Lớp: IT003.P21.CTTN

**BÁO CÁO KẾT QUẢ THỬ NGHIỆM**

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

**Sinh viên thực hiện: Vũ Hoàng Long**

**Nội dung báo cáo: Thực nghiệm các giải thuật sắp xếp**

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 | 86 | 265 | 110 | 131 | 237 |
| 2 | 64 | 220 | 110 | 60 | 227 |
| 3 | 38 | 261 | 165 | 121 | 232 |
| 4 | 50 | 287 | 203 | 138 | 249 |
| 5 | 67 | 315 | 255 | 146 | 276 |
| 6 | 46 | 297 | 222 | 141 | 258 |
| 7 | 68 | 319 | 223 | 150 | 270 |
| 8 | 43 | 307 | 222 | 162 | 244 |
| 9 | 43 | 306 | 197 | 127 | 235 |
| 10 | 38 | 261 | 181 | 120 | 225 |

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

1. ***Kết luận:***
2. **Quicksort nhanh nhất**:
   * Quicksort có thời gian chạy thấp nhất trong hầu hết các lần thử nghiệm, dao động từ **38ms đến 86ms**.
   * Điều này khẳng định rằng Quicksort thường có hiệu suất cao trong thực tế, đặc biệt là khi dữ liệu ngẫu nhiên.
3. **Heapsort chậm nhất**:
   * Heapsort có thời gian chạy lớn nhất, từ **220ms đến 319ms**.
   * Điều này phù hợp với lý thuyết, vì mặc dù Heapsort có độ phức tạp **O(n log n)**, nó có nhiều phép hoán đổi hơn so với Quicksort và Mergesort.
4. **Mergesort trung bình**:
   * Thời gian chạy của Mergesort dao động từ **110ms đến 255ms**, nhanh hơn Heapsort nhưng chậm hơn Quicksort.
   * Điều này hợp lý vì Mergesort có độ phức tạp **O(n log n)** nhưng có chi phí bộ nhớ cao hơn.
5. **sort (C++) hiệu quả**:
   * Hàm sort() trong C++ chạy ổn định và khá nhanh, với thời gian từ **60ms đến 162ms**.
   * Hàm này thường sử dụng **Introsort** (kết hợp giữa Quicksort, Heapsort và Insertion Sort), giúp tối ưu thời gian sắp xếp.
6. **sort (numpy) kém hiệu quả nhất**:
   * Thời gian chạy của sort(numpy) dao động từ **225ms đến 276ms**, chậm hơn hầu hết các thuật toán khác.
   * Điều này có thể do numpy.sort() sử dụng **Timsort** hoặc **Mergesort**, vốn có chi phí bộ nhớ cao hơn và không tối ưu.

**Kết luận chung:** Nếu không có yêu cầu đặc biệt (như sắp xếp ổn định hay bộ nhớ hạn chế), **Quicksort hoặc sort() trong C++ là lựa chọn tốt nhất** để sắp xếp dữ liệu nhanh chóng.

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: [UITlogn/DSA-sort](https://github.com/UITlogn/DSA-sort)

1. [↑](#footnote-ref-1)