Median Finding & QuickSort

Median Finding

A test file has been added to source code.

It has 27 int numbers from 1 to 27, randomly reordered.

Inside sample.txt,

11, 12, 14, 13, 15, 5, 6, 7, 8, 9, 1, 2, 3,27, 4, 10, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,26

```
eddy@ubuntu:~/cs430project/median$ make
Scanning dependencies of target median
[ 50%] Building CXX object CMakeFiles/median.dir/median.cpp.o
[100%] Linking CXX executable median
[100%] Built target median
eddy@ubuntu:~/cs430project/median$ ./median
input a data file name
sample.txt
data loaded.
order statistics kth:
7
choose a median group, 3,5 or 7, or type 0 to choose random select
type 9 to run all 4 methods
9
group 3 7 th order statistics is 7
```

First in put the datafile name.

Then input order statistics k(k is start from 1 not 0).

Then type 3,5,7 to select median group number.

Or type 0 to use random selection methods.

Also, you can type 9 to run all 4 methods at the same time.

Quick Sort.

A testing file Intdata.txt has been included into source code as a testing file. Inside testing file, there are randomly 2000 integers range from -1000 to 1000.
Try to run with
./quicksort Intdata.txt
This will run 4 partition methods and print the running time.

```
MedianPartition<7> using 0.001727s
eddy@ubuntu:~/cs430project/quicksort$ ./quicksort Intdata.txt
data loaded.
RandomPartition using 0.00054s
MedianPartition<3> using 0.002116s
MedianPartition<5> using 0.001797s
MedianPartition<7> using 0.001714s
```

Run 5 times and get the time each methods used in second

runs	1st	2nd	3rd	4th	5th	Average
Random	0.000461	0.000517	0.00048	0.000471	0.00054	0.0004938
MedianGroup 3	0.002123	0.002188	0.002177	0.002149	0.002116	0.0021506
MedianGroup 5	0.001825	0.001807	0.001807	0.001777	0.001797	0.0018026
MedianGroup 7	0.001718	0.001711	0.001729	0.001727	0.001714	0.0017198

As we can see, the random select method is about 4 to 5 times faster than the median group methods.
And as the median group number grows, the sorting is running faster.