Esercitazioni del
Corso di Fondamenti di Informatica

ARRAY

Prof. Giuseppe Manco

A cura di
Ing. Andrea Tagarelli

ESERCIZIO 1

Si definisca un metodo statico

```
public static int[] elaboraVettore (int[] v)
```

che verifica che gli elementi di valore dispari siano disposti in ordine crescente e che gli elementi di valore pari siano disposti in ordine decrescente:

- se la verifica ha esito negativo, il metodo restituisce il riferimento `null`;
- altrimenti restituisce un array di dimensione due contenente in prima posizione la somma degli elementi di valore dispari, e nella seconda posizione la somma degli elementi di valore pari.

Ad esempio:

- l'invocazione del metodo sul vettore [1, 10, 4, 7, 2] restituisce il vettore [8, 16], poiché le due sottosequenze rispettano l'ordine richiesto, e le somme sono, rispettivamente, 8 (=1+7) e 16 (=10+4+2);
- l'invocazione del metodo sul vettore [1, 10, 20, 7, 2] restituisce `null`, poiché la sottosequenza degli elementi pari ([10, 20, 2]) non è ordinata in modo decrescente.

ESERCIZIO 2

Si definisca un metodo statico

```
public static double fAngolo (double[] a, double[] b)
```

che, ricevuti due vettori a e b di uguale dimensione, restituisca la seguente misura:

$$\frac{a \otimes b}{\| a \| \cdot \| b \|}$$

dove $a \otimes b$ denota l'operazione di prodotto scalare fra vettori, cioè (indicata con n la dimensione dei vettori):

$$a \otimes b = \sum_{i=1}^{n} a_i \cdot b_i$$

mentre, dato un generico vettore x, $\| x \|$ denota la norma di x, calcolata come segue:

$$\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

ESERCIZIO 3

Si definisca un metodo statico

```
public static boolean verificaPalindroma (char[] v)
```

che, ricevuto un vettore di caratteri `v`, restituisca true se il vettore è palindromo.
Si ricorda che un vettore è *palindromo* se è uguale al suo inverso, cioè può essere letto identicamente da sinistra a destra e viceversa.
Ad esempio: ['a', 'n', 'n', 'a'], oppure ['r', 'a', 'd', 'a', 'r'], etc.

ESERCIZIO 4

Si definisca un metodo statico

```
public static boolean quasiSimmetrico (int[] v)
```

che restituisce `true` se e solo se il vettore ha dimensione dispari e la somma fra gli elementi in posizione simmetrica è uguale, in valore assoluto, al valore dell'elemento centrale.
Si consideri, ad esempio, il vettore `v` contenente la sequenza di elementi riportata nella seguente figura:

25	26	12	40	-52	14	15

L'applicazione del metodo `quasiSimmetrico` al vettore `v` ritorna il valore `true`, poiché:

| 25 + 15 | (valore assoluto della somma fra il primo e l'ultimo elemento)
= | 26 + 14 | (valore assoluto della somma fra il secondo ed il penultimo elemento)
= | 12 – 52 | (valore assoluto della somma fra il terzo ed il terzultimo elemento)
= | 40 | (valore assoluto dell'elemento centrale)
= 40

ESERCIZIO 5

Si definisca una classe `UtilitaVettore` che contenga i seguenti metodi statici:

- **`public static boolean`** `presente (int[] v, int[] w)`
  che restituisce `true` se almeno un elemento del vettore `w` è presente nel vettore `v;`
- **`public static int`**`[] differenza (int[] v, int[] w)`
  che restituisce un vettore ottenuto sottraendo da ogni elemento del vettore `v` il minimo del vettore `w`.

ESERCIZIO 6

Si definisca un metodo statico

```
public static double[] elaboraVettore (int[] v)
```

che restituisce un vettore di `double` contenente una copia del vettore `v` ricevuto in input, nella quale gli elementi di `v` con valore 0 sono sostituiti con la media degli elementi di `v` con valore diverso da 0.

Ad esempio, l'invocazione del metodo sul vettore

```
[-3, 10, 0, -2, 4, 0, 0]
```

restituisce il vettore

```
[-3.0, 10.0, 2.25, -2.0, 4.0, 2.25, 2.25]
```

essendo la media degli elementi di `a` con valore diverso da 0 uguale a: `(-3+10-2+4) / 4 = 2.25`.

