排序算法一览

曹逸君 1352872

排列一万个不重复正整数表现:

桶排序 bucket_sort

20000 loops/recursive calls in 0.010 seconds

由于使用 bytearray 实现,具有惊人的效率,复杂度始终为 2n.遍历一遍原始列表在 bytearray 中标记.遍历一遍 bytearray 输出.

额外空间为 max_num/8 字节, 当有重复数字时将增至 max_num 字节.

实际应用中桶排序通常将数据分块再交给其他排序算法排序.[插入,快排之类]

基数排序 radix_sort

50000 loops/recursive calls in 0.030 seconds

作为最早的非比较排序算法基数排序简直旋酷吊炸天.这里实现用的是 LSD(Least significant digital).时间复杂度始终为 m*n.由于这里实现用的是十进制位.所以 m=log(max_num)=log(9999)=4. 当然还要加上找到最大值的时间.通常是 n. 但悲剧的是只要有一个 5 位数.LSD 实现就得再加个 n.MSD 从高位开始分桶可以进一步优化基数排序.

实际中基数排序用在字符串上排列字典序之类会比较好.

希尔排序 shell_sort

109264 loops/recursive calls in 0.047 seconds

作为对插入排序的改良.希尔排序效果惊人的好.在某些情况下[通常是数据量中等的时候(我测试结果是 1W 略优于快排,10W 差于快排)]

详 见 http://faculty.simpson.edu/lydia.sinapova/www/cmsc250/LN250_Weiss/L12-ShellSort.htm#increments

PS: 希尔排序的维基页面把 Sedgewick 的序列公式写错了.我已修复.维基百科 希尔排序

快速排序 quick_sort

161117 loops/recursive calls in 0.048 seconds

十分稳定高效的算法

堆排序 heap_sort

161491 loops/recursive calls in 0.158 seconds

不常用.堆倒是经常会用到. 同样还有 AVL 排序[建 AVL 树再输出]

因为建立/维护数据结构的开销不小

合并排序 merge_sort

124352 loops/recursive calls in 0.167 seconds

很棒的算法

插入排序 insert_sort

24895069 loops/recursive calls in 1.149 seconds

在小数据量的时候用的比较多. Python 的 Timsort 便是用合并排序+插入排序完成的.

选择排序 selection_sort

50005000 loops/recursive calls in 3.905 seconds

old school algorithm

冒泡排序 bubble_sort

24831636 loops/recursive calls in 8.074 seconds

older school algorithm

附录测试数据:

```
c19@Acer:/media/c19/DATA/project/数据结构/p10_排序算法概览$ ./sortings.py
         bucket_sort  sorting  10000 ints
      21258 function calls in 0.010 seconds
Ordered by: cumulative time
```

ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 0.010 0.010 /usr/lib/python2.7/cProfile.py:146(runcall)
1 0.005 0.005 0.010 0.010 ./sortings.py:138(bucket_sort)
10000 0.004 0.000 0.004 0.000 ./sortings.py:146(mark)
10000 0.001 0.000 0.001 0.000 {method 'append' of 'list' objects}
1 0.000 0.000 0.000 0.000 {max}
1251 0.000 0.000 0.000 0.000 ./sortings.py:145()
2 0.000 0.000 0.000 0.000 {range}
1 0.000 0.000 0.000 0.000 {method 'enable' of '_lsprof.Profiler' objects}
1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects}

radix_sort sorting 10000 ints

80055 function calls in 0.030 seconds

Ordered by: cumulative time
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 0.030 0.030 /usr/lib/python2.7/cProfile.py:146(runcall)
1 0.015 0.015 0.030 0.030 ./sortings.py:160(radix_sort)
40000 0.012 0.000 0.012 0.000 ./sortings.py:170(digit)
40000 0.003 0.000 0.003 0.000 {method 'append' of 'list' objects}
1 0.000 0.000 0.000 0.000 {max}
4 0.000 0.000 0.000 0.000 ./sortings.py:165(flatten)
40 0.000 0.000 0.000 0.000 {method 'extend' of 'list' objects}
5 0.000 0.000 0.000 0.000 {range}
1 0.000 0.000 0.000 0.000 {len}
1 0.000 0.000 0.000 0.000 {method 'enable' of '_lsprof.Profiler' objects}
1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects}

shell_sort sorting 10000 ints

39 function calls in 0.047 seconds

Ordered by: cumulative time
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 0.047 0.047 /usr/lib/python2.7/cProfile.py:146(runcall)
1 0.047 0.047 0.047 0.047 ./sortings.py:182(shell_sort)
1 0.000 0.000 0.000 0.000 ./sortings.py:189(sedgewick_seq)
11 0.000 0.000 0.000 0.000 {next}
7 0.000 0.000 0.000 0.000 ./sortings.py:190(seq1)
6 0.000 0.000 0.000 0.000 ./sortings.py:195(seq2)
9 0.000 0.000 0.000 0.000 {method 'append' of 'list' objects}
1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects}
1 0.000 0.000 0.000 0.000 {len}
1 0.000 0.000 0.000 0.000 {method 'enable' of '_lsprof.Profiler' objects}

quick_sort sorting 10000 ints

48398 function calls (35072 primitive calls) in 0.048 seconds

Ordered by: cumulative time
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 0.048 0.048 /usr/lib/python2.7/cProfile.py:146(runcall)
13327/1 0.030 0.000 0.048 0.048 ./sortings.py:30(quick_sort)

