Stat 598W: Homework 3

Due Saturday Feb 28 (sveinno@purdue.edu). Include a word/pdf document with your code and relevant output.

Part 1

Write a library of functions containing the following sorting routines:

- Selection sort.
- Insertion sort.
- Merge sort.
- Quick sort.
- Heap sort.
- A sorting method of your choice that was not covered in class.

Since writing a new sorting routine for each data type is clearly not feasible, you should use C++ templates to write a single generic function capable of handling different data types. It should take a reference to a Vector<T> as well as a function pointer so that the user can decide the ordering rule (ascending, descending,...). The function declaration should therefore be along the following lines:

```
template <typename T>
void sort(Vector<T>& v, int(*cmp)(T,T))
```

Part 2

Use your library to compare the performance of the sorting routines. Generate integer Vectors of sizes $n = 100, 1000, 10000, 20000, 30000, \ldots, 100000$, and estimate the time it takes to sort them. To do that you may use the clock function, exported by the standard ctime interface, as follows:

```
#include <ctime>
int main(){
  double start = double(clock()) / CLOCKS_PER_SEC;
  ...
  Perform calculations
  ...
  double finish = double(clock()) / CLOCKS_PER_SEC;
  double elapsed = finish - start;
}
```

Note that for each Vector size, you should generate multiple Vectors and take the average of the sorting times for each method. Also, the instructions above are only guidelines that don't have to be followed exactly. However, your report should include a comparison of the performance of the sorting routines, and a few words about your findings.

If time permits, and for extra points, you are welcome to add a hybrid sorting algorithm to the mix (as described in lecture 7), and see whether it improves the performance. Moreover, you can also consider Vectors of integers in a finite range, and implement a linear sort algorithm such as Counting Sort (see lecture 8), and see whether it outperforms the comparison sort algorithms. Recall that the linear sort algorithms had worst case complexity $O(n)$, while the worst case for comparison sorting was $O(n \log(n))$.