**<a>Phương pháp sort dãy số - Phần 2: Radix sort, merge sort & quick sort</a>**

**<p><b>Radix sort</b></p>**

<p>Radix trong toán học có nghĩa là cơ số, từ radix khiến nhiều người liên tưởng đến việc sắp xếp theo các con số theo cơ số. Vậy phương pháp này hoạt động như thế nào?</p>

<p>Nếu để ý 1 con số sẽ thấy chúng có đơn vị hàng chục, trăm, nghìn, triệu… Khi giá trị số tăng lên thì các chữ số trong hàng đơn vị cũng tăng theo tương ứng. Như vậy có thể dễ dàng so sánh các con số và sắp xếp chúng theo thứ tự tăng dần bằng cách so sánh các con số lần lượt theo đơn vị chục, trăm, nghìn… Lấy một ví dụ đơn giản cho dãy số gồm 81,778,32,797</p>

<p>Đầu tiên sắp xếp theo đơn vị nhỏ nhất theo thứ tự tăng dần sẽ thu được 1, 2, 7, 8 tương ứng các số 81, 32, 797, 778</p>

<p>Tiếp theo so sánh ở đơn vị hàng chục thu được 3, 7, 8, 9 tương ứng các số 32, 778, 81, 797</p>

<p>Và cuối cùng so sánh theo đơn vị hàng trăm thu được 0, 0, 7, 7 tương ứng các số 32, 81, 778, 797</p>

<p>Qua ví dụ trên có thể thấy nguyên tắc then chốt trong việc sắp xếp dãy số theo cơ số là cần xác định được con số số lớn nhất trong dãy số gồm mấy chữ số, giả sử có 3 chữ số như vậy đơn vị cao nhất của dãy số là hàng trăm. Việc thứ hai là xác định được con số theo đơn vị hàng chục, trăm, nghìn… của con số, ví dụ số 7 với đơn vị nhỏ nhất (k=0) là 7, số 10 có đợn vị hàng chục (k=1) là 1, số 877 có đơn vị hàng trăm (k=2) là 8</p>

<p>Mình sẽ lấy một ví dụ phức tạp hơn để mô phỏng cách làm trên</p>

<p>Giả sử cho 1 dãy số gồm các số 1, 6, 99, 100, 31, 777, 32</p>

<p>Đầu tiên mình sẽ tạo ra các array arr1, arr2,…, arr9, với arr1 sẽ chứa các số có đơn vị hàng trăm, chục nghìn… là 1, tương tự cho các arr2…arr9. Ví dụ số 32 nếu xét theo đợn vị nhỏ nhất là 2 nó sẽ thuộc về arr2, nhưng nếu xét theo đơn vị hàng chục nó sẽ thuộc về arr3</p>

<p>Dãy số trên theo quan sát số lớn nhất gồm 3 chữ số 777 và số bé nhất gồm 1 chữ số là 1</p>

<p>Số lớn nhất gồm 3 số như vậy phương pháp này sẽ loop 3 lần với các k = 0 (hàng nhỏ nhất), 1 (hàng chục), 2 (hàng trăm) </p>

<p>Ở lần lặp đầu tiên k = 0 theo đơn vị nhỏ nhất, tiến hành đưa các số vào các arrX (arr1…arr9) tương ứng. Ví dụ số 31 có index[0] phía bên phải là 1 thì đưa số 31 vào arr1, như vậy ở lần chạy k = 0 đầu tiên cho kết quả</p>

<p>Arr0 = [100] </p>

<p>Arr1 = [1,31] </p>

<p>Arr2 = [32] </p>

<p>Arr3, Arr4, Arr5, Arr8 = null</p>

<p>Arr6 = [6] </p>

<p>Arr7 = [777] </p>

<p>Arr9 = [99] </p>

<p>Tiếp theo hợp các arrX này thành 1 arr mới theo thứ tự lần lượt từ arr0 đến arr9 là 100, 1, 31, 32, 6, 777, 99</p>

