**<a>Phương pháp sort dãy số - Phần 1: Bubble sort, selection sort & insert sort</a>**

**<p><b>Bubble sort</b></p>**

<p>Bubble được hiểu là bong bóng dạng bọt, chúng có thể được tạo ra bằng cách dùng một loại que thổi kết hợp với hỗn hợp xà phòng và nước, đây là một trò chơi gắn liền với tuổi thơ của nhiều người. Khi bóng bóng được thổi ra từ que nó sẽ hình thành một chùm bong bóng nối đuôi nhau với các bong bóng sau nhỏ hơn bong bóng trước</p>

<p>Bubble sort là một kỹ thuật sắp xếp dãy số bằng cách đẩy dần dần số lớn hơn với số trước của dãy số ra sau, các số nhỏ hơn cứ như vậy sẽ nối đuôi theo sau, giống như việc thổi bong bóng ra từ que (bong bóng to sẽ được thổi ra trước tiếp đến là các bong bóng nhỏ nối theo sau). Như vậy có thể hiểu nguyên tắc của kỹ thuật này là tìm số lớn hơn trong dãy số và đưa nó ra sau</p>

<p>Mình sẽ lấy một ví dụ đơn giản để minh họa kỹ thuật Bubble sort là sắp xếp dãy số sau 10, 9, 2, 1 theo thứ tự tăng dần</p>

<p>Như đã trình bày nguyên tắc của Bubble sort là đưa số lớn hơn ra sau, có nghĩa là số nào lớn hơn sẽ được đổi vị trí với số nhỏ hơn. Để so sánh các giá trị trong dãy số cần lặp chúng qua từng thành phần của dãy số, lặp đến đâu sẽ so sánh đến đó, nếu thấy số đó lớn hơn số sau thì chuyển chúng lên trước. Lưu ý là nếu dãy số có n số thì chỉ cần kiểm tra đến số n-1. Lặp lại toàn bộ quá trình trên n-1 lần khi đó dãy số sẽ được hoàn toàn</p>

<div class="a">

<img src="pics/**bubblesort1.PNG** " alt="Trulli" width="500" height="400">

</div>

<p>Để thực hiện bubble sort cần cho chạy 2 vòng lặp i, j chồng lên nhau như hình trên. Với dãy số có 4 số thì function phải thực hiện đến 9 lần chạy, j sẽ lần lượt đi qua các giá trị 0, 1, 2 trong 3 lần</p>

<p>0 0 ( j < 3) **0 1 2**</p>

<p>1 0 (j < 3) **0 1 2**</p>

<p>2 0 (j < 3) **0 1 2**</p>

<div class="a">

<img src="pics/**bubblesort1time.PNG** " alt="Trulli" width="500" height="400">

</div>

<p>Nếu giả sử dãy số tăng lên 101 số thì function sẽ phải chạy 100^2 lần. Câu hỏi đặt ra là có cách nào giảm số lần chạy bằng cách bỏ đi nhưng lần chạy không cần thiết? </p>

<p>Nếu để ý sau lần sort đầu tiên thì số lớn nhất của dãy số đã được đẩy ra sau cùng, ở lần sort tiếp theo số này không cần quan tâm nữa vì nó đã được đặt đúng vị trí, lúc này chỉ cần làm việc với các số còn lại trong dãy số, cụ thể như sau</p>

<p>Lần sort 1: số 10 lớn nhất sẽ được đưa ra sau khi đó dãy số trở thành 9, 2, 1, 10</p>

<p>Lần sort 2: không quan tâm đến số 10 chỉ cần so sánh 3 số trước nó là 9, 2, 1. Khi đó số 9 lớn nhất sẽ được đưa ra sau trở thành 2, 1, 9</p>

<p>Lần sort 3: không quan tâm đến số 9 chỉ quan tâm đến 2 số trước nó là 2, 1 kết quả sort sẽ là 1, 2</p>

<p>Tổng hợp 3 lần sort trên thu được dãy số được sort hoàn toàn 1, 2, 9, 10</p>

<div class="a">

<img src="pics/**bubblesort2time.PNG** " alt="Trulli" width="500" height="400">

</div>

<p>Với cách tiếp cận như trên function chỉ cần thực hiện 6 lần chạy ít hơn 3 lần với cách chạy đầu tiên, cụ thể như sau</p>

