Hybrid QuickSort

In last classes we learnt about running time and analysing of algorithms to figure out its running time. In this problem you are required to implement a hybrid version of quicksort based on a parameter and figure out the best value for the parameter depending on the running time.

Implement a hybrid sorting algorithm with an extra parameter k (`sort(A,k)`). The function applies Quicksort algorithm on array A till the size of partition is greater than k. When the size of any partition goes below k, it applies Insertion sort or Bubble sort on the partition of size less or equal to k.

Once you are ready with the routine `sort(A,k)`, vary k for different values and record the time required for running `sort(A,k)`. Subsequently, you are required to plot a graph of value of k against the time for running `sort(A,k)` and figure out the optimal value of k for which the running time is least.

Consider that the size of array to be in the order of 1,00,000 (n), and the value of k to be varied from 1 to 100. Please consider the following different conditions for input

1. Take all random numbers.
2. Take a sorted array.
3. Take reverse sorted array.
4. Take a half sorted array (the other half is random).

For each type of input there would be a seperate graph. Consider saving the ouput of your program to a file and then plotting the graph.

Once you have completed the above, try implementing the same with *Mergesort*. Implement a mergesort routine where once the size of the partition reaches below k, you apply bubblesort/insertion sort. Try finding whether usage of bubblesort and insertionsort changes the value of optimal k.

Notes:

- You will be using "GNUPlot" for plotting the graph. Please make sure that your system has GNUPlot installed.
- Write your programs only in C.
- Use the library functions from `ctime.h` to find out the running time of `sort(A,k)`.

- Use the same array for each call to `sort(A,k)` with different k.