HOMEWORK – TÌM HIỂU VỀ ĐỘ PHỨC TẠP THUẬT TOÁN HEAPSORT

1. Giải thuật Heapsort

Giải thuật Heapsort còn được gọi là giải thuật vun đống, có thể được xem như bản cải tiến của Selection Sort khi chia các phần tử thành 2 mảng con.

- 1 mảng các phần tử đã được sắp xếp.
- 1 mảng các phần tử chưa được sắp xếp.

Trong mảng chưa được sắp xếp, các phần tử lớn nhất sẽ được tách ra và đưa vào mảng đã được sắp xếp. Điều cải tiến ở Heapsort so với Selection Sort ở việc sử dụng cấu trúc dữ liệu heap thay vì tìm kiếm tuyến tính (linear-time search) như Selection sort để tìm ra phần tử lớn nhất.

Heapsort là thuật toán in-place, nghĩa là không cần thêm bất cứ cấu trúc dữ liệu phụ trợ trong quá trình chạy thuật toán. Tuy nhiên, giải thuật này không có độ ổn định (stability).

2. Các bước thực hiện

Giải thuật Heapsort được chia thành 2 giai đoạn.

Giai đoạn 1 :

Từ dãy dữ liệu input, sắp xếp chúng thành một heap (dạng cấu trúc cây nhị phân). Heap này có thể là Min-heap (nút gốc có giá trị bé nhất) hoặc Max-heap (nút gốc có giá trị lớn nhất), trong bài viết này, ta sẽ sử dụng Max-heap với một số yêu cầu thỏa mãn sau:

- Nút cha sẽ luôn lớn hơn tất cả các nút con, nút gốc của heap sẽ là phần tử lớn nhất.
- Heap được tạo thành phải là một cây nhị phân đầy đủ, tức ngoại trừ các nút lá, ở cùng một cấp độ các nút nhánh không được thiếu.

Giai đoạn 2 :

Giai đoạn này gồm các thao tác được lặp đi lặp lại cho đến khi mảng dữ liệu được toàn tất sắp xếp:

- Đưa phần tử lớn nhất của heap được tạo vào mảng kết quả, mảng này sẽ chứa các phần tử đã được sắp xếp.
- Sắp xếp lại heap sau khi loại bỏ nút gốc (có giá trị lớn nhất) để tìm phần tử có giá trị lớn nhất tiếp theo.
- Thực hiện lại thao tác 1 cho đến khi các phần tử của heap đều được đưa vào mảng kết quả.

Như thế, mảng kết quả sẽ chứa các phần tử được sắp xếp giảm dần.

3. Độ phức tạp thuật toán Heap Sort

Qua đó, ta có thể tính toán độ phức tạp như sau:

Độ phức tạp Heap Sort = Chi phí lặp qua từng phần tử của mảng

+Chi phí duyệt từng phần tử trên heap

∗Chi phí xóa và điều chỉnh heap

Nói cách khác, chúng ta có công thức: O(Heap Sort) = N + N · logN (2)

Theo quy tắc lấy max, ta có độ phức tạp của thuật toán là O(max(N, NlogN)), mà ta đã biết rằng tỉ suất tăng của NlogN lớn hơn so với N khi ta xét limn→+.

Do đó độ phức tạp trung bình của thuật toán Heap Sort là: O(N log N)

Với độ phức tạp trong trường hợp xấu nhất bằng O (N log N), giải thuật sắp xếp vung đống Heapsort vẫn thường được sử dụng do có tốc độ chạy nhanh và không quá phức tạp.

4. Source code python về thuật toán Heap Sort:

```
def heapify(arr, n, i):
    largest = i
```

```
        l = i << 1 + 1
        r = i << 1 + 2
        if l < n and arr[largest] < arr[l]:
                largest = l
        if r < n and arr[largest] < arr[r]:
                largest = r
        if largest != i:
                arr[i], arr[largest] = arr[largest], arr[i]
                heapify(arr, n, largest)
def heapSort(arr):
        n = len(arr)
        for i in range(n>>1 - 1, -1, -1):
                heapify(arr, n, i)
        for i in range(n-1, 0, -1):
                arr[i], arr[0] = arr[0], arr[i]
                heapify(arr, i, 0)
```