FAKULTAS
ILMU
KOMPUTER

Two Dimensional Array

Dasar – Dasar Pemrograman 2

Dinial Utami Nurul Qomariah

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Credits

- Liang, Introduction to Java Programming, 11th Edition, Ch. 2
- Downey & Mayfield, Think Java: How to Think Like a Computer Scientist, Ch. 2
- Slide Kuliah Dasar-dasar Pemrograman 2 Semester Genap 2021/2022

Motivations

Thus far, you have used one-dimensional arrays to model linear collections of elements. You can use a two-dimensional array to represent a matrix or a table. For example, the following table that describes the distances between the cities can be represented using a two-dimensional array.

Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

Motivations

```
double[][] distances = {
    {0, 983, 787, 714, 1375, 967, 1087},
    {983, 0, 214, 1102, 1763, 1723, 1842},
    {787, 214, 0, 888, 1549, 1548, 1627},
    {714, 1102, 888, 0, 661, 781, 810},
    {1375, 1763, 1549, 661, 0, 1426, 1187},
    {967, 1723, 1548, 781, 1426, 0, 239},
    {1087, 1842, 1627, 810, 1187, 239, 0},
};
```

Array Multi Dimensi

Declare and Create Two-dimensional Arrays

```
// Declare array ref var
dataType[][] refVar;

// Create array and assign its reference to variable
refVar = new dataType[10][10];

// Combine declaration and creation in one statement
dataType[][] refVar = new dataType[10][10];

// Alternative syntax
dataType refVar[][] = new dataType[10][10];
```

Declare and Create Two-dimensional Arrays

```
int[][] matrix = new int[10][10];
```

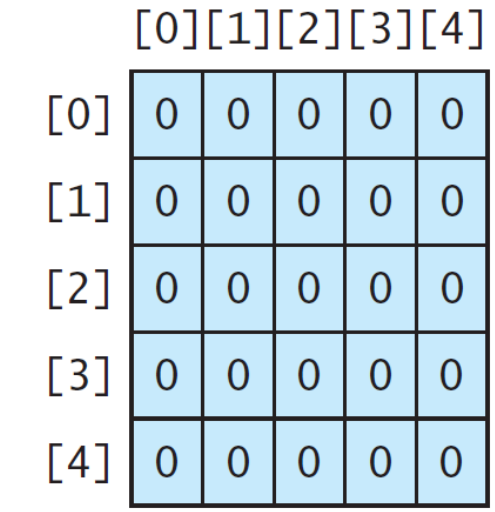
or

```
int matrix[][] = new int[10][10];
matrix[0][0] = 3;

for (int i = 0; i < matrix.length; i++)
  for (int j = 0; j < matrix[i].length; j++)
    matrix[i][j] = (int)(Math.random() * 1000);

double[][] x;
```

Two-dimensional Array Illustration

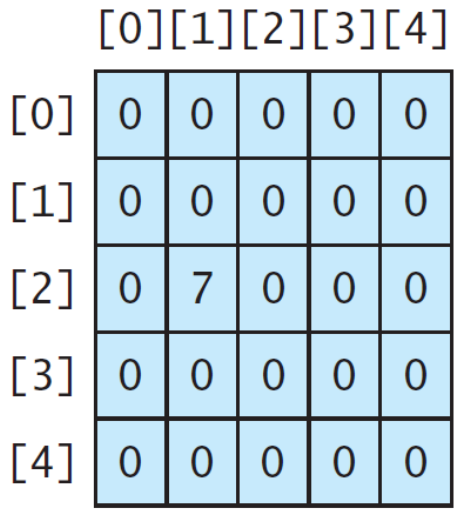



```
matrix = new int[5][5];
```

(a)

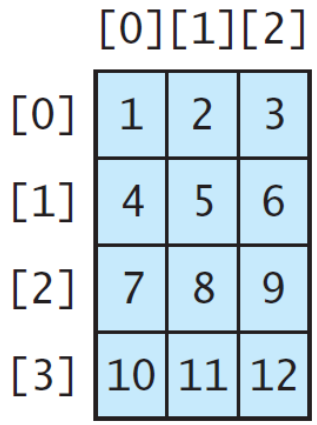
matrix.length? 5

matrix[0].length? 5




```
matrix[2][1] = 7;
```

(b)




```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

(c)

array.length? 4

array[0].length? 3

Declaring, Creating, and Initializing Using Shorthand Notations

You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

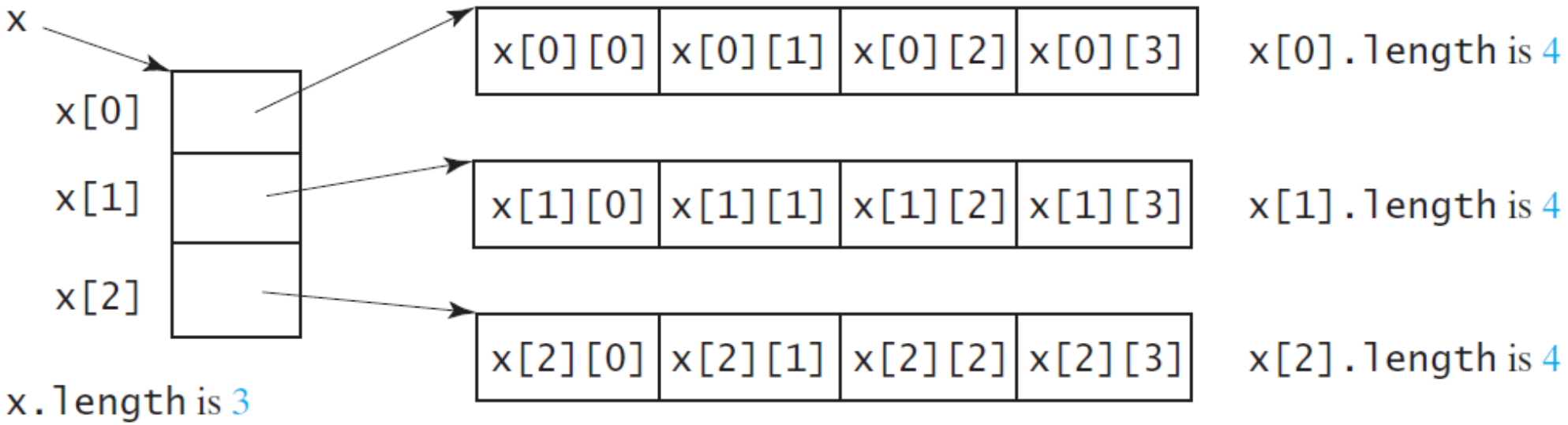
```
int[][] array =
{
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Same as

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

Lengths of Two-dimensional Arrays

```
int[][] x = new int[3][4];
```



Lengths of Two-dimensional Arrays

```
int[][] array = {
   {1, 2, 3},
   {4, 5, 6},
   {7, 8, 9},
   {10, 11, 12}
};
```

```
array.length
array[0].length
array[1].length
array[2].length
array[3].length
```

`array[4].length` ArrayIndexOutOfBoundsException

2-D arrays: Guess the output?

```
double[][] twoDArr = { {0.5,2.5,2.0,5.0},
                       {1.5,0.5,1.0,7.0},
                       {3.5,1.5,3.0,1.0}
                     };

System.out.println(Arrays.toString(twoDArr));
```

2-D arrays: Guess the output?

```
double[][] twoDArr = { {0.5,2.5,2.0,5.0},
                       {1.5,0.5,1.0,7.0},
                       {3.5,1.5,3.0,1.0}
                     };

System.out.println(Arrays.toString(twoDArr[1]));
```