ESERCIZIO 7

Si definisca un classe `UtilitaVettore` che contenga i seguenti metodi statici:

- **`public static boolean`** `presente (int[] v, int x)`

restituisce vero se il valore `x` è presente nel vettore `v`;

- **`public static boolean`** `ordinato (int[] v)`

restituisce vero se il vettore `v` è ordinato in modo crescente;

- **public static double** media (**int**[] v)
  restituisce la media degli elementi di v;
- **public static double** dev (**int**[] v)
  restituisce la deviazione standard degli elementi di v.

Si ricorda che la deviazione standard è definita come: $dev(v) = \sqrt{\frac{\sum_{i=0}^{v.length-1}(v[i] - media(v))^2}{v.length}}$

ESERCIZIO 8

Si definisca un metodo statico

```
public static int[] elaboraVettore (int[] v)
```

che verifica che gli elementi di posizione dispari siano disposti in ordine crescente e che gli elementi di posizione pari siano disposti in ordine decrescente:

- se la verifica ha esito negativo, il metodo restituisce il riferimento `null`;
- altrimenti restituisce un array di dimensione uno contenente la differenza tra la somma degli elementi di valore dispari e la somma degli elementi di valore pari.

Ad esempio:

- l'invocazione del metodo sul vettore [10, 10, 7, 14, 3] restituisce il vettore [4], poiché le due sottosequenze rispettano l'ordine richiesto, e le somme sono, rispettivamente, 20 (=10+7+3) e 24 (=10+14);
- l'invocazione del metodo sul vettore [1, 10, 20, 7, 2] restituisce `null`, poiché la sottosequenza degli elementi di posizione dispari ([1, 20, 2]) non è ordinata in modo decrescente.

ESERCIZIO 9

Si definisca un metodo statico

```
public static int[] maxVettore (int k, int[] v)
```

che, ricevuto un intero `k` ed un vettore di interi `v` (la cui dimensione è indicata con `n`), restituisca:

(a) `null`, se la dimensione `n` del vettore `v` non è un multiplo di `k`;
(b) un vettore di interi di dimensione `k`, con i massimi dei sottovettori di dimensione `n / k` ottenuti partizionando `v` in `k` parti.

Ad esempio, ricevuti come parametri l'intero 3 ed il seguente vettore di dimensione 12

```
[3, 10, 0, 8, 2, 4, 0, 1, 0, 5, 8, 2]
```

il metodo deve restituire il seguente vettore di dimensione 3: `{10,4,8}`.
Infatti: `max{3,10,0,8}=10`, `max{2,4,0,1}=4` e `max{0,5,8,2}=8`.

ESERCIZIO 10

Si definisca un metodo statico

```
public static void invertiVettore (int[] v)
```

che, ricevuto un vettore di interi `v`, lo modifica invertendo l'ordine degli elementi.
Ad esempio, se `v` = [3, 9, 0, 8, 2], il metodo deve modificarlo nel vettore [2, 8, 0, 9, 3].

ESERCIZIO 11

Si definisca un metodo statico

```
public static int[] stampaTriangoloDiTartaglia (int n)
```

che, ricevuto un intero `n`, scriva in output le prime `n` righe del triangolo di Tartaglia e restituisca un array di interi contenente l'ultima riga (riga `n`) del triangolo.
Ad esempio, ricevuto come parametro il valore `n` =5 si ottiene il seguente triangolo di Tartaglia:

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
```

ESERCIZIO 12

Si definisca un metodo statico

```
public static boolean verificaVettore (int[] v)
```

che, ricevuto un vettore di interi `v`, restituisca `true` se ogni elemento di `v` è maggiore o uguale della somma dei due valori precedenti (se esistono), `false` altrimenti.
Ad esempio, se `v` = [1, 2, 4, 7, 13, 20, 35], il metodo restituirà `true`.

ESERCIZIO 13

Si definisca un metodo statico

