Sinh viên: Bùi Nguyễn Kim Hải

Mã số sinh viên: 22520377

Lớp học: KHTN2022

Github: whynotkimhari · GitHub

Repository: clickme

****Disclaimer: Mọi số liệu trong báo cáo sẽ không là số liệu chính xác với mọi trường hợp sinh test case, cũng như sẽ không nói lên được điều gì quá khẳng định về chất lượng của các thuật toán một các toàn vẹn nhất.*

****P/s Laptop được sử dụng làm bài báo cáo này là Dell Inspiron 5584 – khá cũ nên sẽ không cho hiệu năng tốt -> số liệu chỉ để tham khảo.*

SORTING METHODS REPORT

1. Tạo data cho báo cáo.

1.1 Thông tin bộ data:

Thông tin data	Xếp tăng dần	Xếp giảm dần	Xếp ngẫu nhiên
Số thực Một triệu phần tử Được lưu trữ trong folder data	1 bộ (data1.txt)	1 bộ (data2.txt)	8 bộ (còn lại)

1.2 Cách tạo ra data:

- Chương trình khởi tạo: generate.py trong folder data.

- Công thức random theo format, sử dụng thư viện random.

- Không mất tính tổng quát, ta cũng nhận thêm các giá trị âm qua việc sử dụng random.choice() để chọn tính chất âm dương cho số. Thiết lập như sau:

isPositive = [True, False]

choice_ = random.choice(isPositive)

if choice_:

    current_ = random.random() * random.randint(0, 1000000)

else:

    current_ = random.random() * random.randint(0, 1000000) * (-1)

- Sau đó ép kiểu xuống chỉ nhận 4 số sau dấu " , " để tương ứng với cài đặt đã thiết lập bên chương trình chính để tạo hiệu quả tốt nhất:

file.write(f"{current_:.4f} ")

2. Chương trình chính

2.1 Mô tả sơ bộ

- Chương trình được viết bằng ngôn ngữ C++, được chia làm ba phần chính gồm Headers, Main, và Footers.

- Headers là nơi khai báo các hàm được viết và chạy trong chương trình gồm QuickSort, Partition, heapify, HeapSort, merge, và MergeSort.

- Main là nơi hàm main được chạy. Hàm main sẽ gọi các hàm còn lại trong chương trình với ý tưởng như sau. Với mỗi lần chạy ta sẽ tạo ra một đường dẫn đến file output đích và truy cập vào nó. Với mỗi đường dẫn đích được mở, ta truy cập vào folder data và chạy chương trình sắp xếp tương ứng với 10 bộ dữ liệu sau đó ghi nhận kết quả tại file đích đã mở.

- Để việc tổ chức được đơn giản, ta đặt tên các bộ dữ liệu có dạng data_.txt, $0 < \_ < 11$, được lưu trong folder data. Và để nhận biết được sẽ chạy phương thức sắp xếp nào thì ta sẽ quy định rõ như sau:

    + Vòng lặp lớn thứ nhất – i = 0 – QuickSort.

    + Vòng lặp lớn thứ hai – i = 1 – HeapSort.

    + Vòng lặp lớn thứ ba – i = 2 – MergeSort.

    + Vòng lặp lớn thứ tư – i = 3 – IntroSort.

2.2 Mô tả chi tiết

**Để tránh bài báo cáo bị dài không đáng có do trình bày code, code sẽ được thể hiện rõ trong file runner.cpp ở github.*

2.2.1 Quick sort

- Quick sort có hai hàm trong chương trình là QuickSort và Partition. QuickSort sẽ hoạt động bằng đệ quy, Partition là hàm phân hoạch - ở đây pivot được chọn ngẫu nhiên như giải thuật của Hoare[1].

2.2.2 Heap sort

- Heap sort có hai hàm trong chương trình là heapify và HeapSort. HeapSort sẽ hoạt động dựa trên hàm heapify – một hàm đệ quy.

- Lưu ý chỉ có ở phần code của HeapSort, các giải thuật khác trong bài sẽ call 2 parameters có dạng chỉ số cuối của dãy, i.e 0 và n – 1. Tuy nhiên, parameter được call trong HeapSort sẽ là n.