2-D arrays: Guess the output?

```
double[][] twoDArr = { {0.5,2.5,2.0,5.0},
                       {1.5,0.5,1.0,7.0},
                       {3.5,1.5,3.0,1.0}
                     };

System.out.println(Arrays.toString(twoDArr[1][3]));
```

2-D arrays: Guess the output?

```
double[][] twoDArr = { {0.5,2.5,2.0,5.0},
                       {1.5,0.5,1.0,7.0},
                       {3.5,1.5,3.0,1.0}
                     };

System.out.println(twoDArr[1][3]);
```

2-D arrays: Guess the output?

```
double[][] twoDArr = { {0.5,2.5,2.0,5.0},
                       {1.5,0.5,1.0,7.0},
                       {3.5,1.5,3.0,1.0}
                     };

System.out.println(twoDArr[2][1]);
```

Quiz time

- Create a method **print2D** to print the content of a 2D-array of doubles! For example, given:

```
double[][] twoDArr = { {0.5,2.5,2.0,5.0},
                       {1.5,0.5,1.0,7.0},
                       {3.5,1.5,3.0,1.0}
                     };
```

`print2D(twoDArr)` will print:

```
0.5 2.5 2.0 5.0
1.5 0.5 1.0 7.0
3.5 1.5 3.0 1.0
```

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Quiz time

- Create a method **sum2D** to sum all elements in a two dimensional array !

Quiz time

- What's the output?

```
double[][] twoDArr = { {0.5,2.5,2.0,5.0},
                       {},
                       {3.5,1.5}};

print2D(twoDArr);
```

Ragged Arrays

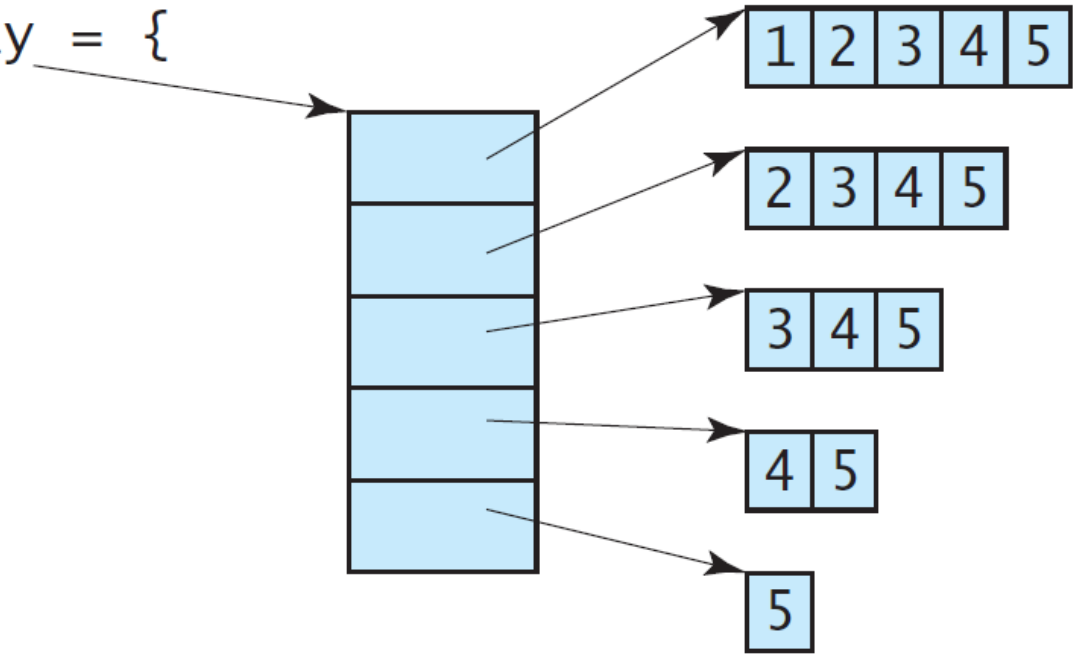
Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

```
int[][] matrix = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5}
};
```

matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1

Ragged Arrays

```
int[][] triangleArray = {
   {1, 2, 3, 4, 5},
   {2, 3, 4, 5},
   {3, 4, 5},
   {4, 5},
   {5}
};
```