<p>Ba lần chạy đầu tiên sẽ lặp qua 3 giá trị đầu tiên của dãy số có index 0 1 2 tương ứng j < 3</p>

<p>Hai lần chạy tiếp theo sẽ lặp qua 2 giá trị đầu tiên của dãy số có index 0 1 tương ứng j < 2</p>

<p>Lần chạy cuối cùng sẽ lặp qua giá trị đầu tiên của dãy số có index 0 tương ứng j < 1</p>

<p>Với logic như trên vòng lặp sẽ được thiết lập như sau</p>

<p>For (let i = arrayX.length; i > 1; i--) // i = 4, 3, 2</p>

<p>For (let j = 0; j < i -1; j++) j = 0,1,2; 0,1; 0</p>

<div class="a">

<img src="pics/**bubblesort2.PNG** " alt="Trulli" width="500" height="400">

</div>

<p>Với cách làm trên j sẽ chạy liên tục cho đến khi j < 1, vậy nếu giả sử sau lần chạy đầu tiên với i = 4 => j < 3 vòng j sẽ chạy 3 lần (0,1,2) thu được dãy số đã được sort hoàn toàn, khi đó việc chạy tiếp vòng i = 3, 2, 1 là không cần thiết, cần phải ngắt ngay sau khi chạy i = 4. Để làm được điều này cần thiết lập trạng thái noSwaps = true ngay trước vòng j, nếu khi chạy có sự hoán đổi vị trí con số thì noSwaps = false, sau khi chạy hết vòng j nếu noSwaps vẫn bằng true thì bẻ gãy vòng i và trả về dãy số được sort hoàn toàn</p>

**<p><b>**Selection sort**</b></p>**

<p>Trái ngược với Bubble sort, selection sort sẽ chọn giá trị nhỏ nhất đưa lên trước thay vì giá trị lớn nhất đưa ra sau trong phương pháp bubble sort. Quá trình trên sẽ lặp lại khi đi tới giá trị cuối cùng của dãy số</p>

<p>Ví dụ cho dãy số sau 9, 10, 3, 2, 1</p>

<p>Đầu tiên đi từ giá trị đầu tiên của dãy số là 9, 9 < 10, 9 > 3, 3 > 2, 2 > 1. Như vậy sau lần đầu tiên tìm ra số 1 là số nhỏ nhất, tiến hành thay đổi vị trị giữa số 9 và 1 khi đó dãy số trở thành 1, 10, 3, 2, 9</p>

<p>Ở lần sort 2 di chuyển index lên 1 là số 10. Lập lại quá trình như trên thu được số 2 là nhỏ nhất. tiến hành đổi vị trí số 2 với số 10, lúc này dãy số trở thành 1, 2, 3, 10, 9</p>

<p>Ở lần sort 3 số 3 ở vị trí xuất phát là số nhỏ nhất lên không cần phải thay đổi vị trí</p>

<p>Ở lần sort 4 số xuất phát là số 10, đổi vị trí 9 cho 10 khi đó dãy số được sort hoàn toàn 1, 2, 3, 9, 10</p>

<div class="a">

<img src="pics/**selectionsort1.PNG** " alt="Trulli" width="500" height="400">

</div>

<p>Một vấn đề phát sinh là trong trường hợp trên là ở lần sort 3 không cần thay đổi vị trí vì số xuất phát đã là số nhỏ nhất, cho nên đoạn code trên có thể nâng cấp bằng cách gán điều kiện nếu giá trị nhỏ nhất trùng với giá trị xuất phát thì không cần hoán đổi vị trí, như vậy cần thêm điều kiện if (minValueLocation != i) vào đoạn code trên</p>

<div class="a">

<img src="pics/**selectionsort2.PNG** " alt="Trulli" width="500" height="400">

</div>

<p>Một hạn chế của selection sort là phải đi hết qua dãy số ngay cả khi các số phía sau nó đã được sắp xếp. Đối với Bubble sort khi các phần tử sau đã được sắp xếp thì việc hoán đổi vị trí của số sẽ không xảy ra, khi đó vòng lặp i sẽ được bẻ gãy và quá trình sort sẽ dừng lại</p>

**<p><b>Insert sort</b></p>**

<p>Insert được hiểu là đặt thêm một vật gì đó vào vật đã có sẵn, chức insert bắt gặp rất nhiều trong các chương trình máy tính chẳng hạn như khi bạn muốn chèn hình, bảng, comment vào nội dung sẵn có</p>

