Lesson 10

Introduction to algorithms Sorting algorithms

Bubble sort

Selection sort

Counting sort

IT TALENTS

Training Camp

Introduction to algorithms

The goal of an algorithm –
to solve a particular problem as efficiently as possible

IT TALENTS
Training Camp

Introduction to algorithms

Algorithm Complexity –
a measure for the efficiency of an algorithm

Introduction to algorithms

Time Complexity

how fast the algorithm is

IT TALENTS
Training Camp

Introduction to algorithms

Space Complexity

how much memory the algorithm requires

IT TALENTS
Training Camp

Introduction to algorithms

Big *O* Notation (simplified)

O(*N*) - the algorithm executes approximately *N* simple operations, where *N* is the size of the input

IT TALENTS
Training Camp

Introduction to algorithms

Popular complexities for input of size N

Number of operations	Big O	Short name
1, 5, 10, 10 000	$O(1)$	constant
$\log N$, $\log N + 10$	$O(\log N)$	logarithmic
N, $2N$, $10N + 5$	$O(N)$	linear
$N \log N$, $N \log 2N$	$O(N \log N)$	linearithmic
N^2, $3N^2 + N + 1$	$O(N^2)$	quadratic
N^3, $N^3 + N^2$	$O(N^3)$	cubic
N^K, K – constant	$O(N^K)$	polynomial
2^N, 3^N, $2^N + N^2$	$O(2^N)$	exponential
$N!$	$O(N!)$	factorial

IT TALENTS Training Camp

Introduction to algorithms

Complexity in the:

best case
average case
worst case

IT TALENTS
Training Camp

Sorting algorithms I

Sorting problem –
rearrange a sequence of items,
so that all the elements in the resulting sequence
are in increasing (decreasing) order

IT TALENTS
Training Camp

Sorting algorithms I

Sort an array of integers in increasing order

Mostly used in practice

- Usually using predefined function for sorting

```
public static void main(String[] args) {
		int[] array = {12, 3, 53, 6, 9, 3, 43, -4, 0};

		Arrays.sort(array);

		for (int i = 0; i < array.length; i++) {
			System.out.print(array[i] + " ");
		}
	}
```

IT TALENTS
Training Camp

Bubble Sort

- Simple sorting algorithm
- Worst case scenario of N^2 operations
- Best case scenario of N operations
- Used for small arrays

IT TALENTS
Training Camp

Bubble Sort Philosophy

- Traverse elements
- Compare adjacent elements
- Swap adjacent elements if the next element is greater

IT TALENTS
Training Camp

Bubble Sort Pseudo Code

- Compare array[0] & array[1]
- If array[0] > array [1] swap it.
- Compare array[1] & array[2]
- If array[1] > array[2] swap it.
- ...
- Compare array[n-1] & array[n]
- if array[n-1] > array[n] then swap it.
- Repeat the same steps for array[1] to array[n-1]

IT TALENTS
Training Camp

Try it!

Also try to optimize the algorithm.

Bubble Sort

```java
public static int[] bubbleSort(int[] intArray) {
        int temp = 0;

        for(int i=0; i < intArray.length; i++){
            for(int j=1; j < (intArray.length - i); j++){

                if(intArray[j-1] > intArray[j]){
                    //swap the elements!
                    temp = intArray[j-1];
                    intArray[j-1] = intArray[j];
                    intArray[j] = temp;
                }
            }
        }
        return intArray;
    }
```

IT TALENTS
Training Camp

Selection Sort

- Used only for small arrays
- Slow for bigger arrays
- Worst case scenario of N^2 operations
- Best case scenario of N^2 operations
- Find the next smallest element and move it to its final position
- Start by the smallest element
- Subset of sorted and unsorted elements

IT TALENTS
Training Camp

Selection Sort Algorithm

- Initially, the sorted sublist is empty and the unsorted sublist is the entire input list
- Start with first element of the unsorted subset
- Test with all elements after the assumed min element to find the smallest
- Swap the smallest element to the starting index

Try it!

Also try to optimize the algorithm.

Selection Sort

```java
public static int[] selectionSort(int[] a){
/* a[0] to a[n-1] is the array to sort */
    int temp = 0;
    int iMin = 0;
// advance the position through the entire array
// (could do j < n-1 because single element is also min element)
    for (int j = 0; j < a.length-1; j++) {
    /* find the min element in the unsorted a[j .. n-1] */
    /* assume the min is the first element */
    iMin = j;
    /* test against elements after j to find the smallest */
    for ( int i = j+1; i < a.length; i++) {
    /* if this element is less, then it is the new minimum */
        if (a[i] < a[iMin]) {
        /* found new minimum; remember its index */
            iMin = i;
        }
    }
```

IT TALENTS
Training Camp

Selection Sort

```
    /* iMin is the index of the minimum element.
       Swap it with the current position */
       if ( iMin != j ) {
       //swap the elements!
           temp = a[j];
           a[j] = a[iMin];
           a[iMin] = temp;
       }
    }
    return a;
}
```

IT TALENTS
Training Camp

Counting Sort

- Counting sort is an algorithm for sorting small integers.
- Its main idea is to count the number of different objects.
- Runs in linear running time in the number of items.

Counting Sort Algorithm

Basic implementation

- Find min/max values
- For every element, count in separate array the number of times it was met.
- Then for every *value* from min to max in the array

  ouput K times this value,where K is the number calculated in the previous array.

Counting Sort Algorithm

Input Data

0	4	2	2	0	0	1	1	0	1	0	2	4	2

Count Array

0	1	2	3	4
5	3	4	0	2

Sorted Data

0	0	0	0	0	1	1	1	2	2	2	2	4	4

Counting Sort Basic Version

```java
public static int[] countingSort(int[] numbers, int max)
{
   int[] counts = new int[max + 1];

   for (int i = 0; i < numbers.length; i++)
      counts[numbers[i]]++;

   int[] results = new int[numbers.length];

   int counter = 0;
   for(int i = 0; i <= counts.length-1; i++)
   {
       while(counts[i]>0)
       {
           results[counter++] = i;
           counts[i]--;
       }
   }

   return results;
}
```

Counting Sort Complexity

- Best Case – O(*N+K*)
- Average Case – O(*N+K*)
- Worst Case – O(*N+K*)

Where K is the largest element in sequence.

Sorting Complexity Refresh

	Best	Average	Worst
• Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$
• Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
• Counting Sort	$O(n+k)$	$O(n+k)$	$O(n+k)$
• Radix Sort	$O(n\lg_b n)$	$O(n\lg_b n)$	$O(n\lg_b n)$

Resources

Counting and Radix Sort Explained :

http://courses.csail.mit.edu/6.006/spring11/rec/rec11.pdf

Summary

- Algoritm Complexity
- Most used in practice
- Bubble Sort
- Selection Sort
- Counting Sort

IT TALENTS
Training Camp