The Architecture:

April 14, 2009

The basic architecture of sortings done here consists of a header file *sortutil.h*, which includes a class named SORT - containing the *basic utilities* on array functions such as to generate random numbers - *genRand()*, printing the array *printArray()* and to check if the array elements are in non-descending order *checkSort()*. These functions are basic to every array containing integer numbers. For sake of simplicity, only integer values are used.

The SORT class also inherits another class named *sortdefinitions* from file *sortdefinitions.h*. The sortdefinitions.h file acts like a source of all the codes of the sorting techniques done here. It includes the function definitions of all the sorting techniques, heap sort, quick sort, bubble sort, selection sort, and thus the two classes *SORT* and *sortdefinitions* forms the header file *sortutil.h*.

A main function is just written for every sorting technique by including the header file *sortutil.h*. Thus there are four *main*'s for every sort. The basic idea here was to use the resources provided by the *sortutil.h* file and perform the necessary tasks. *A word on main:* A pointer to array has been used to dynamically allocate the size. A file pointer (fp) has been used to write the runtimes to a file, say, heap.dat, which eases to plot the graph. Each time the size of the array is incremented by an OFFSET value, the value being between RANGELOW and RANGEHIGH. A simple call to genRand with size 'i' will fetch 'i' random numbers into the array 'a'. All printArray calls can be unmasked to view the elements of the array, obviously when the size of array is very low. Then their is call to get the present time in seconds and microseconds, the start time of heap sorting. On calling heapSort, the 'i' elements of the array are heap sorted. Finally the end time is obtained from a call to gettimeofday. RunTime is calculated by subtracting start time from end time. The runtime is displayed on console as well as written to the file only if checkSort is successful. The array size is incremented and the process is iterated.

Running script:
1. Change to the sorts directory
2. Run the shell script by typing ./run at command prompt
3. Type a filename in *.cpp format [say, heap.cpp]
4. Script terminates by plotting the graph