Multidimensional Array (3-D)

- Yes, we can go further.
- Previously, our arrays contain arrays. Now, what if our arrays contain arrays of arrays?

```
int[][][] threeDArr1 = new int[3][3][3];
int[][][] threeDArr2 = {{{1,2,3},{3,2,1},{2,1,5}},
                        {{5,2,5},{1,1,1},{7,1,0}},
                        {{4,6,7},{4,5,4},{4,6,6}}};
```

Multidimensional Array (3-D)

- What's the output?

```
int[][][] threeDArr1 = new int[3][3][3];
int[][][] threeDArr2 = {{{1,2,3},{3,2,1},{2,1,5}},
                        {{5,2,5},{1,1,1},{7,1,0}},
                        {{4,6,7},{4,5,4},{4,6,6}}};
System.out.println(threeDArr1[0][0][1]);
System.out.println(Arrays.toString(threeDArr1[0][0]));
System.out.println(threeDArr2[0][1][2]);
System.out.println(threeDArr2[2][1][1]);
System.out.println(Arrays.toString(threeDArr2[1][2]));
System.out.println(Arrays.toString(threeDArr2[0]));
```

Multidimensional Array (3-D)

- What's the output?

```
int[][][] threeDArr1 = new int[3][3][3];
int[][][] threeDArr2 = {{{1,2,3},{3,2,1},{2,1,5}},
                        {{5,2,5},{1,1,1},{7,1,0}},
                        {{4,6,7},{4,5,4},{4,6,6}}};
System.out.println(threeDArr1[0][0][1]);
System.out.println(Arrays.toString(threeDArr1[0][0]));
System.out.println(threeDArr2[0][1][2]);
System.out.println(threeDArr2[2][1][1]);
System.out.println(Arrays.toString(threeDArr2[1][2]));
System.out.println(Arrays.toString(threeDArr2[0]));
```

Output:

```
0
[0, 0, 0]
1
5
[7, 1, 0]
[[I@3ac3fd8b, [I@5594a1b5, [I@6a5fc7f7]
```

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Searching And Sorting

a

Searching dan Sorting merupakan proses yang umum dikenakan terhadap data di dalam array.

Searching → proses mencari sebuah nilai spesifik di dalam array

Sorting → proses penyusunan elemen pada array dari acak menjadi teratur menurut aturan tertentu (ascending/descending)

Algoritma searching:

- Linear Search
- Binary Search

Algoritma sorting:

- Selection Sort
- Insertion Sort
- Bubble Sort
- Merge Sort
- Quick Sort
- Heap Sort

Linear Search

- Pada linear search, pencarian nilai X di dalam array dilakukan dengan membandingkan nilai X terhadap setiap nilai yang ada di dalam array secara sekuensial.
- Pencarian berhenti jika nilai X ditemukan atau iterasi sudah mencapai elemen terakhir array.