<p>Cho array mới gộp chạy lần 2 với k = 1 từ phải qua</p>

<p>Arr0 = [100, 1, 6] #1, 6 với đơn vị hàng chục (k = 1) là 0</p>

<p>Arr1, Arr2, Arr4, Arr5, Arr6, Arr8 = null</p>

<p>Arr3 = [31, 32] </p>

<p>Arr7 = [777] </p>

<p>Arr9 = [ 99 ] </p>

<p>Tiếp tục gộp thành array mới 100, 1, 6, 31, 32, 777, 99</p>

<p>Cho array mới gộp chạy lần cuối với k = 2 từ phải qua</p>

<p>arr0 = [1, 6, 31, 32, 99] </p>

<p>arr1 = [100] </p>

<p>arr2, arr3, arr4, arr5, arr6, arr8, arr9 = null</p>

<p>arr7 = [777] </p>

<p>Hợp các array trên thành array mới 1, 6, 31, 32, 99, 100, 777</p>

**<p><b>Tiến hành code</b></p>**

<p>Qua các phân tích trên có thể thấy, để sort được theo phương pháp cơ số đầu tiên cần xác định được con số ở hàng đơn vị nhỏ nhất cho đến hàng chục, trăm, nghìn… hàm getDigit(num) ở phần code dưới sẽ giúp xác định điều này. Một yếu tố nữa cần quan tâm là xác định xem số đó có bao nhiêu con số để từ đó có thiết lập giá trị k phù hợp, hàm digitCount(num) sẽ giúp tìm được giá trị k phù hợp</p>

<div class="a">

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

</div>

<p>Ngoài ra cần sử dụng thêm 1 hàm mostDigit(arr) để xác các con số trong array gồm bao nhiêu chữ số là tối đa ví dụ array 1, 6, 31, 32, 99, 100, 777 sẽ trả về 3 vì array đó có phần tử lớn nhất với 3 chữ số</p>

<p>Cuối cùng, tiến hành viết hàm radixSort(arrayX). Trong hàm này điểm cần lưu ý là digitBuckets sẽ tạo ra 1 array chứa 10 array con để chứa các con số lần lượt từ 0 đến 9, sau mỗi lần sắp xếp theo hàng đơn vị (tương ứng với k = 0 là đơn vị nhỏ nhất của dãy số, k = 1 đơn vị hàng chục…) tiếp tục tiến hành gộp các array này lại bằng câu lệnh [].concat(...digitBuckets). Lập lại thao tác trên cho các hàng đơn vị hàng chục, trăm nghìn… sẽ thu được 1 dãy số được sort hoàn toàn</p>

<div class="a">

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

</div>

**<p><b>Merge sort</b></p>**

<p>Trong excel, nhiều người đã từng sử dụng tính năng merge cell gộp các ô thành 1 ô duy nhất. Trong sort, merge sort được hiểu là sort bằng cách gộp nhiều array đã được sort lại với nhau. Để đơn giản mình sẽ lấy ví dụ là một dãy số cần được sort bao gồm các số 4, 500, 700, 51, 60, 89, đầu tiên cần chia dãy số này ra làm 2 với dãy 1 là array1 = 4, 500, 700 và dãy 2 là array2 = 51, 60, 89. Để merge sort cho array1 & array2 cần tiến hành so sánh từng phần tử trong mỗi array, phần tử nào nhỏ hơn sẽ được đưa vào dãy số sort trước, lập lại quá trình cho đến khi toàn bộ các số trong 2 array được đưa vào hết trong array sort, phần dưới sẽ minh họa cho kỹ thuật này</p>

<p>Đầu tiên so sánh 2 phần tử trong 2 array ở vị trí 0 là 4 & 51, 4 < 51 như vậy sẽ đưa 4 vào dãy số sort trước , sort = 4</p>