```
public static int[] restituisciDistinti (int[] v)
```

che, ricevuto un vettore di interi `v`, restituisca un nuovo vettore contenente tutti e soli gli elementi distinti di `v`.
Ad esempio, se `v` = [2, 1, 5, 2, 6, 2, 5, 9, 3, 1] allora il vettore da restituire è [1, 2, 3, 5, 6, 9].

ESERCIZIO 14

Si definisca un metodo statico

```
public static boolean verificaSomma (int[] v1, int[] v2)
```

che, ricevuto due vettori di interi `v1` e `v2`, restituisca `true` se la somma degli elementi di `v1` che si trovano nelle posizioni pari è uguale alla somma degli elementi di `v2` che si trovano nelle posizioni dispari, `false` altrimenti. La posizione 0 è da considerarsi pari.
Ad esempio, se `v1` = [2, 3, 6, 7, 10] e `v2` = [10, 11, 2, 7, 6], allora il risultato è `true`.

ESERCIZIO 15

Si definisca un metodo statico

```
public static boolean verificaProdotto (int[] v1, int[] v2)
```

che, ricevuto due vettori di interi `v1` e `v2`, restituisca `true` se il prodotto degli elementi di `v1` che si trovano nelle posizioni pari è uguale al prodotto degli elementi di `v2` che si trovano nelle posizioni dispari, `false` altrimenti. La posizione 0 è da considerarsi pari.
Ad esempio, se `v1` = [1, 3, 6, 7, 10] e `v2` = [12, 20, 2, 3, 17], allora il risultato è `true`.

Soluzioni degli esercizi proposti

ESERCIZIO 1

La seguente soluzione verifica le condizioni imposte nella traccia e calcola il risultato tramite una sola scansione del vettore:

```
public static int[] elaboraVettore ( int[] v ) {
	int[] vSomme = new int[2];
	boolean ok = true;
	int posPrecPari = 0, posPrecDispari = 0;
	// memorizzano, rispettivamente, la posizione del precedente valore pari e del precedente valore dispari incontrato nella scansione del vettore
	for (int i=0; i<v.length && ok; i++)
		if (v[i]%2 == 0) { //l'elemento v[i] ha valore pari
			if (posPrecPari != 0  &&  v[i] > v[posPrecPari])
				// v[i] è maggiore dell'elemento pari che lo precedente nella sequenza: viola l'ordine decrecente
				ok = false;
			posPrecPari = i;
			vSomme[0] += v[i];
		}
		else { //l'elemento v[i] ha valore pari
			if (posPrecDispari !=0  &&  v[i] < v[posPrecDispari])
			// v[i] è minore dell'elemento dispari che lo precedente nella sequenza: viola l'ordine crecente
			ok = false;
			posPrecDispari = i;
			vSomme[1] += v[i];
		}
	if (ok)
		return vSomme;
	else
		return null;
}
```

ESERCIZIO 2

Osservando, dato un vettore $\boldsymbol{x}$, il prodotto scalare $\boldsymbol{x} \otimes \boldsymbol{x}$ rappresenta il quadrato della norma di $\boldsymbol{x}$, una possibile soluzione dell'esercizio è la seguente:

```
static double prodottoScalare (double[] a, double[] b) {
	double ret=0;
	for(int i=0; i<a.length; i++)
		ret+=a[i]*b[i];
	return ret;
}

public static double fAngolo (double[] a, double[] b) {
	double normaA = Math.sqrt(prodottoScalare(a,a));
	double normaB = Math.sqrt(prodottoScalare(b,b));
	return prodottoScalare(a,b) / (normaA*normaB);
}
```

ESERCIZIO 3

Il ciclo che verifica se il vettore è palindromo è iterato fino a che, procedendo con due indici di scansione inizializzati rispettivamente ad 1 e al valore pari alla dimensione del vettore -1, si incontrano coppie identiche di caratteri ed esiste

ancora una porzione utile di vettore da scandire (*i<j*). Quando si esce dal ciclo con *i>=j* allora la parola è certamente palindroma. I due casi >= sono associati rispettivamente ad un valore di dimensione del vettore pari o dispari:

```
public static boolean verificaPalindroma (char[] v) {
	int i=0;
	int j = v.length-1;
	while (i<j && v[i] == v[j]) {
		i++;
		j--;
	}
	if (i>=j)
		return true;
	else
		return false;
}
```

ESERCIZIO 4