6663 0.015 0.000 0.017 0.000 /usr/lib/python2.7/random.py:291(sample)
19990 0.001 0.000 0.001 0.000 {len}
6663 0.001 0.000 0.001 0.000 {method ‘random’ of ‘_random.Random’ objects}
876 0.001 0.000 0.001 0.000 {hasattr}
876 0.000 0.000 0.000 0.000 {method ‘add’ of ‘set’ objects}
1 0.000 0.000 0.000 0.000 {method ‘enable’ of ‘_lsprof.Profiler’ objects}
1 0.000 0.000 0.000 0.000 {method ‘disable’ of ‘_lsprof.Profiler’ objects}

heap_sort sorting 10000 ints

369919 function calls in 0.158 seconds

Ordered by: cumulative time
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 0.158 0.158 /usr/lib/python2.7/cProfile.py:146(runcall)
1 0.009 0.009 0.158 0.158 ./sortings.py:90(heap_sort)
10000 0.009 0.000 0.125 0.000 ./sortings.py:125(pop_min)
10000 0.084 0.000 0.114 0.000 ./sortings.py:110(shift_down)
10000 0.007 0.000 0.022 0.000 ./sortings.py:98(add)
118504 0.016 0.000 0.016 0.000 ./sortings.py:92(lchild)
118504 0.014 0.000 0.014 0.000 ./sortings.py:94(rchild)
10000 0.010 0.000 0.014 0.000 ./sortings.py:102(shift_up)
22906 0.004 0.000 0.004 0.000 ./sortings.py:96(parent)
40001 0.003 0.000 0.003 0.000 {len}
20000 0.002 0.000 0.002 0.000 {method ‘append’ of ‘list’ objects}
10000 0.001 0.000 0.001 0.000 {method ‘pop’ of ‘list’ objects}
1 0.000 0.000 0.000 0.000 {method ‘disable’ of ‘_lsprof.Profiler’ objects}
1 0.000 0.000 0.000 0.000 {method ‘enable’ of ‘_lsprof.Profiler’ objects}

merge_sort sorting 10000 ints

641298 function calls (629492 primitive calls) in 0.167 seconds

Ordered by: cumulative time
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 0.167 0.167 /usr/lib/python2.7/cProfile.py:146(runcall)
11807/1 0.129 0.000 0.167 0.167 ./sortings.py:65(merge_sort)
513159 0.029 0.000 0.029 0.000 {len}
116329 0.009 0.000 0.009 0.000 {method ‘append’ of ‘list’ objects}
1 0.000 0.000 0.000 0.000 {method ‘disable’ of ‘_lsprof.Profiler’ objects}
1 0.000 0.000 0.000 0.000 {method ‘enable’ of ‘_lsprof.Profiler’ objects}

insert_sort sorting 10000 ints

10003 function calls in 1.149 seconds

Ordered by: cumulative time
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 1.149 1.149 /usr/lib/python2.7/cProfile.py:146(runcall)
1 1.136 1.136 1.149 1.149 ./sortings.py:36(insert_sort)
9991 0.014 0.000 0.014 0.000 {method ‘insert’ of ‘list’ objects}
8 0.000 0.000 0.000 0.000 {method ‘append’ of ‘list’ objects}
1 0.000 0.000 0.000 0.000 {method ‘enable’ of ‘_lsprof.Profiler’ objects}
1 0.000 0.000 0.000 0.000 {method ‘disable’ of ‘_lsprof.Profiler’ objects}

selection_sort sorting 10000 ints

30005 function calls in 3.905 seconds

Ordered by: cumulative time
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 3.905 3.905 /usr/lib/python2.7/cProfile.py:146(runcall)
1 0.025 0.025 3.905 3.905 ./sortings.py:49(selection_sort)
10000 3.879 0.000 3.879 0.000 {min}
10000 0.001 0.000 0.001 0.000 {method 'append' of 'list' objects}
10001 0.001 0.000 0.001 0.000 {len}
1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects}
1 0.000 0.000 0.000 0.000 {method 'enable' of '_lsprof.Profiler' objects}

bubble_sort sorting 10000 ints

10005 function calls in 8.074 seconds

Ordered by: cumulative time
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 8.074 8.074 /usr/lib/python2.7/cProfile.py:146(runcall)
1 7.846 7.846 8.074 8.074 ./sortings.py:58(bubble_sort)
10000 0.229 0.000 0.229 0.000 {range}
1 0.000 0.000 0.000 0.000 {len}
1 0.000 0.000 0.000 0.000 {method 'enable' of '_lsprof.Profiler' objects}
1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects}