Sorting

L'esercitazione si propone di realizzare, testare e confrontare alcuni algoritmi di sorting, su vari input generati dal driver che viene messo a disposizione. Gli algoritmi che possono essere realizzati e testati includono:

```
mergeSort
heapSort
selectionSort
insertionSort
quickSort
radixSort
bucketSort
```

Il file *utils.c* / *Utils.java* può essere facilmente modificato per testarne altri. Nel medesimo file è già implementato un *bubbleSort* che può essere usato per testare il driver.

Implementazione

Gli algoritmi vanno implementati nel file **sort.c** / **Sort.java**. Tutti gli algoritmi di ordinamento hanno segnature:

```
void <sort_alg>(array *);
```

```
public void <sort_alg>(int[] array);
```

In C Il tipo *array* è così definito:

```
typedef struct {
    int *arr;
    int size;
} array;
```

Compilazione ed esecuzione

Per compilare i file usiamo:

- il Makefile in C: `make`
- in Java: `javac *.java`

Una volta compilati i file, il driver si usa come di seguito descritto:

1. **test**: ai fini della correttezza: esegue <*sort_alg*> (uno di quelli su elencati) su un input di dimensione <*size_input*> , della tipologia <*tipo_input*>, stampando info sulla correttezza; `./driver test <sort_alg> <tipo_input> <size_input>` `java Driver test <sort_alg> <tipo_input> <size_input>`
2. **run**: esegue ordinamento con <*sort_alg*> su un input di tipo <*tipo_input*> e dimensione <*size_input*>; stampa tempo di elaborazione; `./driver run <sort_alg> <tipo_input> <size_input>` `java Driver run <sort_alg> <tipo_input> <size_input>`

3. **cmp**: confronta i tempi di esecuzione dei due algoritmi . `./driver cmp <nome-alg1> <nome-alg2> <tipoInput> <size>` `java Driver cmp <nome-alg1> <nome-alg2> <tipoInput> <size>`
4. **file**: come run, ma prendendo input da file; `./driver file <sort_alg> <file_name>` `java Driver file <sort_alg> <file_name>`
5. **gen**: genera un possibile input (tipologia <*tipo_input*>, dimensione <*size_input*>), scrivendolo nel file <*file_name*>; `./driver gen <tipo_input> <size_input> <file_name>` `java Driver gen <tipo_input> <size_input> <file_name>`

Gli input per gli ordinamenti possono essere generati randomicamente dal driver, scegliendone: la dimensione <*size_input*>, (la dimensione minima *MIN_SIZE* e massima *MAX_SIZE*, sono definite nel file *utils.h* / *Utils.java*) e la tipologia <*tipo_input*>:

- C (ordinato crescente),
- D (ordinato decrescente),
- R (random),
- c (quasi-ordinato crescente),
- d (quasi ordinato decrescente).