2.2.3 Merge sort

- Merge sort có hai hàm trong chương trình là merge và MergeSort. MergeSort sẽ hoạt động bằng đệ quy, merge là hàm gộp hai dãy con tăng lại với nhau.

2.2.4 Intro sort

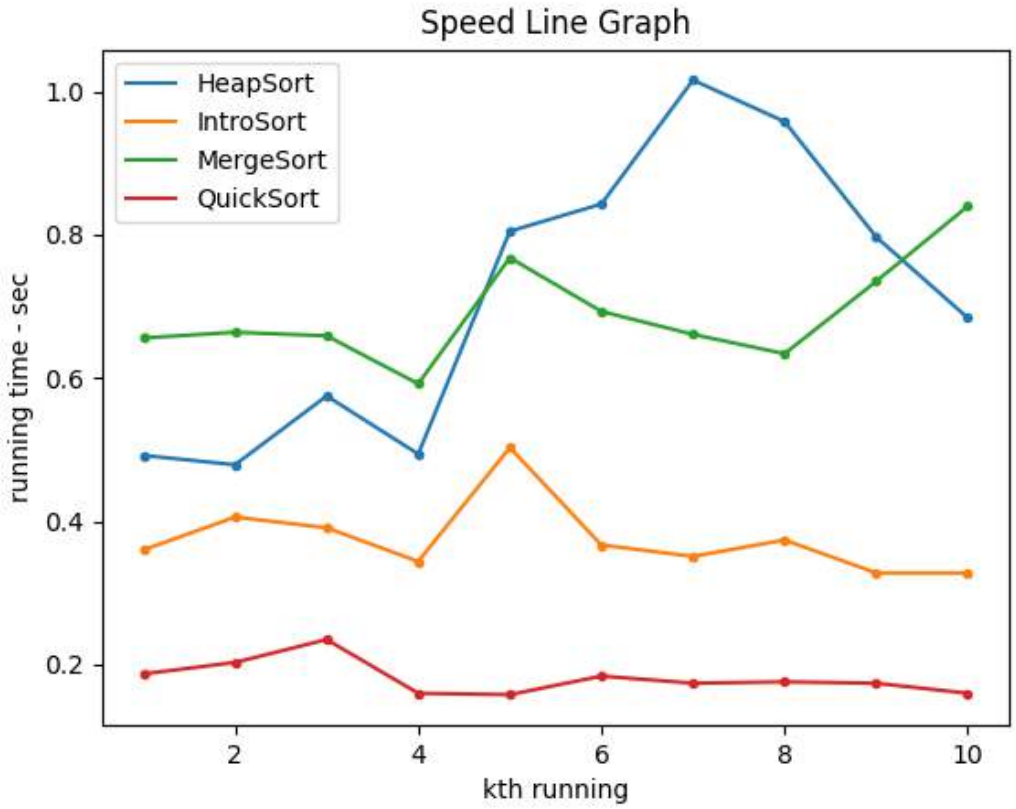
- Là hàm của thư viện C++. Ngon, xịn, bổ, rẻ! Cũng được dùng để kiểm tra tính đúng đắn khi viết các hàm sắp xếp khác trong bài báo cáo này.

3. Kết quả thu hoạch

3.1 Dạng bảng

	QuickSort	HeapSort	MergeSort	IntroSort	Order
Running 1	**0.187**	0.492	**0.656**	0.36	**Ascending**
Running 2	**0.203**	0.479	**0.664**	0.406	**Descending**
Running 3	**0.235**	0.575	**0.659**	0.391	Random
Running 4	**0.16**	0.494	**0.592**	0.344	Random
Running 5	**0.158**	**0.805**	0.768	0.503	Random
Running 6	**0.184**	**0.843**	0.693	0.367	Random
Running 7	**0.174**	**1.016**	0.661	0.351	Random
Running 8	**0.176**	**0.958**	0.634	0.374	Random
Running 9	**0.174**	**0.797**	0.735	0.328	Random
Running 10	**0.16**	0.684	**0.84**	0.328	Random

3.2 Đồ thị - Line Graph



- X-axis: các lần chạy – lần chạy thứ k.

- Y-axis: thời gian chạy – độ chia 0.2 giây.

