Lớp:

**BÁO CÁO KẾT QUẢ THỬ NGHIỆM**

Thời gian thực hiện: 011/03 – 16/03/2022

**Sinh viên thực hiện:**

**Nội dung báo cáo:**

1. ***Kết quả thử nghiệm***
   1. ***Bảng thời gian thực hiện[[1]](#footnote-1)***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dữ liệu** | **Thời gian thực hiện (ms)** | | | | |
| **Quicksort** | **Heapsort** | **Mergesort** | **sort (C++)** | **sort (numpy)** |
| 1 | 303 | 593 | 122 | 86 | 13.59 |
| 2 | 693 | 574 | 106 | 78 | 12.75 |
| 3 | 367 | 763 | 233 | 221 | 13.59 |
| 4 | 358 | 395 | 221 | 217 | 12.85 |
| 5 | 359 | 428 | 230 | 228 | 12.74 |
| 6 | 369 | 402 | 239 | 229 | 12.64 |
| 7 | 360 | 384 | 232 | 225 | 13.49 |
| 8 | 375 | 403 | 242 | 246 | 13.62 |
| 9 | 365 | 390 | 216 | 226 | 13.56 |
| 10 | 358 | 385 | 246 | 226 | 12.99 |
| Trung bình | 390.7 | 471.7 | 208.7 | 198.2 | 13.182 |

* 1. ***Biểu đồ (cột) thời gian thực hiện***

1. ***Kết luận:***

* Heapsort có thời gian chạy cao nhất, đặc biệt ở một số lần chạy (cột màu cam).
* Quicksort và Mergesort có hiệu suất khá ổn định nhưng vẫn chậm hơn sort(C++) và sort(numpy).
* sort(C++) (màu vàng) và sort(numpy) (màu xanh) có thời gian chạy thấp nhất, cho thấy mức độ tối ưu cao hơn.
* sort(numpy) có thời gian chạy cực kỳ nhỏ, gần như không đáng kể.
* Trung bình, sort(C++) và sort(numpy) là lựa chọn tối ưu nhất về tốc độ.

1. ***Thông tin chi tiết – link github, trong repo gibub cần có***
   1. Báo cáo
   2. Mã nguồn
   3. Dữ liệu thử nghiệm

Link github: https://github.com/odixe06/sortalgorithm

1. Số liệu chỉ mang tính minh họa [↑](#footnote-ref-1)