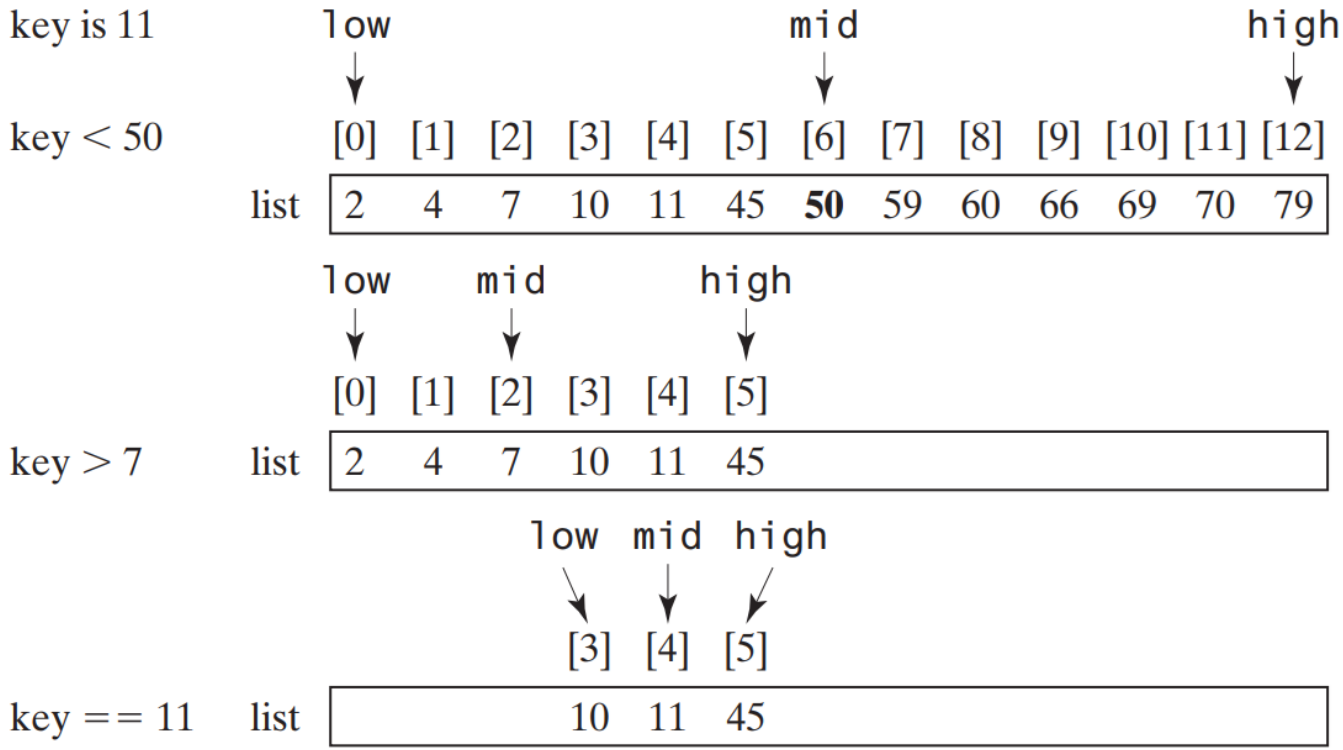
```
public static int linearSearch(int[] list, int key) {
    for (int i = 0; i < list.length; i++) {
        if (key == list[i])
            return i;
    }
    return -1;
}
```

Return indeks i yang menunjukkan posisi key di dalam array.

Return -1 jika key tidak ditemukan.

Binary Search

- Pada binary search, array yang digunakan harus dalam kondisi terurut (diasumsikan terurut menaik).
- Pencarian dilakukan dengan membandingkan nilai X (nilai yang dicari) terhadap nilai elemen tengah array.
  - Jika nilai **X < nilai elemen tengah**, maka pencarian dilanjutkan terhadap subarray di sebelah kiri elemen tengah.
  - Jika nilai **X = nilai elemen tengah**, maka pencarian dihentikan.
  - Jika nilai **X > nilai elemen tengah**, maka pencarian dilanjutkan terhadap subarray di sebelah kanan elemen tengah.



