Algorithm PA1 report

EE2 B05901172 黃禹靖

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case  Size (words) | Insertion sort | | Merge sort | | Heap sort | | Quick sort | |
| Time  (sec) | Memory  (KB) | Time  (sec) | Memory  (KB) | Time  (sec) | Memory  (KB) | Time  (sec) | Memory  (KB) |
| Case 1  1362 | 0.02 | 14108 | 0.016 | 14108 | 0.016 | 14108 | 0.016 | 14108 |
| Case 2  9216 | 0.32 | 15476 | 0.055 | 14856 | 0.057 | 14712 | 0.055 | 14712 |
| Case 3  82360 | 24.3 | 26504 | 0.418 | 20484 | 0.46 | 19840 | 0.415 | 19840 |
| Case 4  185462 | 227 | 44108 | 0.952 | 27256 | 1.088 | 25808 | 0.958 | 25808 |
| Case 5  187607 | 142 | 40092 | 0.943 | 27244 | 1.079 | 25780 | 0.951 | 25780 |
| Case 6  37555 | 4.36 | 19684 | 0.198 | 17220 | 0.207 | 16924 | 0.196 | 16924 |
| Case 7  140106 | 77 | 35728 | 0.726 | 26504 | 0.802 | 25408 | 0.712 | 25408 |

1. The memory is measured by top command.
2. It seems that case 4 is an exception. Both time and space cost by case 4 are greater than which of case 5 even though the size of case 5 is slightly greater than the size of case 4.

So the plot below would not contain case 4 but other 6 cases.

Insertion sort

1. Algorithm flow:
2. The sort is like the pseudocode in lecture.
3. Data structure:
4. Vector of pair <string,size\_t>, where string is the word and size\_t is the appearing order.
5. Discussion:
6. The strings are not memorized additionally besides Algparser in merge sort, heap sort, quick sort. But the efficiency of insertion sort is too awful, so the strings are memorized in insertion sort to avoid calling Algparser::Querystring() every time. So the space cost in insertion sort is greater than that of the other 3 sort algorithm.

Merge sort

1. Algorithm flow:
2. It is like the pseudocode in lecture.
3. Data structure:
4. Vector<size\_t>, where size\_t represents the appearing order of a words.

To query the string, call Algparser::Querystring() every time.

1. Discussion:
2. In merge sort, additionally vector or array is required when merging two array. To avoid repeated construct and destruct vectors, I construct a vector in global and use it in merge and clear it every time.

Heap sort

1. Algorithm flow:
2. Different from pseudocode in lecture, it heapify every time when push in a new element, rather than push in all elements then do heapifying.
3. Data structure:
4. Vector<size\_t>, where size\_t represents the appearing order of a words.

To query the string, call Algparser::Querystring() every time.

1. Using a vector to implement a tree structure. For ith element, (2\*i+1)th element is its left child, and (2\*i+2)th element is its right child.
2. Discussion:
3. None.

Quick sort

1. Algorithm flow:
2. It is like the pseudocode in lecture.
3. Data structure:
4. Vector<size\_t>, where size\_t represents the appearing order of a words.

To query the string, call Algparser::Querystring() every time.

1. Discussion:
2. It picks first element of an array as key to parting it.