

Laboratorio 9

Ricorsione co-variante: l'indice i scende

metodo(int[] a) wrapper che chiama: metodoRic(a, a.length - 1)

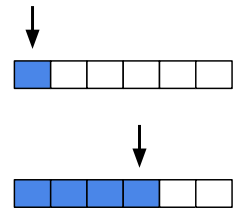
CASO BASE: $i == 0$

La proprietà vale nell'elemento 0.

PASSO RICORSIVO: $i > 0$

Se la proprietà vale nell'intervallo $[0, i-1]$, allora vale anche su $[0, i]$.

return (proprietà su i) && metodoRic(a, i - 1)



Ricorsione contro-variante: l'indice i cresce

metodo(int[] a) wrapper che chiama: metodoRic(a, 0)

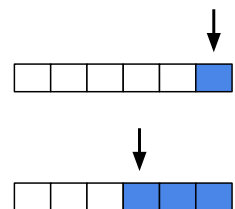
CASO BASE: $i == a.length - 1$

La proprietà vale sull'ultimo elemento.

PASSO RICORSIVO: $i < a.length - 1$

Se la proprietà vale nell'intervallo $[i+1, a.length)$, allora vale anche su $[i, a.length)$.

return (proprietà su i) && metodoRic(a, i + 1)



Ricorsione dicotomica: gli indici [i,j] indicano un intervallo

metodo(int[] a)

wrapper che chiama: metodoRic(a, 0, a.length-1)

si esclude il caso $a.length == 0$, che si gestisce a parte.

CASO BASE: $i == j$

L'intervallo contiene un solo elemento.

La proprietà vale sull'elemento i.

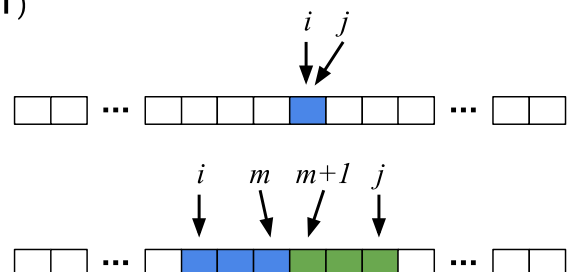
PASSO RICORSIVO: $i < j$

Definisco m come punto intermedio tra i e j.

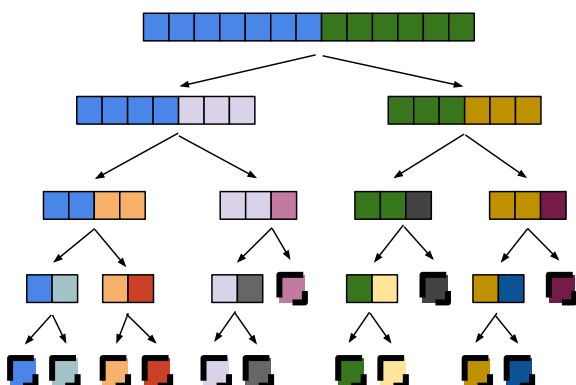
La proprietà vale nell'intervallo $[i, j]$ se vale tra $[i, m]$ e tra $[m+1, j]$

int m = (i + j) / 2; // punto mediano tra i e j

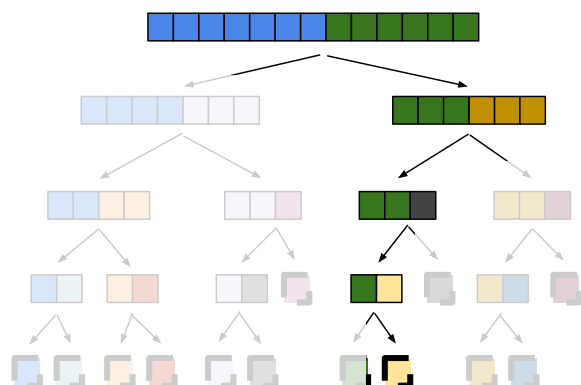
return metodoRic(a, i, m) && metodoRic(a, m+1, j);



Esempio ricorsione dicotomica:



Esempio ricerca dicotomica:



Esercizio 1

In una classe **LeggiArray** scrivere un metodo **ricorsivo** *leggiArrayInt* che chiede all'utente di inserire una sequenza di numeri letti con *Sin.readInt*, terminati dallo 0.

Realizzare la ricorsione tramite un metodo wrapper *leggiArrayInt*, ed un metodo ricorsivo:

```
int[] leggiArrayIntRic(int i)
```

dove *i* è il numero di elementi letti fino a quel momento.

Scrivere un metodo *main* per verificare il funzionamento, e stampare l'array inserito.

NOTA: di quale tipo di ricorsione si tratta?

NOTA: qual è il momento giusto per allocare l'array?

Esercizio 2

Aggiungere un metodo **ricorsivo contro-variante** *void stampaArrayIntRic(int[] intArr)* che stampa gli elementi dell'array.