Binary Search

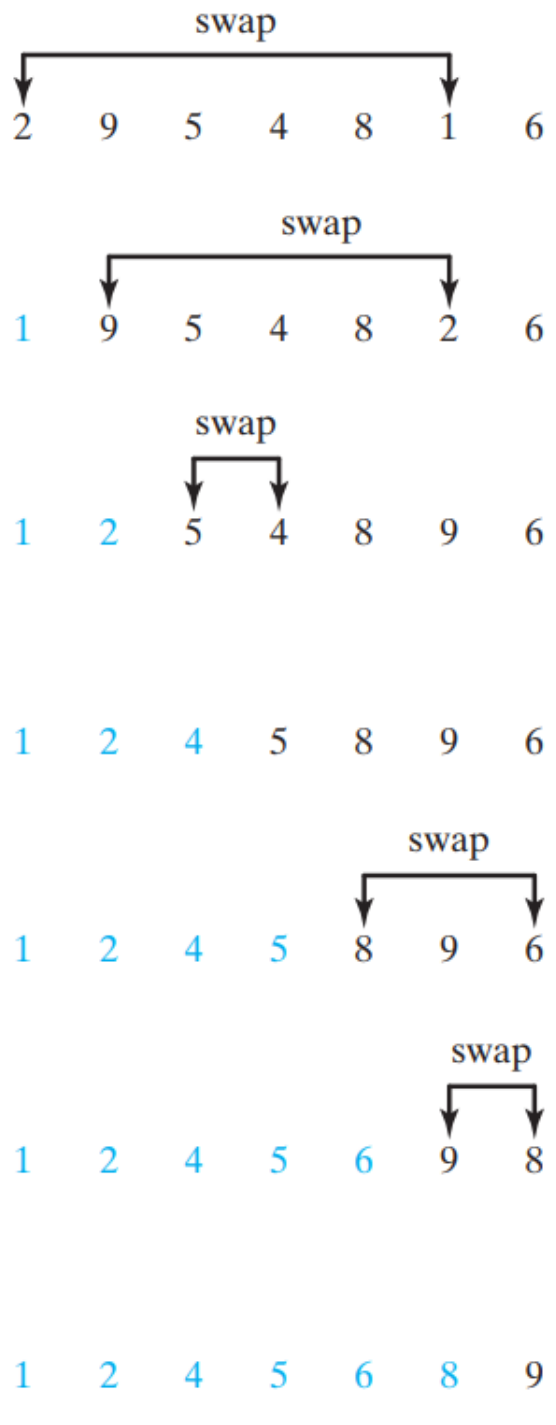
```
public static int binarySearch(int[] list, int key) {
    int low = 0;
    int high = list.length - 1;
    while (high >= low) {
        int mid = (low + high) / 2;
        if (key < list[mid]){
            high = mid - 1;}
        else if (key == list[mid]){
            return mid;}
        else{
            low = mid + 1;}
    }
    return -low - 1;
}
```

Pengecekan kondisi:
- Jika key < list[mid], maka high diupdate
- Jika key = list[mid], maka return mid
- Jika key > list[mid], maka low diupdate

Return negatif jika key tidak ditemukan.
- Bisa juga return -1
- Return -low - 1 untuk dapat mengindikasikan posisi penyisipan elemen yang tidak ditemukan

Selection Sort

- Proses pengurutan array (ascending) menggunakan selection sort:
  - Cari elemen terkecil dan tukar dengan elemen indeks 0.
  - Cari elemen terkecil kedua dan tukar dengan elemen indeks 1.
  - Cari elemen terkecil ketiga dan tukar dengan elemen indeks 2.
  - Dan seterusnya hingga semua elemen berada pada posisiseharusnya.
- Selection sort membutuhkan sebanyak N - 1 iterasi di mana N menyatakan panjang array.



Selection Sort

```
public static void selectionSort(int[] list) {
    for (int i = 0; i < list.length - 1; i++) {
        int currentMin = list[i];
        int currentMinIndex = i;
        for (int j = i + 1; j < list.length; j++) {
            if (currentMin > list[j]) {
                currentMin = list[j];
                currentMinIndex = j;
            }
        }
        if (currentMinIndex != i) {
            liqst[currentMinIndex] = list[i];
            list[i] = currentMin;
        }
    }
}
```

Pencarian elemen terkecil di setiap iterasi.

Elemen terkecil setiap iterasi dipindahkan ke posisi yang seharusnya.