<p>Sau khi đưa 4 vào dãy số sort lúc này array1 chỉ còn [500,700], tiếp tục so sánh 500 và 51, 51 < 500 như vậy sẽ đưa số 51 vào dãy số sort, sort = 4, 51</p>

<p>Tiếp tục so sánh 60 < 500, như vậy 60 sẽ được đưa vào dãy số sort, sort = 4, 51, 60</p>

<p>Tiếp tục so sánh 89 < 500, như vậy 89 sẽ được đưa vào dãy số sort, sort = 4, 51, 60, 89</p>

<p>Lúc này array2 đã đi hết qua các phần tử như vậy chỉ cần thêm các phần tử còn lại trong array1 vào dãy số sort sẽ thu được kết quả sau cùng sort = 4, 51, 60, 89, 500, 700</p>

<p>Phương pháp này tỏ ra hiệu quả đối với hai dãy số đã được sort, trong thực tế dãy số cần sort luôn là 1 dãy số gồm rất nhiều số, nếu tiến hành chia dãy số này ra làm 2 thì 99,9% là 2 dãy số con này chưa được sort, khi đó việc áp dụng merge sort là không khả thi. Ví dụ cho arrayX = [9,7,11,6,31,28] nếu chia ra làm đôi sẽ được [9,7,11] và [6,31,28] rõ ràng là 2 array này chưa được sort nên việc áp dụng merge sort là không được</p>

<p>Nhưng nếu chia array đó thành các phần tử riêng lẻ thì sao? </p>

<p>Ví dụ [4, 60, 89, 51] chia ra làm các phần tử riêng lẻ [4], [60], [89], [51] rõ ràng khi đó dùng mergeSort là khả thi, đầu tiên merge sort bên trái [4, 60] và bên phải [51, 89], lúc này 2 array con đã được sort, tiến hành merge sort 2 array con này lại lần nữa sẽ thu được kết quả [4, 51, 60, 89] </p>

<p>Như vậy nguyên tắc cơ bản của phương pháp trên là chia array thành các phần tử riêng lẻ và tiến hành sort từ các phần tử riêng lẻ đó, cứ sau mỗi lần sort sẽ tiến hành gộp (merge) lại, sau cùng sẽ thu được một array đã được sort hoàn toàn</p>

<p>Mình sẽ lấy 1 ví dụ phức tạp hơn để mô phỏng kỹ thuật này</p>

<p>Cho arrayX = 1, 9, 7,14, 22, 51, 87, 6</p>

<p>B1: Chia array làm đôi L = 1, 9, 7, 14 & R = 22, 51, 87, 6</p>

<p>B2: Chia bên trái tiếp tục làm đôi L = 1, 9 & R = 7, 14</p>

<p>B3: Chia bên trái tiến tục làm đôi L = 1 R = 9</p>

<p>B4: Return merge thành 1, 9</p>

<p>B5: Chia bên phải làm đôi L = 7 & R = 14</p>

<p>B6: Return merge thành 7, 14</p>

<p>B7: MergeSort bên trái ban đầu thành 1, 7,9 , 14</p>

<p>B8: Tiếp tục chia bên phải làm đôi L = 22, 51 & R = 87, 6</p>

<p>B9: Làm tương tự B3, B4, B5, B6, B7 khi đó phía bên phải ban đầu sẽ được sort thành 6, 22, 51, 87</p>

<p>B10: Lúc này bên trái và bên phải của dãy số ban đầu đã được sort, tiến hành merge sort cho chúng lần nữa sẽ thu được dãy số được sort hoàn toàn</p>

**<p><b>Tiến hành code</b></p>**

<p>Điểm then chốt của phương pháp trên là cần viết được 1 hàm merge cho 2 dãy số đầu vào đã được sort, hàm này trả về dãy số được sort hoàn toàn kết hợp từ 2 dãy số đầu vào</p>

