Lớp: ANTN 2021

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Thời gian thực hiện: 11/03/2022 – 16/03/2022

Sinh viên thực hiện: Võ Nguyên Chương - 21520011

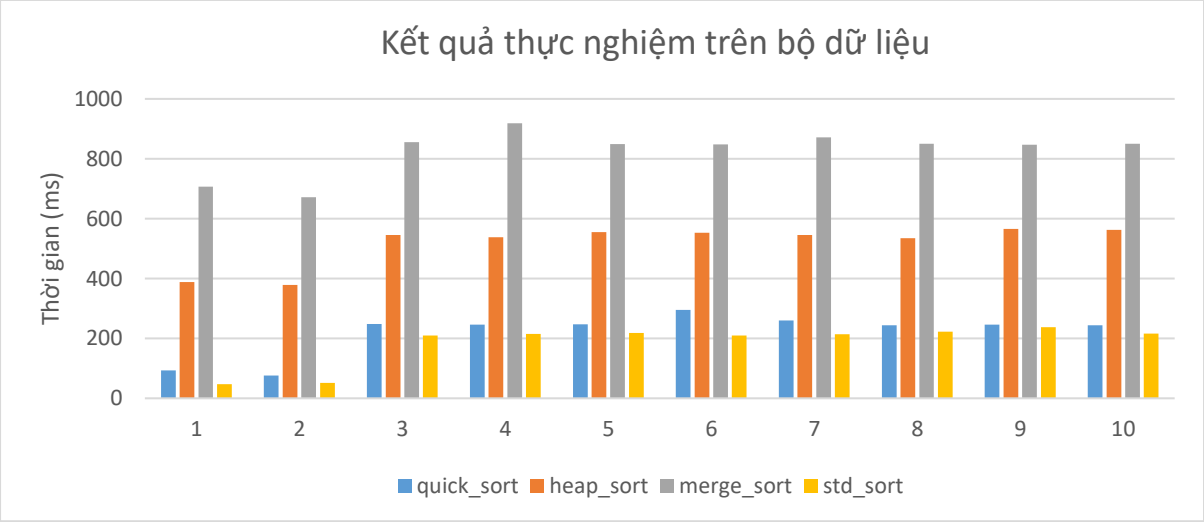
Nội dung báo cáo: Thực nghiệm các phương pháp sắp xếp

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện

Dữ liệu	Thời gian thực hiện (ms)			
	Quicksort	Heapsort	Mergesort	sort (C++)
1	93	388	707	47
2	76	379	672	51
3	248	545	856	210
4	246	538	919	215
5	247	555	849	218
6	295	553	848	210
7	260	545	871	214
8	244	535	850	223
9	246	566	847	237
10	244	563	850	216

2. Biểu đồ (cột) thời gian thực hiện



II. Kết luận:

1. Về mặt số liệu thì **std_sort** và **quick_sort** là hai thuật toán sắp xếp nhanh nhất (cả hai đều có thời gian thực thi trung bình xấp xỉ **N * log(N)**) trong đó **std_sort** nhanh hơn một chút so với **quick_sort.** Xếp sau đó là **heap_sort** và sau cùng là **merge_sort.**
2. **Std_sort** trong **C++** sử dụng thuật toán **intro_sort** là sử kết hợp chủ yếu của ba thuật toán **quick_sort**, **heap_sort** và **insertion_sort** để giảm thiểu tối đa

thời gian chạy bằng cách giới hạn số lần gọi đệ quy (khi số lần đệ quy vượt quá **log(N)** sẽ chuyển sang thuật toán khác là **heap_sort**)

Pesudocode – intro_sort:

```
procedure sort(A : array):
    let maxdepth = ⌈log2(length(A))⌋ × 2
    introsort(A, maxdepth)

procedure introsort(A, maxdepth):
    n ← length(A)
    if n ≤ 1:
        return
    else if maxdepth = 0:
        heapsort(A)
    else:
        p ← partition(A)
        introsort(A[0:p-1], maxdepth - 1)
        introsort(A[p+1:n], maxdepth - 1)
```

Do đó nó được xem như là thuật toán sắp xếp tốt nhất hiện nay và được sử dụng rộng rãi.

3. Về mặt lý thuyết thì **quick_sort** trong trường hợp tệ nhất có độ phức tạp là **O(N^2)** còn **heap_sort** và **merge_sort** thì đều có độ phức tạp là **N * log(N)** trong mọi trường hợp. Thế nhưng kết quả thực nghiệm lại cho thấy **quick_sort** chạy nhanh hơn **merge_sort** và **heap_sort**. Nguyên nhân là **quick_sort** tốn ít không gian bổ sung hơn (**merge_sort** cần thêm bộ nhớ để thực hiện thao tác trộn hai dãy đã được sắp xếp) và khả năng định vị bộ nhớ **cache** tốt hơn. Điều này ảnh hưởng tới thời gian thực thi chương trình nên trong nhiều trường hợp **quick_sort** nhanh hơn **heap_sort** và **heap_sort** sẽ nhanh hơn **merge_sort** (vì **heap_sort** không cần thêm bộ nhớ và gọi đệ quy nhiều lần như **merge_sort**)

III. 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/vnc1106/IT003.M21.ANTN---VN_Sorting

Thông tin chi tiết có mô tả trong file README.md trên github