```
public static boolean quasiSimmetrico (int[] v) {
	if (v.length == 0)
		return false;
	for (int i = 0; i < v.length/2; i++)
		if ( Math.abs(v[i] + v[v.length-1 - i]) !=
		     Math.abs(v[v.length / 2]) )
		   return false;
	return true;
}
```

ESERCIZIO 5

```
class UtilitaVettore {
	public static boolean presente (int[] v, int[] w) {
		boolean trovato = false;
		for (int i=0; i<w.length && !trovato; i++)
			for (int j=0; j<v.length && !trovato; j++)
				if (w[i] == v[j])
					trovato = true;
		return trovato;
	}

	static int minimo (int[] v) {
		int min;
			// il primo elemento di v viene scelto come minimo provvisorio
		min = v[0];
			// scandisce v alla ricerca del minimo
		for (int i=1; i<v.length; i++)
			if (v[i] < min)
				min = v[i];
		return min;
	}

	public static int[] differenza (int[] v, int[] w) {
		int min = minimo(w);
		int[] d = new int[v.length];
		for (int i=0; i<v.length; i++)
			d[i] = v[i] - min;
		return d;
	}
}
```

ESERCIZIO 6

La seguente soluzione verifica le condizioni imposte nella traccia e calcola il risultato tramite una sola scansione del vettore:

```
public static double[] elaboraVettore (int[] v ) {
	double [] copiaV = new double [v.length];
	double media = media(v);
	for (int i=0; i<v.length; i++)
		if (v[i] != 0)
			copiaV[i] = v[i];
		else
			copiaV[i] = media;
	return copiaV;
}

static double media (int[] v) {
	double m = 0;
	int nonZero = 0;		// n° di elementi diversi da zero
		// calcola la somma degli elementi di v con valore diverso da 0
	for (int i=0; i<v.length; i++)
		if (v[i] != 0) {
			m += v[i];
			nonZero++;
		}
	return (m / (double)nonZero);
}
```

ESERCIZIO 7

```
class UtilitaVettore {
	public static boolean presente (int[] v, int x) {
		boolean trovato = false;
		for (int i=0; i<v.length && !trovato; i++)
			if (v[i] == x)
				trovato = true;
		return trovato;
	}

	public static boolean ordinato (int[] v) {
		boolean crescente = true;
		for (int i=1; crescente && i<v.length; i++)
			if (v[i] <= v[i-1])
				crescente = false;
		return crescente;
	}

	public static double media (int[] v) {
		double m = 0;
		for (int i=0; i<v.length; i++)
			m += v[i];
		return (m / (double)v.length);
	}

	public static double dev (int[] v) {
		double dev = 0;
		double media = media(v);
		for (int i=0; i<v.length; i++)
			dev += Math.pow((v[i] - media), 2);
		return Math.sqrt( dev / (double)v.length );
	}
}
```

ESERCIZIO 8

La seguente soluzione verifica le condizioni imposte nella traccia e calcola il risultato tramite una sola scansione del vettore:

```
public static int[] elaboraVettore (int[] v) {
	int[] vDifferenze = new int[1];
	boolean ok = true;
	int sommaPari = 0, sommaDispari = 0;
	for (int i=0; i<v.length && ok; i++)
		if (i%2 == 0) { // l'elemento si trova in posizione pari
			if (i+2 < v.length && v[i+2] >= v[i])
				ok = false;
			sommaPari += v[i];
		}
		else { // l'elemento si trova in posizione dispari
			if (i+2 < v.length  && v[i+2] <= v[i])
				ok = false;
			sommaDispari += v[i];
		}
	if (ok) {
		vDifferenze[0] = sommaDispari - sommaPari;
		return vDifferenze;
	}
	else
		return null;
}
```

ESERCIZIO 9