4. Xử lý kết quả và báo cáo

4.1 Xử lý kết quả

- Đối với dạng bảng, số liệu được điền thủ công.

- Đối với dạng đồ thị, số liệu được đổ từ file output vào chương trình để vẽ nên đồ thị - visualize.py bằng việc sử dụng thư viện matplotlib[2].

4.2 Báo cáo

- Từ bảng và đồ thị, ta rút ra được một vài nhận xét như sau:

  + QuickSort chiếm thế thượng phong sau cả 10 bộ test, cho thấy mình xứng đáng với cái tên nằm trong top thuật toán khoa học trong thế kỉ XX[3].QuickSort cho thời gian chậm nhất ở bộ dữ liệu thứ ba, và cho hiệu quả tốt nhất ở bộ dữ liệu thứ 5. Độ biến thiên là không nhiều qua 10 bộ.

+ MergeSort cho thấy sự ổn định trong thuật toán – thời gian rải trong khoảng 0.6 – 0.7 giây, không có sự biến động quá lớn. Thể hiện đúng tính chất của một thuật toán có độ phức tạp O(nlogn) với mọi trường hợp dù là best case hay worst case.

+ HeapSort cho thấy kết quả ở mức chấp nhận được với tỉ lệ 5:5 – 5 tệ : 5 khá ổn. HeapSort tuy là thuật toán với độ xử lý cũng khá tốt. Tuy nhiên nếu HeapSort phải gặp những testcase có những số lớn nằm ở cuối dãy nhiều i.e dãy có vẻ "tăng" thì HeapSort sẽ tốn rất nhiều thời gian khi cứ phải lộn đi lộn lại.

+ IntroSort cho thấy mình cũng là một thuật toán rất nhanh khi chỉ để thua QuickSort ở bài báo cáo. Ngoài ra, IntroSort cũng cho được một hiệu quả khá ổn định xuyên suốt các lần chạy. IntroSort gặp đỉnh cực đại ở bộ data thứ 5 và cùng đạt cực tiểu ở hai bộ data cuối.

+ Thành tích tốt nhất thuộc về QuickSort khi cả 10 lần đều vượt đèn xanh thành công. Trong khi đó, HeapSort và MergeSort chia nhau thành tích cùng đứng ở cuối bảng xếp hạng.

5. Viết thêm – Ngoài báo cáo

- Đây là lần thứ 2 em viết bài báo cáo này. Qua lần làm bài này, em cảm thấy đã hiểu hơn về cách các ý tưởng này hoạt động trên lý thuyết – và trong thực tế có sự khác nhau như thế nào. Vì đôi khi khái niệm big O không thể giải thích một cách tường tận được như tự tay em thực hành.

- Nhắc về bản cũ hơn của bài báo cáo này, em không xoá mà vẫn để ở một repo khác tại đây[4]. Tại vì nhờ việc làm lại bài báo cáo này, mà em mới cải tiến được một số thứ trong cách trình bày code và cách bố trí. Hơn thế nữa, ở bản cũ HeapSort cho em một số liệu khá là nhanh nên em đã nghĩ Heap rất tốt. Sau bài tập này thì em mới hiểu câu không có thuật toán nào là tốt nhất mà chỉ có thuật toán phù hợp nhất. Đúng như vậy, nó sẽ có từng trường hợp mà thuật toán sẽ phát huy khả năng của nó nhiều hay ít.

- Em cảm ơn thầy vì đã cho re-submit và dời deadline để em có thời gian phát triển bài làm của mình được đầy đủ hơn.

6. Sources

[1] https://en.wikipedia.org/wiki/Tony_Hoare

[2] https://matplotlib.org

[3] https://www.andrew.cmu.edu/course/15-355/misc/Top%20Ten%20Algorithms.html

[4] https://github.com/whynotkimhari/report_of_searching_algorithms

- Ngoài ra, các dòng code trong bài cũng có sự tham khảo tại các video của 28tech:

MergeSort https://www.youtube.com/watch?v=hTHO1Mprj8g&t=1240s

HeapSort https://www.youtube.com/watch?v=XFI96Z7i3LE

QuickSort https://www.youtube.com/watch?v=eT9Epyf0XLM&t=1786s