PHẦN BÁO CÁO ĐỒ ÁN

1)Trình bày các thuật toán sắp xếp ở mục 2.1.1

- Selection Sort:

+ Ý tưởng sắp xếp: Là một trong những thuật toán dễ cái đặt nên ý tưởng khá dễ .Đầu tiên,tạo một mảng gồm những phần tử cho trướt và chưa được sắp xếp. Tiếp theo,tìm và chọn phần từ nhỏ nhất(lớn nhất) trong mảng đưa ra đầu mảng và đổi vị trí này với vị trí đầu mảng. Sau đó, ta duyệt từ vị trí kế đầu mảng cho tới hết mảng, tìm phần tử nào nhỏ nhất(lớn nhất) rồi đổi chỗ vị trí ấy cho vị trí kế đầu mảng.Khi mà các phần tử đã được sắp xếp thì ta duyệt từ vị trí kế đó cho tới cuối mảng .Qúa trình này kết thức khi các phần tử trong mảng đã được sắp xếp hoàn toàn.

+ Các bước thực hiện thuật toán: Theo em thì sẽ có 5 bước để thực hiện thuật toán này

- Bước 1: Cho min(Max) về vị trí đầu tiên trong mảng (vị trí 0)
- Bước 2: Tìm kiếm phần tử nhỏ nhất(Max) trong mảng
- Bước 3:Hoán đổi với giá trị vị trí min(Max)
- Bước 4: Tăng min(Max) để trỏ tới phần tử tiếp theo
- Bước 5: Lặp cho tới khi các phần tử đã được sắp xếp hoàn toàn

+ Đánh giá tính hiệu quả:
Tuy ít đổi chỗ các các phần tử nhưng khi với số lượng phần tử lớn và rất lớn thì thuật toán này khá lâu so với 4 thuật toán đang làm,thậm chí gấp n lần lận.Trong trường hợp dữ liệu khác nhau thì thuật toán này cũng có sự thay đổi thấy rõ rệt. Số lần hoán vị ở trường hợp tốt nhất là 0.Trường hợp xấu nhất là: $3n(n-1)/2$ .Với mỗi giá trị của i thu t toán thực hiện (n – i – 1) phép so sánh và vì i chạy từ 0 cho tới (n–2), thuật toán sẽ cần $O(n^2)$ phép so sánh(độ phức tạp).

-Heap Sort:

+ Ý tưởng sắp xếp: Để tạo được Heap Sort thì trướt tiên phải tạo cây Heap(CreatHeap) sao cho với mọi nút cha đều có giá trị lớn hơn nút con. Khi đó nút gốc là nút có giá trị lớn nhất.Sau đó tiến hành thủ tục hiệu chỉnh dãy thành Heap(Shift). Lặp đi lặp lại CreatHeap và Shift với nút cuối trong Heap , sẽ thu được danh sách đã theo thứ tự sắp xếp mà mình mong muốn

+ Các bước thực hiện thuật toán:

- Bước 1: Tạo cây ,xây dựng Heap.Sử dụng thao tác CreatHeap để tao cây Heap và Shift để chuyển đổi mảng bình thường thành Heap
- Bước 2: Sắp xếp
  + Hoán vị phần tử cuối cùng của Heap với phần tử đầu tiên của Heap
  + Loại bỏ phần tử cuối cùng
  + Thực hiện thao tác Shift để điều chỉnh phần tử đầu tiên

+ Đánh giá tính hiệu quả:
Là thuật toán được sử dụng phổ biến,khá phức tạp nhiều thao tác mới sắp xếp. Tuy nhiên thuật toán này vẫn nhanh hơn Selection Sort nhưng chậm hơn Merge Sort và Quick Sort .Ở mỗi trong trường hợp dữ liệu , thuật toán này chạy khá nhanh và cho dù số lượng các phần tử có lớn là bao thì cũng chênh lệch không nhiều so với sắp xếp chọn.Thuật toán Heap Sort trong trường hợp xấu nhất sẽ tốt hơn Quick Sort.Thời gian sắp xếp tức độ phức tạp là: $0(n\log_2 n)$ .khi mà mảng đã được sắp xếp 1 phần thì sẽ chạy nhanh hơn Quick Sort . Thuật toán này không sài đệ quy.