Esercizio 3

Creare una nuova classe **MetodiRicorsiviSuArray** con i seguenti metodi:

- Un metodo **ricorsivo co-variante** *tuttiPari(int[] a)* che ritorna true se tutti gli elementi di un array sono pari.
- Un metodo **ricorsivo contro-variante** *esisteMultiplo(int[] a, int m)* che ritorna true se *a* contiene un elemento multiplo di *m*

Aggiungere anche un metodo *main* che chiede all'utente di inserire un array di interi, e stampa se contiene tutti numero pari e se contiene multipli di 5.

Esercizio 4

Leggere la classe **RicorsioneDicotomica.java** e capirne il funzionamento.

Esercizio 5

Aggiungere alla classe **MetodiRicorsiviSuArray** un metodo **ricorsivo dicotomico** *int sommaDispari(int[] a)* che ritorna la somma di tutti i numeri dispari di un array *a*. Modificare il metodo *main* per richiamare *sommaDispari* sui seguenti array:

```
final int[] a0 = {0,1,2,3,4,5,6,7};  
final int[] a1 = {3,7,9,4,5,12,11};  
final int[] a2 = null;  
final int[] a3 = {0,10,40,60,20};
```

Esercizio 6

Aggiungere alla classe **MetodiRicorsiviSuArray** un metodo **ricorsivo dicotomico** *int indiceMassimo(int[] a)* che ritorna l'indice dell'elemento più grande in *a*.

Definire nel *main* due arrays:

```
final int[] altezze = {5895, 4810, 6194, 4897, 4884, 8848, 6962};  
final String[] nomi = {  
    "Kilimangiaro", "Monte Bianco", "Monte Denali",  
    "Massiccio Vinson", "Puncak Jaya", "Everest", "Aconcagua"};
```

e trovare l'indice in *altezze[]* corrispondente al valore massimo con il metodo *indiceMassimo(altezze)*, e stampare il nome ed il valore corrispondente.

Esercizio 7

Aggiungere alla classe **MetodiRicorsiviSuArray** un metodo **ricorsivo co-variante** *int[]*

filtraMaggioriDi(int[] a, int limiteInferiore) che ritorna un nuovo array contenente solo gli elementi di a che sono maggiori o uguali a limiteInferiore.

Realizzare il metodo in modo che, ad ogni passo della ricorsione, si mantenga un conteggio degli elementi che hanno passato il test. In questo modo, nel caso base, si ha a disposizione il numero di elementi filtrati.

Nel main visualizzare il risultato delle seguenti chiamate:

```
filtraMaggioriDi(a0, 3);
filtraMaggioriDi(a1, 3);
filtraMaggioriDi(a2, 3);
filtraMaggioriDi(a3, 3);
```

Esercizio 8 (opzionale)

Aggiungere alla classe **MetodiRicorsiviSuArray** un metodo **ricorsivo dicotomico** int[]

filtraPari(int[] a) che ritorna un nuovo array contenente solo gli elementi di a che sono pari.

Realizzare il metodo in modo che, ad ogni passo della ricorsione, si effettua il concatenamento tra i due array parziali risultanti dalla divisione dicotomica dell'intervallo.

Per semplicità, scrivere il codice per il concatenamento in forma iterativa.

Esercizio 9

Scrivere un metodo di nome *eDue* con le seguenti caratteristiche

- *eDue* ha due parametri formali, entrambi riferimenti ad array di interi, di nome **a** e **b**;
- I due array in input a[] e b[] possono avere lunghezze differenti, e possono anche essere null;
- *eDue* restituisce un intero pari alla somma delle differenze tra a[i] e b[i]; se uno dei due array è più grande dell'altro, la somma delle differenze deve fermarsi alla più piccola delle due lunghezze;
- *eDue* deve richiamare un secondo metodo ricorsivo di supporto *eDueRic* che esegue la somma delle differenze. Tale metodo deve avere un argomento **contro-variante**, l'indice i nell'array.

Scrivere inoltre un metodo main che stampa il risultato di *eDue* nei seguenti casi:

```
eDue(a0, a1)
eDue(a1, a0)
eDue(a2, a3)
eDue(a0, a3)
```

Esercizio 10

Leggere la classe **ArrayDiArrays.java** e capirne il funzionamento.

Esercizio 11

Creare una classe **MatriciBase** che definisce, nel main:

- Un array di array mat1:
`final int[][] mat1 = { {1,0,0}, {0,1,0}, {0,0,1} };`
- Un array di array di interi mat2, costruito con le seguenti regole:
 - mat2 ha 10 righe;
 - Ogni riga di mat2 ha 10 elementi
 - L'elemento mat2[i][j] è pari a i*j
- Un array di array di interi mat3, costruito con le seguenti regole:
 - L'array mat3 ha N=8 righe;
 - La riga mat3[0] è uguale a {8,3,2,4,1,6,9,1}

- Ciascuna riga i -esima (con $0 < i < N$) ha lunghezza pari alla riga $(i-1)$ -esima meno uno;
- Ciascun elemento delle righe $i > 0$ è pari a: $\text{mat3}[i,j] = \text{mat3}[i-1, j] + \text{mat3}[i-1, j+1]$

Nel metodo *main* visualizzare a video le matrici create, usando il metodo *stampaMatrice* fornito.