<div class="a">

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

</div>

<p>Cuối cùng sẽ viết hàm mergeSort, hàm này sẽ liên tục chia dãy số cần sort đầu vào thành 2 dãy số con bằng phương pháp đệ quy và chỉ dừng lại khi không thể chia thêm nữa đồng nghĩa là chia tới khi còn 1 phần tử, khi đó hàm merge sẽ sort từ phần tử riêng lẻ này tăng lên dần dần cho đến hết dãy số, lúc này kết quả thu được 1 dãy số được sort hoàn toàn</p>

<div class="a">

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

</div>

**<p><b>Quick sort</b></p>**

<p>Phương pháp này chọn số đầu tiên trong dãy số làm điểm cơ sở, từ điểm cơ sở này tiến hành so sánh các số tiếp theo trong dãy số, nếu số nào nhỏ hơn điểm cơ sở thì tiến hành đưa chúng lại sát gần điểm cơ sở, thao tác trên lặp lại cho đến hết dãy số, cuối cùng đẩy điểm cơ sở lên phía trên các điểm vừa được kéo lại gần. Sau khi điểm cơ sở đã dịch chuyển lên tiếp tục chọn điểm cơ sở kế bên điểm cơ sở tiếp theo. Quá trình sort sẽ kết thúc khi điểm cơ sở là điểm kề cuối của dãy số</p>

<p>Để đơn giản mình sẽ lấy một dãy số làm ví dụ 8, 6, 4, 9, 11, 3, 2</p>

<p>Đầu tiên chọn điểm cơ sở ở vị trí đầu tiên của dãy số là 8, các số còn lại 6, 4, 3, 2 sẽ nhỏ hơn hơn 8, kéo các số này lại gần 8 và đưa số 8 lên lên trước các số đó kết quả thu được là một dãy số 6, 4, 3, 2, 8, 9, 11</p>

<p>Tiếp tục di chuyển điểm cơ sở lên 1 đơn vị là số 9 – số này nằm ngay sau điểm cơ sở cũ là số 8, tiến hành so sánh 11 > 9 suy ra không cần đưa điểm cơ sở này lên trên. Như vậy sau lần sort 1 thu được kết quả là dãy số gồm 6, 4, 3, 2, 8, 9, 11</p>

<p>Tiếp tục lần sort lần 2 với điểm cơ sở là 6, quá trình sẽ lặp lại y hệt các bước như trên thu được kết quả là dãy số 4, 3, 2, 6, 8, 9, 11</p>

<p>Tiếp tục lần 3 với điểm cơ sở là 4 thu được kết quả 3, 2, 4, 6, 8, 9, 11</p>

<p>Tiếp tục lần 4 với điểm cơ sở là 3 thu được kết quả 2, 3, 4, 6, 8, 9, 11</p>

<p>Như vậy sau 4 lần thực hiện thu được dãy số được sort hoàn toàn 2, 3, 4, 6, 8, 9, 11</p>

<p>Dãy số này có 7 số cần chạy 4 lần sort, câu hỏi đặt ra cần chạy bao nhiêu lần sort để dãy số được sort hoàn toàn? Nếu giả sử dãy số lên tới hàng triệu số thì quá trình này tiêu tốn rất nhiều thời gian, cần phải có một cách tiếp cận khác hiệu quả hơn</p>

<p>Một ý tưởng được đưa ra để giải quyết bài toàn trên là kéo điểm cơ sở vào vị trí gần trung tâm dãy số, phần dưới sẽ mô tả các tiếp cận này</p>

<p>Vẫn quay lại dãy số trên 8, 6, 4, 9, 11, 3, 2</p>

<p>Ở lần đầu tiên chọn điểm cơ sở ở vị trí ban đầu là 8, sau lần sort đầu tiên điểm cơ sở này sẽ di chuyển vào trong tạo thành dãy số 6, 4, 3, 2, 8, 9, 11</p>