-Quick Sort:

+ Ý tưởng sắp xếp: Cho 1 mảng gồm các phần tử cho trướt ngẫu nhiên.Đầu tiên, chia mảng thành 2 phần gồm phần lớn và nhỏ.Tiếp đến, chọn 1 phần tử chốt(thường được gọi là pivot).Xét trong mảng xem thử những phần tử nào nhỏ hơn hoặc bằng phần tử chốt thì nằm bên trái tức đưa vào phần nhỏ, còn không thì nằm bên phải tức đưa vào phần lớn.Cuối cùng gọi đệ quy để sắp xếp 2 phần này tạo thành 1 mảng đã được sắp xếp hoàn toàn theo yêu cầu.

+ Các bước thực hiện thuật toán:

- Bước 1: Lấy phần tử chốt là phần tử ở cuối danh sách(hoặc giữa dãy hoặc theo yêu cầu bài cho)
- Bước 2: Chia mảng bởi sử dụng phần tử chốt
- Bước 3: Gọi đệ quy Quick Sort với mảng con bên trái
- Bước 4: Gọi đệ quy Quick Sort với mảng con bên trái.

+ Đánh giá tính hiệu quả:
Đúng với tên gọi, thuật toán này chạy khá nhanh trong các trường hợp dữ liệu nên khi các trường hợp dữ liệu khác nhau sẽ chênh lệch không lớn cho lắm.Và được sử dụng phổ biến cho nhiều ngôn ngữ bởi lẽ tính hiệu quả và sử dụng đệ quy nó nhanh , gọn và dễ hiểu , đơn giản cho người xem.

- Độ phức tạp thuật toán
  - Trường hợp tốt: O(nlog(n))
  - Trung bình: O(nlog(n))
  - Trường hợp xấu: O(n^2)
- Không gian bộ nhớ sử dụng: O(log(n))

Tuy nhiên, nếu không chia 2 phần tốt và không chọn phần tử mốc(pivot) tốt thì dễ dẫn tới tình trạng trường hợp xấu và thời gian chạy và độ phức tạp sẽ lâu.Thuật toán sử dụng đệ quy, không ổn định cho lắm nhưng vẫn hơn 2 thuật toán trên

-Merge Sort:

+ Ý tưởng sắp xếp: Đầu tiên, cho 1 mảng gồm các phần tử cho sẵn. Sau đó, chia đôi mảng hiện tại thành những mảng con nhỏ hơn qua quá trình chia đôi mảng lớn và chúng ta tiếp tục chia đôi các mảng con cho tới khi mảng con nhỏ nhất chỉ còn 1 phần tử. Kế tiếp, tiến hành so sánh 2 mảng con có cùng mảng cơ sở.Khi so sánh chúng sẽ vừa sắp xếp và vừa ghép 2 mảng con đó lại thành mảng cơ sở. Vẫn tiếp tục so sánh và ghép mảng con lại cho tới khi còn lại mảng duy nhất thì đó là mảng đã được sắp xếp hoàn toàn mà mình mong muốn.

