

**Hướng dẫn:** Trương Tấn Khoa, Phạm Trọng Nghĩa

SORT

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

MỤC LỤC

1. [THÔNG TIN 1](#_Toc106084592)
2. [BÁO CÁO 2](#_Toc106084593)
3. [THAM KHẢO 3](#_Toc106084594)

# THÔNG TIN

Cá Nhân

|  |  |  |
| --- | --- | --- |
| MSSV | HỌ TÊN | EMAIL |
| 19127517 | Hồ Thiên Phước | htphuoc19@clc.fitus.edu.vn |

# BÁO CÁO

**Task 3**

(file size: 370105 rows)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a | b | c | d |
| Bubble sort | Large number | Large number | Large number | Large number |
| Selection sort | Large number | Large number | Large number | Large number |
| Insertion sort | Large number | 66 | Large number | Large number |
| Merge sort | 1618 | 1134 | 1106 | 1186 |
| Quick sort | 1256 | 918 | 936 | 1681 |
| Radix sort | 1477 | 1440 | 1459 | 1473 |

(time unit: ms)

We consider the cases:

- Random list (a): Radix sort is the best algorithm because it just runs O(k \* n) (k is the max number of characters of the names) but the names are quite short .

- Sorted list (b): Insertion sort is the best because it just runs O(n) (it didn't actually sort (slide and insert elements) anything it just runs from the start to the end of the list).

- Reverse sorted list (c): Radix sort is the best algorithm, as we considered in the random list, its time complexity is quite stable in every case and very fast because the sorted names are short. Meanwhile, the bubble sort is very costly, because it took too many swap actions (and swap actions in C++ are very considerable).

- Almost sorted (d): Like the sorted list case, the insertion sort just slides a few elements to sort the list, so it’s the best algorithm in this case.

# 

# THAM KHẢO

[1] Hồ Nhật Linh