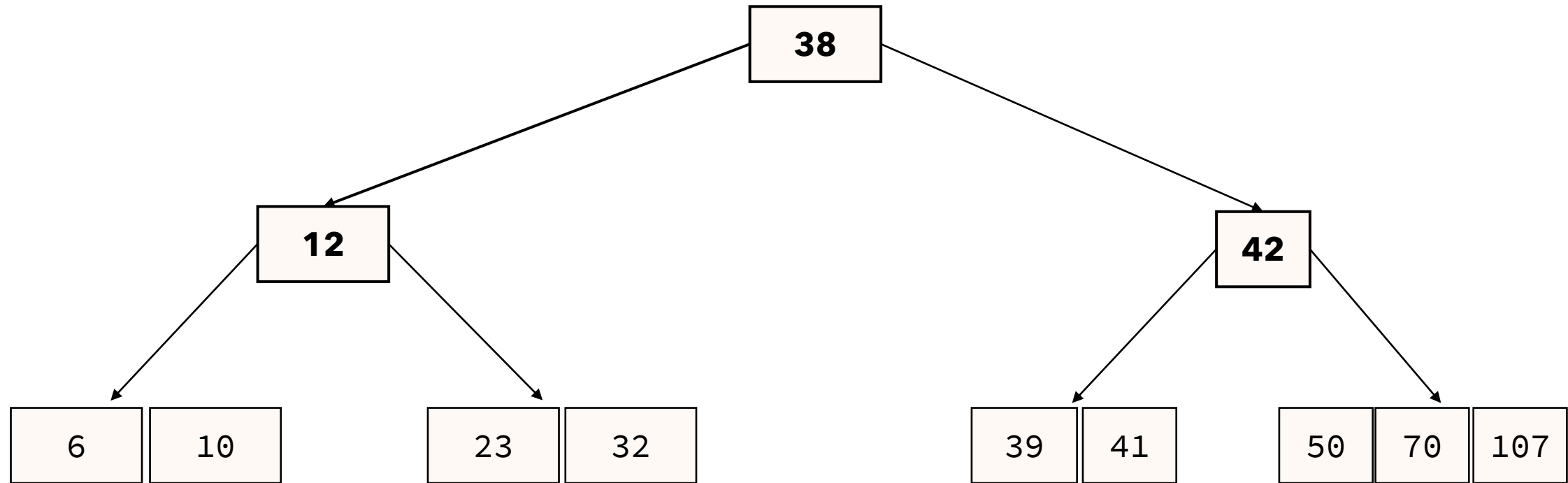
E se l'array fosse ordinato?

6	10	12	23	32	38	39	41	42	50	70	107
---	----	----	----	----	----	----	----	----	----	----	-----

vediamo l'array come un **albero binario**, dove la radice è l'elemento centrale, il figlio sinistro è il sotto-array a sinistra dell'elemento centrale, e il figlio destro è il sotto-array a destra dell'elemento centrale. Poi applichiamo la suddivisione ricorsivamente