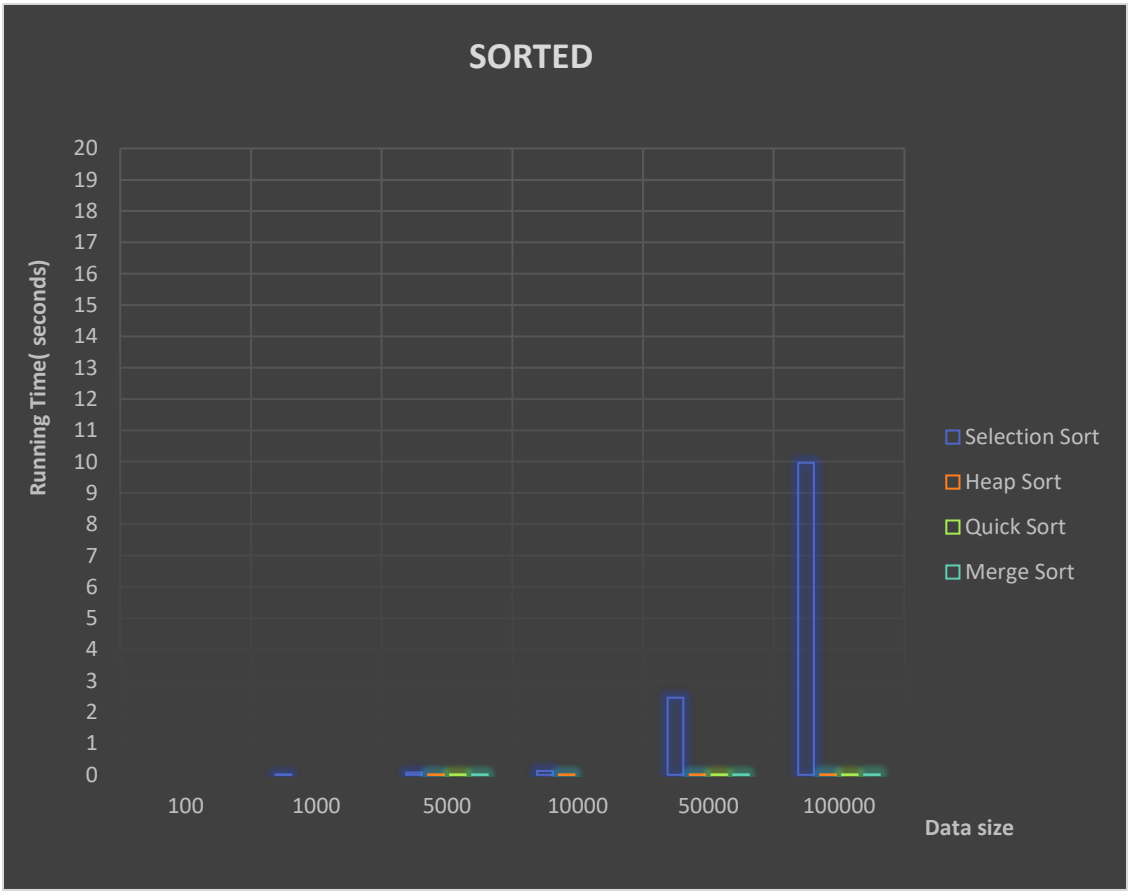
+ Các bước thực hiện thuật toán: Sẽ tiến hành theo 3 bước(trộn đệ quy)

- Nếu tồn tại 1 phần tử trong dãy thì xem như dãy này đã được sắp xếp. Sẽ trả về dãy hay 1 giá trị nào đó.
- Đệ quy dãy(mảng) thành 2 nữa cho tới khi không thể còn chia được .
- Kết hợp các dãy đã được sắp xếp thành 1 dãy mới.

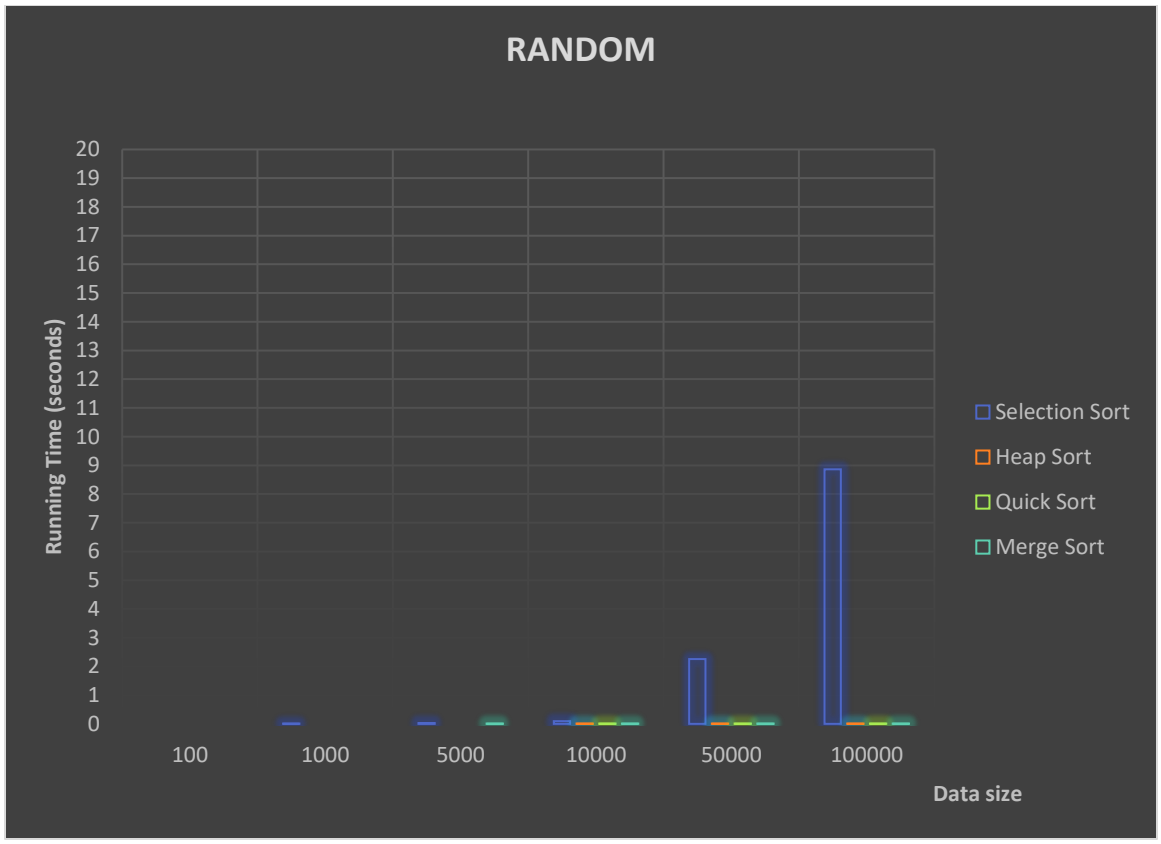
+ Đánh giá tính hiệu quả:
Sắp xếp nhanh hơn 2 thật toán đầu, tuy nhiên chậm so với Quick Sort.Trong 1 số trường hợp thì thuật toán này lại nhanh hơn so với sắp xếp nhanh .Là 1 thuật toán khó cài đặt và không nhận dạng được mảng đã sắp xếp.Tính hệu quả hay độ phức tạp được tính theo công thức: O(n log n) trong mọi trường hợp dữ liệu. Bởi vậy mà dữ liệu lớn và cần ít thao tác sẽ tối ưu hơn.

