Thomas van Lingen

2074690

Deel 1

Opgave 1

```
public class Sorter {
   public static void bubblesortArray(int[] toSort){
      for(int endIndex = 0; endIndex < toSort.length-1; endIndex++){
         for(int currentIndex = toSort.length-1; currentIndex > 0; currentIndex--){
            if(toSort[currentIndex] < toSort[currentIndex-1]){
               swapArrayItems(toSort, currentIndex, currentIndex-1);
            }
         }
      }
   }
   private static void swapArrayItems(int[] toSwap, int swapIndex1, int swapIndex2){
      int copy = toSwap[swapIndex1];
      toSwap[swapIndex1] = toSwap[swapIndex2];
      toSwap[swapIndex2] = copy;
   }
}
```

Hoeveelheid comparisons met 10 integers: 45

Hoeveelheid comparisons met 20 integers: 217

Dus +- 4* comparisons met een verdubbeling van array

Dit wordt ook wel O(N^2) genoemd

```
public static void main(String args[]){
      int[] banaan = {21, 6, 4, 3, 2, 10, 3, 9};
      Sorter.bubblesortArray(banaan);
      for(int i = 0; i < banaan.length; i++){
            System.out.print(banaan[i] + " ");
      }
      System.out.println("");
}
```

Geeft:

2 3 3 4 6 9 10 21

Process finished with exit code 0

Opgave 2

```
public static void bubblesortNAWArray_split(NAW[] toSort){
   int inner, outer;
   NAW copy;
   //Sort by cityOfResidence
   for(outer = toSort.length-1; outer > 0; outer--){
      for(inner=0; inner < outer; inner++){
         if(toSort[inner].getCityOfResidence().compareTo(toSort[inner+1].getCityOfResidence()) >= 0){
            copy = toSort[inner];
            toSort[inner] = toSort[inner+1];
            toSort[inner+1] = copy;
         }
      }
   }
   //Sort by address
   for(outer = toSort.length-1; outer > 0; outer--){
      for(inner=0; inner < outer; inner++){
         if(toSort[inner].getAddress() - toSort[inner+1].getAddress() >= 0){
            copy = toSort[inner];
            toSort[inner] = toSort[inner+1];
            toSort[inner+1] = copy;
         }
      }
   }
}
```

Dankzij het = teken is het algorithme nu niet meer stabiel: betekende dat de 2e sort het gesorteer van het 1e gedeelte compleet door elkaar gooit

Testwijze:

```
public static void main(String args[]){
   int size  = 50;
   NAW[] NAWList = new NAW[size];
   Random RNG = new Random();
   for(int i = 0; i < size; i++){
      NAWList[i] = new NAW();
      NAWList[i].setName("Name " + RNG.nextInt(10));
      NAWList[i].setAddress(RNG.nextInt(10));
      NAWList[i].setCityOfResidence("Stad " + RNG.nextInt(10));
   }
   for(int i = 0 ; i < size; i++){
      System.out.println("[" + i + "] " + NAWList[i].getCityOfResidence() + " : " + NAWList[i].getAddress() + " , "
+ NAWList[i].getName());
   }
   System.out.println("SORTING stuff");
   Sorter.bubblesortNAWArray_split(NAWList);
   for(int i = 0 ; i < size; i++){
      System.out.println("[" + i + "] " + NAWList[i].getCityOfResidence() + " : " + NAWList[i].getAddress() + " , "
+ NAWList[i].getName());
   }
}
```

Deel 2

Opgave 1

```
public static void selectionSortArray(int[] toSort){
	long start_time = System.nanoTime();
	for(int currentPlacementIndex = toSort.length-1; currentPlacementIndex >= 0;
currentPlacementIndex--){
		int highestIndex = currentPlacementIndex;
		for(int currentIndex = 0; currentIndex < currentPlacementIndex; currentIndex++){
			if(toSort[currentIndex] > toSort[highestIndex]){
				highestIndex = currentIndex;
			}
		}
		swapArrayItems(toSort, currentPlacementIndex, highestIndex);
	}
	long end_time = System.nanoTime();
	double difference = (end_time - start_time)/1e6;
	System.out.println("time taken: " + difference);
}
```

Ik heb de tijd gemeten van het sorteren van een random array van 50 ints vs het sorteren van er een met de grote van 100.

Time taken: 0.047526

Time taken: 0.271368

Hier zit een factor 4 verschil in wat duidt dat het algoritme van O(N^2) is

Opgave 2

```
//Make note that sortedIndex should indicate the index from which on the array is already
sorted


//Therefore 7, for an array of the size of 10, would indicate that 7-9 is sorted.
public static void insertionSortArray(int[] toSort, int sortedIndex){
    for(int currentSortingIndex = sortedIndex-1; currentSortingIndex >= 0;
currentSortingIndex--){
        int copy = toSort[currentSortingIndex];
        for(int currentCompareIndex = currentSortingIndex+1; currentCompareIndex <=
toSort.length; currentCompareIndex++){
            if(currentCompareIndex < toSort.length){
                if(copy > toSort[currentCompareIndex]) {
                    toSort[currentCompareIndex - 1] = toSort[currentCompareIndex];
                } else {
                    toSort[currentCompareIndex-1] = copy;
                    break;
                }
            } else {
                toSort[toSort.length-1] = copy;
                break;
            }
        }
    }
}

public static void insertionSortArray(int[] toSort){
    insertionSortArray(toSort, toSort.length-1);
}
```

Ik heb de tijd gemeten van het sorteren van een random array van 50 ints vs het sorteren van er een met de grote van 100.

Time taken: 0.03856

Time taken: 0.138102

Hier zit een factor 4 verschil in wat duidt dat het algoritme van O(N^2) is

Uitgevoerde test met gegeven array:

Time taken: 0.00438

0123456789