<p>Điểm cơ sở là số 8 đã được đưa vào gần vị trí trung tâm, lúc này số 8 là số ở giữa dãy số và nó không cần sort, công việc sort chỉ cần thực hiện cho phía bên trái và bên phải của nó</p>

**<p><i>Sort bên trái</i></p>**

<p>Tiến hành sort phía bên trái của điểm cơ sở 8 là dãy số 6, 4, 3, 2</p>

<p>B1: Đầu tiên tiến hành chọn điểm cơ sở là số đầu tiên của dãy số có giá trị 6, số 6 sẽ được đưa ra sau số 4,3,2 khi đó kết quả thu được sẽ là 4, 3, 2, 6, số 6 trở thành điểm cơ sở và nó không cần được sort, lúc này công việc chỉ cần làm là sort bên trái 4, 3, 2</p>

<p>B2: Quay lại thao tác như trên với điểm cơ sở ban đầu là 4, số 4 > 3 & 2, đưa nó ra sau sẽ thu được kết quả 3, 2, 4, lúc này số 4 đã được đưa vào trong, lúc này chỉ quan tâm đến phía bên trái và bên phải của số 4, bên phải không có chỉ còn có bên trái là 3, 2</p>

<p>B3: Tiếp tục sort [3,2] thu đuọc [2,3], điểm cơ sở lúc này là 3</p>

<p>B4: Tiếp tục sort [2] thu đuọc 2</p>

<p>Gộp 4 bước trên thu được kết quả 2, 3, 4, 6</p>

**<p><i>Sort bên phải</i></p>**

<p>B1: Bên phải [9,11] chọn điểm cơ sở là 9, 9 < 11 như vậy 9 sẽ được giữ nguyên, lúc này chỉ quan tâm đến bên phải của số 9 là 11, bên trái không có</p>

<p>B2: Tiếp tục bên phải là 11, kết quả sort sẽ ra là 11</p>

<p>Gộp kết quả B1 & B2 thu được [9,11] </p>

<p>Như vây sau khi sort bên trái và phải kết qua thu được sẽ là</p>

<p>Bên trái là 2, 3, 4, 6</p>

<p>Bên phải là [9,11] </p>

<p>Điểm cơ sở đầu tiên là 8</p>

<p>Gộp kết quả sort bên trái, bên phải và điểm cơ sở lại thu được dãy số được sort hoàn toàn 2, 3, 4, 6, 8, 9, 11</p>

<p>Kết luận: chìa khóa để sort dãy số theo phương pháp này là dựa vào điểm cơ sở, cứ sau mỗi lần xác định được điểm cơ sở thì điểm cơ sở này sẽ được giữ nguyên vị trí và không cần sort, lúc này công việc sort chỉ cần thực hiện ở phía bên trái và bên phải. Khi xác định được điểm cơ sở cũng đồng nghĩa với việc dãy số sẽ bớt đi 1 số không cần sort, quá trình này sẽ được thực hiện lặp lại cho đến khi dãy số chỉ còn một số, khi đó sẽ thu được dãy số được sort hoàn toàn. Việc xác định điểm cơ sở được lặp đi lặp lại nhiều lần có thể được thực hiện bằng phương pháp đệ quy, phần code dưới sẽ mô tả chi tiết điều này</p>

**<p><b>Tiến hành code</b></p>**

<p>Đầu tiên cần tiến hành viết hàm để xác định điểm cơ sở pivot, hàm này sẽ trả về điểm cơ sở pivot sau mỗi lần chạy</p>

<div class="a">

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

</div>

<p>Tiếp theo sử dụng hàm pivot để viết hàm quickSort, hàm quickSort sẽ được gọi liên tục bằng phương pháp đệ quy cho đến khi quá trình sắp xếp chỉ còn 1 số pivot duy nhất lúc này index bên trái sẽ bằng index bên phải là 0, khi đó hàm sẽ trả về dãy số đầu vào, lúc này toàn bộ dãy số đã được sort hoàn toàn</p>

<div class="a">

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

</div>