**BÁO CÁO KẾT QUẢ THỬ NGHIỆM**

Thời gian thực hiện: 27/2/2024-12/3/2024

**Sinh viên thực hiện: Đồng Quốc Thắng**

**Nội dung báo cáo:**

1. ***Kết quả thử nghiệm***
2. ***Bảng thời gian thực hiện***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dữ liệu | Thời gian thực hiện( tính bằng s) | | | | |
| Quicksort | Heapsort | Mergesort | Sort(c++) | Sort(numpy) |
| 1 | 7.728 | 15.259 | 10.157 | 0.070835 | 0.056 |
| 2 | 8.304 | 15.488 | 10.737 | 0.059607 | 0.065 |
| 3 | 9.471 | 18.414 | 14.151 | 0.201224 | 0.112 |
| 4 | 9.16 | 18.641 | 14.332 | 0.201829 | 0.112 |
| 5 | 9.161 | 18.522 | 14.646 | 0.202771 | 0.113 |
| 6 | 9.026 | 18.735 | 14.638 | 0.202948 | 0.114 |
| 7 | 9.131 | 20.212 | 14.7 | 0.204010 | 0.114 |
| 8 | 8.529 | 19.728 | 14.641 | 0.208814 | 0.113 |
| 9 | 9.132 | 19.731 | 14.828 | 0.210256 | 0.113 |
| 10 | 8.754 | 19.653 | 14.582 | 0.213515 | 0.114 |
| Trung bình | 8.8396 | 18.4383 | 13.7412 | 0.1775809 | 0.1026 |

1. ***Biểu đồ (cột) thời gian thực hiện***
2. **Kết luận:**

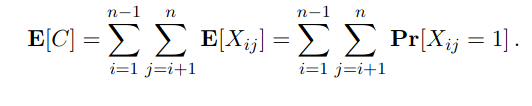
-So sánh giữa các thuật toán sort mà em tự cài:

+) Heap sort, Merge sort và Quick sort đều có độ phức tạp trong average case là O(nlogn) nên ta có thể thấy thời gian thực hiện của các thuật toán sắp xếp này sai khác nhau một hệ số( ví dụ như thời gian thực hiện heap sort gần như gấp 2 lần thời gian thực hiện quicksort)

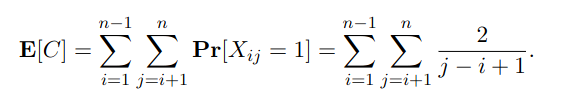
+) Quicksort mặc dù có độ phức tạp trong trường hợp tệ nhất là O(n2 ) nhưng theo tính xác xuất thì tỉ lệ chọn được 1 pivot khá tốt(có thể chia mảng thành 2 phần 25-75 hoặc đều hơn) là 50% nên khi random rất dễ được 1 pivot khá tốt. Quick sort còn không cần thêm bộ nhớ ngoài 1 list các phần tử input ban đầu, các operation trên list đều chỉ là swap value nên thời gian thực hiện khá nhanh so với 2 thuật toán còn lại. Quick sort dựa vào pivot random nên không ổn định bằng merge sort nhưng khá ổn định vì với số lượng phần tử khá lớn nên việc random sẽ được tính theo xác xuất

Gọi Xij là số lần phần tử nhỏ thứ i của list input được so sánh với phần tử thứ j của list input. Ta có Xij <= 1( vì trong 1 lần partition, i và j có thể hoặc có thể không được so sánh với nhau, chỉ được so sánh với nhau khi i hoặc j được chọn làm pivot)

Số lần được so sánh trung bình là:L



Với E(C) là số lần so sánh 2 phần tử trung bình sau khi thực hiện quick sort



(tỉ lệ i hoặc j được chọn làm pivot)

phương trình sẽ giải được về dạng O(nlogn) với hằng số khá nhỏ, vì vậy nên quick sort nhanh hơn các thuật toán sắp sếp khác.

+) Merge sort có thời gian ở mọi case là O(nlogn), và là một thuật toán khá ổn định, cần các operation như slicing, copy list nên mặc dù là O(nlogn) nhưng hằng số khá lớn nên thời gian chạy sẽ cao hơn quick sort. Một điểm trừ của merge sort là cần thêm bộ nhớ đệm khá lớn.

+) Heap sort mặc dù trên lý thuyết có thời gian chạy lý tưởng cho các thuật toán sắp xếp là O(nlogn), nhưng trên thực tế, heapsort có thời gian chạy chậm hơn các thuật toán khác khá nhiều vì mỗi lần loại bỏ được một phần tử ra khỏi cấu trúc dữ liệu heap thì phải điều chỉnh lại cấu trúc dữ liệu nên mặc dù có độ phức tạp nhỏ nhưng có hằng số khá lớn( khoảng gấp đôi quick sort dựa vào đồ thị trên). Ngoài ra heap sort còn khá không ổn đỉnh(có thể thấy dựa trên biểu đồ), lý do là vì với mỗi bước, nó thay đổi thứ tự tương đối của các phần tử trong list khá nhiều( để giữ vững cấu trúc heap) nên heap sort không ổn định.

-So sánh với sort được cung cấp trong numpy và C++:

+) Có thể thấy các thuật toán sort tự implement rất chậm so với các thuật toán được cung cấp từ trong các ngôn ngữ lập trình, có khá nhiều lý do, em nghĩ có các lý do chính sau:

++) Các hàm sort được cung cấp sẵn được người ta thiết kế với những thuật toán phức tạp và hiệu quả, ổn định hơn(ví dụ: python dùng Timsort)

++) Được implement một cách gần như hoàn hảo và implement từ các ngôn ngữ bậc thấp hơn và được compile sẵn thay vì những script python được viết và operate trên những cấu trúc dữ liệu tốn nhiều chi phi hơn( list)

++) Sort của numpy có thời gian chạy rất nhanh( nhanh hơn sort của C++) vì numpy có cấu trúc dữ liệu của riêng nó, vừa có hiệu quả cao về truy xuất bộ nhớ, vừa có hiệu quả cao vì implementation tốt

++) Một lý do khác mà thời gian thực hiện của các thuật toán tự viết là khá lớn là vì em sử dụng python nên thời gian thực hiện cũng khá lâu(so với các ngôn ngữ như C và C++)

**III) Thông tin chi tiết:**

Các thông tin chi tiết về log của các lần chạy thử, mã nguồn và bộ dữ liệu mà em thử nghiệm đều được upload tại link github sau: