排序作业报告

1190501001 李恩宇

功能

sort 函数原型: `void x_sort(int *begin, int *end)`, 实现了:

- 二路归并排序 `merge_sort`
- 基数排序 `radix_sort` (基数可变, 默认 2^8)
- 堆排序 `heap_sort`
- 快速排序 `quick_sort`
- 冒泡排序 `bubble_sort`
- 选择排序 `selection_sort`
- 插入排序 `insertion_sort`

生成测试数据并运行

1. 生成测试数据 (Debian GNU/Linux 10 (buster) on Windows 10 x86_64):
   - 编译测试数据生成器

     ```
     cd /path/to/sort
     gcc -Wall descending_gen.c -o descending_gen.out
     gcc -Wall ascending_gen.c -o ascending_gen.out
     gcc -Wall random_gen.c -o random_gen.out
     ```

   - 生成测试数据

     ```
     echo '10000 20000 50000 70000 100000 200000 500000 700000 1000000' |
     ./descending_gen.out > ./descending_data.txt
     echo '10000 20000 50000 70000 100000 200000 500000 700000 1000000' |
     ./ascending_gen.out > ./ascending_data.txt
     echo '10000 20000 50000 70000 100000 200000 500000 700000 1000000
     2000000 5000000 7000000 10000000 20000000 50000000 70000000
     100000000' | ./random_gen.out > ./random_data.txt
     echo '10' | ./random_gen.out > ./test_data.txt
     ```

2. 验证排序算法正确性 (Debian GNU/Linux 10 (buster) on Windows 10 x86_64):

   ```
   cd /path/to/sort
   g++ -Wall -O2 test.cpp sort.cpp -o test.out
   ./test.out < test_data.txt
   ```

3. 测试环境: cmd.exe @ Windows 10 Professional 64-bit, Intel Core i7 8550U @ 1.80GHz
   - 编译指令:

     ```
     cd /path/to/sort
     g++ -Wall -O2 main.cpp sort.cpp -o main.exe
     ```

- 测试排序耗时:

```
main.exe < random_data.txt & REM 随机数据
main.exe < ascending_data.txt & REM 顺序数据
main.exe < descending_data.txt & REM 逆序数据, 快速排序可能发生栈溢出, 需修改栈大小限制
```

4. 测试环境: bash.exe @ Ubuntu 18.04.3 LTS x86_64, Intel Xeon X5650 @ 2.67GHz
   - 编译指令:

```
cd /path/to/sort
g++ -Wall -O2 main.cpp sort.cpp -o main.out
```

   - 测试排序耗时:

```
./main.out < random_data.txt # 随机数据
./main.out < ascending_data.txt # 顺序数据
./main.out < descending_data.txt # 逆序数据
```

各排序算法实测性能

随机数据: `random_data.txt`

data size	merge sort	radix sort	heap sort	quick sort	bubble sort	selection sort	insertion sor
10000	0.006	0.001	0.001	0.000	0.136	0.035	0.02
20000	0.016	0.001	0.002	0.001	0.559	0.119	0.09
50000	0.031	0.001	0.005	0.004	3.817	0.826	0.64
70000	0.037	0.002	0.008	0.005	7.779	1.668	1.26
100000	0.053	0.001	0.010	0.007	15.286	3.428	2.65
200000	0.096	0.003	0.023	0.016	62.686	14.200	10.72
500000	0.216	0.012	0.068	0.043	407.253	82.481	67.28
700000	0.304	0.016	0.104	0.062	timeout	182.407	140.43
1000000	0.418	0.025	0.169	0.089	timeout	358.780	300.27
2000000	0.859	0.052	0.359	0.181	timeout	timeout	timeou
5000000	2.186	0.144	1.377	0.673	timeout	timeout	timeou
7000000	3.133	0.076	1.703	0.673	timeout	timeout	timeou
10000000	4.331	0.249	2.889	1.038	timeout	timeout	timeou
20000000	9.111	0.532	5.975	2.079	timeout	timeout	timeou
50000000	22.808	1.472	17.530	5.826	timeout	timeout	timeou
70000000	32.678	2.063	25.535	7.571	timeout	timeout	timeou
100000000	45.997	2.902	38.469	10.891	timeout	timeout	timeou

顺序数据: `ascending.txt`

data size	merge sort	radix sort	heap sort	quick sort	bubble sort	selection sort	insertion sort
10000	0.006	0.001	0.000	0.016	0.000	0.030	0.016
20000	0.012	0.001	0.001	0.062	0.000	0.116	0.057
50000	0.025	0.001	0.003	0.404	0.000	0.707	0.352
70000	0.044	0.001	0.005	0.907	0.000	1.377	0.687
100000	0.044	0.003	0.006	2.134	0.000	2.878	1.487
200000	0.084	0.006	0.016	8.849	0.000	11.637	6.106
500000	0.189	0.011	0.037	54.823	0.000	76.187	39.644
700000	0.268	0.016	0.057	103.516	0.001	148.814	76.178
1000000	0.365	0.021	0.071	200.987	0.001	279.007	142.795

逆序数据: `descending.txt`

data size	merge sort	radix sort	heap sort	quick sort*	bubble sort	selection sort	insertion sort
10000	0.004	0.001	0.000	0.052*	0.067	0.035	0.034
20000	0.010	0.000	0.001	0.188*	0.266	0.131	0.122
50000	0.023	0.001	0.003	1.174*	1.577	0.689	0.714
70000	0.034	0.001	0.005	2.304*	3.141	1.368	1.396
100000	0.046	0.001	0.005	4.697*	6.478	2.817	2.940
200000	0.082	0.003	0.012	18.804*	25.369	11.205	12.012
500000	0.189	0.010	0.031	117.550*	166.622	70.259	76.886
700000	0.289	0.015	0.046	230.190*	320.107	140.996	149.185
1000000	0.356	0.019	0.072	469.676*	661.972	284.351	309.463

> * Since the stack overflowed on Windows 10, we got this data on a Linux server powered by Ubuntu 18.04.3 LTS x86_64 with Intel Xeon X5650 @ 2.67GHz.