03 January 2011

Algorithm Analysis Project

All project members have to be present at the time of the demo.
You need to submit your report and Java programs, including a readme.txt file that explains how to run your program.

Part I

Consider the following algorithms:

Group 1:	Selection Sort, Bubble sort, Insertion Sort
Group 2:	Merge Sort, Quick Sort

Choose one algorithm from Group 1 and one from Group 2. Implement the algorithms in Java. Your algorithm should take the array size as an input. You need to generate the array elements randomly. It should write the input array, the output arrays produced by each algorithm and the result into a file. Arrays should be printed into files in columns. The first column should be input array, the second and the third columns should be the results of the algorithms. Seperate the columns using tab character (\t). Let A be the input array, B and C be the results of the algorithms you have chosen. So the file format should be as follows:

```
Algorithm:                 Algorithm1        Algorithm2
Array Size:                (result here)     (result here)
Running Time (msec) :      (result here)     (result here)
No. of Key Comparsions:    (result here)     (result here)

Input          Algorithm1   Algorithm2
A[0]           B[0]         C[0]
A[1]           B[1]         C[1]
A[2]           B[2]         C[2]
................................
................................
```

Generate arrays of integers with given sizes below. Randomly generate array values.

- For the same array, measure the number of key comparisons and running time of the algorithms. (You need to use the same array for both algorithms in order to compare them.)
- Plot the values and compare them with their theoretical results.
- Compare algorithms with each other in terms of number of key comparisons and running time.

Array Size	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
Running Time (millisecond)										
Number of Key Comparsions										

- Your report should include results and your evaluation. Complete the table given for each of algorithm you have chosen.

You are not allowed to use Java libraries to sort the array!

Part II

Implement the knapsack problem in Java. Implement both approaches provided below. Your program should take an input file as an instance of the problem. A sample input file is as follows: (Input file format should be the same).

```
capacity = 5
item          weight          value
1             2               12
2             1               10
3             3               20
4             2               15
```

Approach 1: Apply the exhaustive search approach (Chapter 3 Section 3.4). Generate all the subsets of the set of n items given, compute the total weight and corresponding values of each subset to identify feasible subsets.

Approach 2: Apply dynamic programming approach.

Output file for approach 1: Print each subset and corresponding weight and value to a seperate line. At the end of the file print the result. Format:

```
item1, item2,...,itemi ; weight ; value
..........
..........
result: item1, item2,...,itemi ; weight ; value
time: (in milliseconds)
```

i.e.

```
.....
2,3 ; 4 ; 30
2,4 ; 3 ; 25
......
1,2,4 ; 5 ; 37
.....
result: 1,2,4 ; 5 ; 37
time: (in milliseconds)
```

Output file for approach 2: Print the table generated by dynamic programming. Print the result at the end of the file.

```
0     0     0     0     0     0
0     0     12    12    12    12
0     10    12    22    22    22
0     10    12    22    30    32
0     10    15    25    30    37
result: 1,2,4 ; 5 ; 37
time: (in milliseconds)
```

- Compare the running time of both approaches with each other. Compare them with their teoretical results.