2) Trình bày kết quả thực nghiệm(Đồ thị):

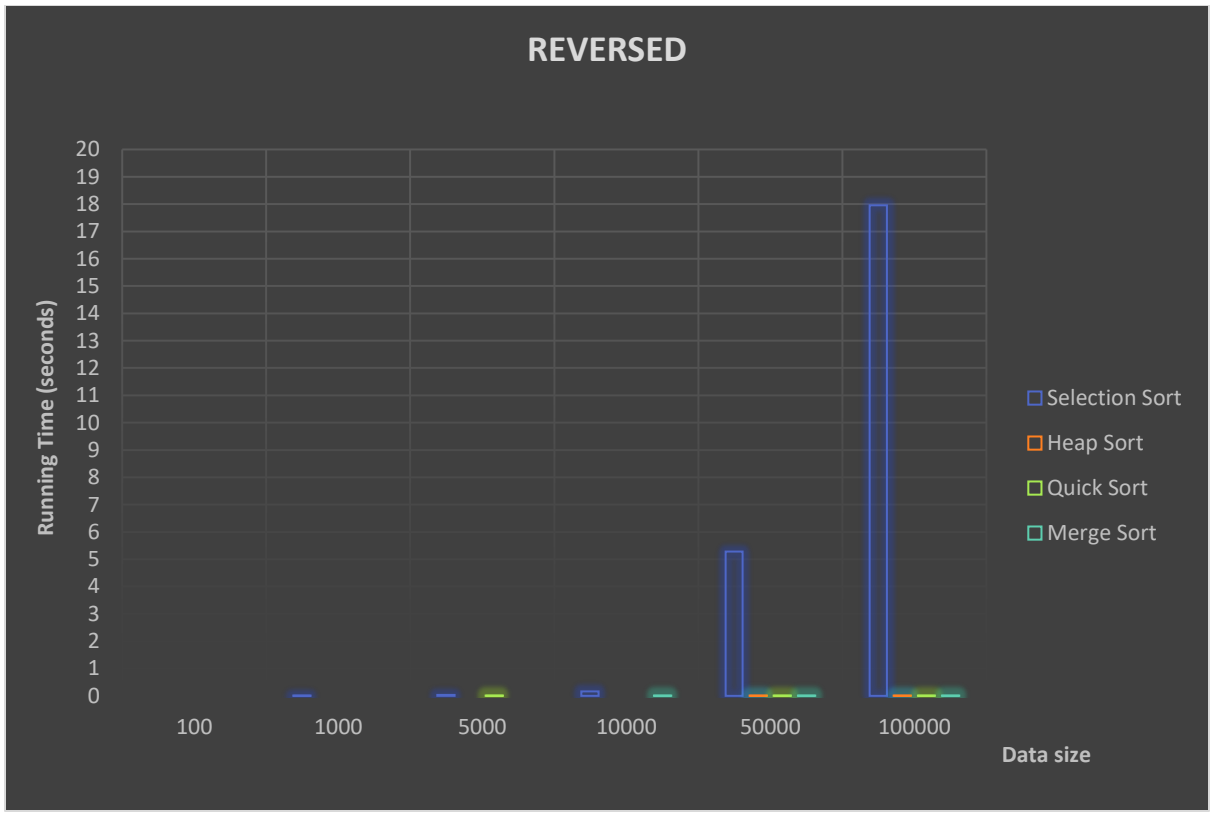
2.1) Dữ liệu có thứ tự



2.2) Dữ liệu phát sinh ngẫu nhiên



2.3) Dữ liệu có thứ tự ngược



3) Nhận xét:

-*Selection sort:* Trong 3 trường hợp kiểu dữ liệu , thuật toán này có thời gian chạy lâu nhất với các thuật toán còn lại.Số lượng(size) càng lớn thì thời gian chạy càng lâu.Cho nên , thuật toán này chỉ hiệu quả tốt khi chạy với data size vừa và nhỏ. Theo bảng thống kê và đồ thị thì trường hợp RANDOM là chạy ít thời gian nhất khi mà size= 100000 thì sẽ là 8,861 giây, còn SORTED thì size= 100000 là 9,967 giây,REVERSED là 17,958 giây.Từ các case, thì thời gian chạy thuật toán sắp xếp này tăng dần trông thấy rõ rệt vì số lần so sánh quá nhiều so với các thuật toán còn lại.Trong thực tế, người ta ít dùng thật toán này để ứng dụng.

-*Heap sort:* Là thuật toán cải tiến từ Seclection sort, nên sẽ tốt hơn và thời gian chạy ít hơn so với thuật toán trên.Trong 3 trường hợp dữ liệu thì trường hợp có thứ tự ngược lại chiếm ưu thế khi mà thời gian chạy ít hơn so với 2 trường hợp còn lại.Trong trường hợp REVERSED từ size=50000 và size=100000 thì mới có thời gian chạy và lần lượt là 0,004 giây và 0,007 giây.Trường hợp có thứ thự(SORTED) kết quả đo chạy lâu nhất và tốn thời gian hơn hẳn cái trường hợp kia.Nếu như 0,007 giây trong trường hợp REVERSED thì sẽ là 0,011 giây ở SORTEDvà 0,008 giây ở RANDOM khi size =100000.Heap sort là thuật toán khá sử dụng trong

cuộc sốn và học tập.Và nó được biểu diễn ở dạng nhị cây nhị phân.Trong mọi trường hợp theo bảng thống kê và đồ thị thì Heap sort sẽ chậm hơn Quick sort và ngang ngang với Merge sort.

-*Quick sort:* Sắp xếp tối ưu hơn 3 thuật toán còn lại.Theo bảng số liệu thì tính ổn định vừa phải, 3 trường hợp dữ liệu thì thời gian chạy khá ổn và bất chấp số lượng phần tử có lớn hay nhỏ là bao đi chăng nữa thì cũng chạy chênh lệch xấp xỉ nhau.Trường hợp dữ liệu có thứ tự khi size =100000 thì Quick sort chạy 0,005 giây so với 0,011 giây của Heap sort ,Merge sort và 9,967 giây của Selection sort. Tính nhanh, tiện lợi đã được thấy rõ.Tuy nhiên, không có thuật toán nào là hoàn hảo cả, bỡi vậy con người đã và đang phát minh ra nhiều thuật toán mà hơn hẵn gấp n lần so với các thuật toán thông thường.

-*Merge sort:* Thuật toan sử dụng chia để trị,chạy tốt và khá tốt khi mà size vừa và nhỏ, nhưng khi size lớn thì sẽ chạy lâu hơn nhiều so với trướt. Từ bảng thống kê cho thấy, Merge sort chạy khá bằng và chênh lệch ít hơn Heap sort khi mà size =50000 trở xuống. Còn size=100000 thì thời gian thực thi sẽ khác hẵn Heap sort. Nó chạy lâu hơn Heap sort, Quick sort nhưng tối ưu hơn Selection sort nhiều lần.Trong trường hợp dữ liệu có thứ tự thì sẽ chạy lâu nhất trong 3 trường hợp với 0,011 giây khi size=100000 so với RANDOM là 0,008 giây và REVERSED là 0,007 giây.

4) Đánh giá mức độ hoàn thành:

-Em hoàn thành 100% theo yêu cầu đồ án này, 9.75đ