<p>Một hạn chế của Bubble sort hay Selection sort là chỉ làm việc được trên dãy số cố định. Khi đối mặt với một dãy số kiểu động với các phần tử được thêm liên tục theo thời gian thì Bubble sort hay selection sort sẽ không phát huy tác dụng, chính điều này đã tạo điều kiện cho sự ra đời của insert sort. Insert sort sẽ đi từ đầu đến cuối dãy số, đi tới đâu nó sẽ sort với các số phía trước tới đó, khi đó việc thêm số vào phía sau dãy số hoàn toàn không ảnh hưởng đến quá trình sort. Một trường hợp trong thực tế cần sử dụng Insert sort là khi sever cần sort theo số Subcription đăng ký, người đăng ký sẽ được update liên tục theo từng thời điểm trong ngày, trường hợp này Insert sort sẽ phát huy tác dụng</p>

<p>Để minh họa cho kỹ thuật Insert sort mình sẽ lấy một dãy số gồm các số **2, 1, 9, 76, 4**</p>

<p>Bắt đầu tại vị trí X[1] 1 < 2 , đảo 1 ra trước thành 1, 2</p>

<p>Tiếp tục X[2] = 9, 9 > 2, suy ra 1, 2, 9</p>

<p>Tiếp tục X[3] = 76, 76 > 9, suy ra 1, 2, 9, 76</p>

<p>Tiếp tục X[4] = 4, 4 < 76 & 4 < 9 & 4 > 2, suy ra đặt 4 vào giữa 2 & 9, suy ra 1, 2, 4, 9, 76</p>

<p>Để tiến hành code cho dãy số trên, tiến hành sử dụng 2 vòng lặp, vòng thứ nhất xuất phát từ i = 1, vòng nest bên trong với j = i - 1 đi từ vị trí i quay về đầu là j--</p>

<p>Chi tiết cách thực hiện: giả sử i đã được lặp tới phần tử cuối X[i] = X[4] = 4 đầu tiên cần tiến hành lưu giá trị curentValue = 4 vào trong hàm, tiếp theo thiết vòng vòng lặp j bên trong theo vòng lặp i với j = i - 1, với i = 4 thì j = 4 - 1 = 3, j[3] = 76, 76 > 4 suy ra vị trí số 4 cần được hoán đổi với vị trí số 3 là X[4] = X[j+1] = X[j], tiếp tục cho vòng lặp j này lùi lại thành j = 3 - 1 = 2, ở vị trí số 2 là 9 , 9 > curentValue = 4, suy ra 9 phải được đẩy lên phía trước ở vị trí 3, tiếp tục j lùi lại ở vị trí 1 và 0 đều nhận giá trị nhỏ hơn currentValue = 4 lên nó sẽ không chạy vòng lặp. Như vậy vòng lặp j chỉ hoán đổi được đến giá trị j = 2 thì ngưng, tiến hành gán giá trị currentValue cho X[j+1], phải cộng j cho 1 vì ở cuối vòng lặp j đã trừ đi 1. Tiếp tục quay lại vòng i và thực hiện tiếp</p>

<div class="a">

<img src="pics/**insertsort.PNG** " alt="Trulli" width="500" height="400">

</div>

<p>Qua đoạn code trên có thể thấy một ưu điểm nữa của Insert sort là khi dãy số đã được sort gần hết đồng nghĩa với việc vòng lặp j sẽ chạy rất ít, khi đó thời gian thực thi của Insert sort sẽ rất tốt</p>

**<p><b>Tổng kết</b></p>**

<p>Ba cách sort trên đều có chung đặc điểm là mỗi lần sort chúng cần làm việc gần như trên toàn bộ dãy số thông qua việc so sánh một số trong dãy số với các số còn lại của dãy số nên độ phức tạp trong thời gian của chúng sẽ là O(n), khi dãy số quá lộn xộn thì độ phức tạp trong thời gian thực thi có thể lên tới O(n^2). Do đó nếu sort dãy số quá nhiều số thì sẽ tiêu tốn rất nhiều thời gian, lúc này đòi hỏi cần phải có phương pháp khác tiếp cận tốt hơn. Trong phần hai bài tiếp theo mình sẽ giới thiệu ba phướng pháp sort khác với hiệu quả thời gian thực thi tốt hơn 3 phương pháp sort trên</p>