```
public static int[] maxVettore (int k, int[] v) {
	if (v.length % k != 0)		// la dimensione del vettore non è multipla di k
		return null;
	int[] r = new int[k];
	int dimS = v.length / k;	// dimensione dei sottovettori
	for (int i=0; i<k; i++) {
		int j = dimS * i;	// passo di scanzione del vettore v
		int max = v[j];
			// scandisce il sottovettore v alla ricerca del massimo
		for (int p = j+1; p < dimS+j; p++)
			if (v[p] > max)
				max = v[p];
		r[i] = max;
	}
	return r;
}
```

ESERCIZIO 10

La seguente soluzione risolve il problema scandendo il vettore fino a metà della sua lunghezza:

```
public static void invertiVettore (int[] v) {
	int d = v.length -1;
	int j = d;
	for (int i=0; i<j; i++) {
		int tmp = v[d-i];
		v[d-i] = v[i];
		v[i] = tmp;
		j--;
	}
}
```

ESERCIZIO 11

L'*n*-esima riga del triangolo di Tartaglia contiene gli *n*+1 coefficienti della potenza *n*-esima di un binomio, calcolabili direttamente come coefficienti binomiali $C_{n,k}$. Tuttavia i coefficienti della riga *n*-esima sono anche determinabili a partire dai coefficienti della riga *n*-1. L'elemento in prima ed in ultima posizione assume sempre valore 1, mentre il generico elemento in posizione $1< i <n$ è calcolabile come somma tra l'*i*-esimo elemento e l'*i*-1 esimo della riga precedente. La soluzione fornita si basa sull'uso di due vettori `a` e `b`. Su `b` viene memorizzata la riga al passo precedente, mentre `a` è destinato a contenere la nuova riga da calcolare al passo corrente:

```
public static int[] stampaTriangoloDiTartaglia (int n) {
	int[] a = new int[n];
	int[] b = new int[n];
	for (int i=0; i<n; i++) {
		a[0] = 1;
		a[i] = 1;
		for (int j=1; j<i; j++)
			a[j] = b[j] + b[j-1];
		for (int j=0; j<=i; j++)
			System.out.print(a[j] +"\t");
		System.out.println();

		for (int j=0; j<i+1; j++)
			b[j] = a[j];
	}
	return b;
}
```

ESERCIZIO 12

```
public static boolean verificaVettore (int[] v) {
	boolean verifica = true;
	for (int i=2; verifica && i<v.length; i++)
		if (v[i] < v[i-1] + v[i-2])
			verifica = false;
	return verifica;
}
```

ESERCIZIO 13

Una possibile soluzione che sfrutta l'ordinamento del vettore `v` prima dell'estrazione degli elementi distinti:

```
static void ordina (int v[]) {
	boolean scambio;
	int j = v.length-1;
	do {
		scambio = false;
		for (int i=0; i<j; i++) {
			int temp;
			if (v[i]>v[i+1]) {
				temp = v[i];
				v[i] = v[i+1];
				v[i+1] = temp;
				scambio = true;
			}
		}
		j--;
	 } while (scambio);
}
```

```
public static int[] distinti (int[] v) {
	int i;
	ordina (v);
	int nuovo[] = new int[v.length];
	int k=0;
	for (i=0; i<v.length-1; i++) {
		while (v[i]==v[i+1])
			i++;
		 nuovo[k] = v[i];
		 k++;
	}
	if (i<v.length)
		nuovo[k]=v[i];
	k++;
		// copia del vettore dei distinti in un nuovo vettore di dimensione appropriata
	int copia[] = new int[k];
	for (i=0; i<k; i++)
		copia[i] = nuovo[i];
	return copia;
}
```

ESERCIZIO 14

```
public static boolean verificaSomma (int[] v1, int[] v2) {
	int sommaPari = 0;
	for (int i=0; i < v1.length; i+=2)
		sommaPari += v1[i];
	int sommaDispari = 0;
	for (int i=1; i < v2.length; i+=2)
		sommaDispari += v2[i];
	return (sommaPari == sommaDispari);
}
```

ESERCIZIO 15

```
public static boolean verificaProdotto (int[] v1, int[] v2) {
	int prodPari = 1;
	for (int i=0; i < v1.length; i+=2)
		prodPari *= v1[i];
	int prodDispari = 1;
	for (int i=1; i < v2.length; i+=2)
		prodDispari *= v2[i];
	return (prodPari == prodDispari);
}
```