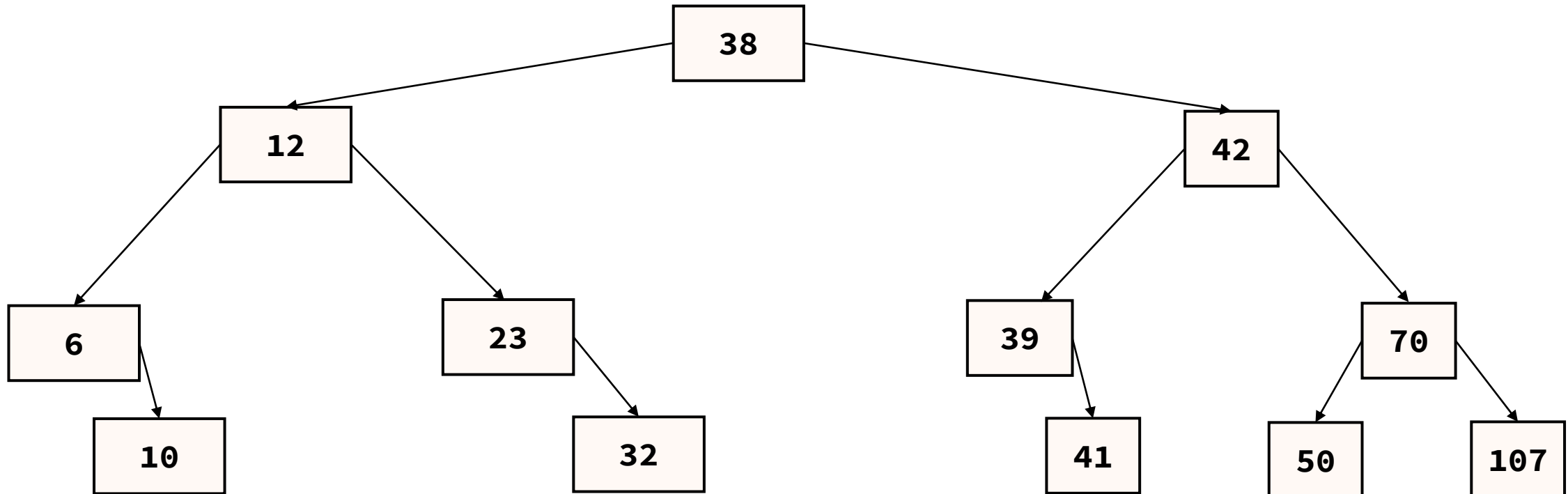


E se l'array fosse ordinato?



Ricerca binaria (*binary search*)

- **NB:** l'albero che abbiamo costruito è solo un modello di calcolo. In memoria c'è solo l'array
- Quale procedura potremmo seguire per trovare un elemento all'interno dell'albero?



Ricerca binaria

- **La ricerca binaria (o logaritmica, o dicotomica) è un algoritmo che permette di non perdere tempo nel cercare le cose dove sicuramente non esistono**
- L'albero che abbiamo visto è solo una nostra costruzione mentale per capire il funzionamento dell'algoritmo che scriveremo
- Quello che dobbiamo fare è modificare gli indici dell'array in modo da riprodurre la ricerca che abbiamo fatto sull'albero

Ricerca binaria

iteration 1

searched key: 37

	low				mid					high
indexes	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
values	5	9	12	13	20	20	30	40	42	44

Ricerca binaria

iteration 2

searched key: 37

						low		mid		high
indexes	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
values	5	9	12	13	20	20	30	40	42	44

Ricerca binaria

iteration 3

searched key: 37

						low mid	high			
indexes	0	1	2	3	4	5	6	7	8	9
values	5	9	12	13	20	20	30	40	42	44

Ricerca binaria

iteration 4

searched key: 37

							low mid high			
indexes	0	1	2	3	4	5	6	7	8	9
values	5	9	12	13	20	20	30	40	42	44

Ricerca binaria

iteration 5

searched key: 37

							mid high	low		
indexes	0	1	2	3	4	5	6	7	8	9
values	5	9	12	13	20	20	30	40	42	44

low > high
la chiave cercata non esiste nell'array,
l'algoritmo termina

Ricerca binaria

iteration 1

searched key: 40

	low				mid					high
indexes	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
values	5	9	12	13	20	20	30	40	42	44

Ricerca binaria

iteration 2

searched key: 40

						low		mid		high
indexes	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
values	5	9	12	13	20	20	30	40	42	44

`array[mid] == key`
la chiave è stata trovata,
l'algoritmo termina

Ricerca binaria

- Se l'elemento cercato è maggiore di `array[mid]`, si aggiorna solo l'indice `low`:
 - **`low = mid + 1`**
- Se l'elemento cercato è minore di `array[mid]`, si aggiorna solo l'indice `high`:
 - **`high = mid - 1`**
- Se `low > high`, significa che la ricerca è fallita. L'algoritmo termina

Ricerca binaria

```
int low = 0;
int high = size - 1;
int mid;
bool found = false;

while (!found && low <= high) {
    mid = (low + high) / 2;
    if (v[mid] == key) {
        found = true;
    }
    else if (key > v[mid]) {
        low = mid + 1;
    }
    else {
        high = mid - 1;
    }
}
```