Binsort

Bởi:

unknown

Giải thuật

Nói chung các giải thuật đã trình bày ở trên đều có độ phức tạp là $O(n^2)$ hoặc O(nlogn). Tuy nhiên khi kiểu dữ liệu của trường khoá là một kiểu đặc biệt, việc sắp xếp có thể chỉ chiếm O(n) thời gian. Sau đây ta sẽ xét một số trường hợp.

Trường hợp đơn giản:

Giả sử ta phải sắp xếp một mảng A gồm n phần tử có khoá là các số nguyên có giá trị khác nhau và là các giá trị từ 1 đến n. Ta sử dụng B là một mảng cùng kiểu với A và phân phối vào phần tử b[j] một phần tử a[i] mà a[i].key = j. Khi đó mảng B lưu trữ kết quả đã được sắp xếp của mảng A.

Ví dụ 2-7: Sắp xếp mảng A gồm 10 phần tử có khoá là các số nguyên có giá trị là các số 4, 7, 1, 2, 5, 8, 10, 9, 6 và 3

Ta sử dụng mảng B có cùng kiểu với A và thực hiện việc phân phối a[1] vào b[4] vì a[1].key = 4, a[2] vào b[7] vì a[2].key = 7, a[3] vào b[1] vì a[3].key = 1,...

Hình sau minh họa cho việc phân phối các phần tử của mảng a vào mảng b.

Mảng a	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
Khóa	4	7	1	2	5	8	10	9	6	3
Khóa	1	2	3	4	5	6	7	8	9	10
Mảng b	b[1]	B[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]	b[10]

Hình 2-18: Phân phối các phần tử a[i] vào các bin b[j]

Để thực hiện việc phân phối này ta chỉ cần một lệnh lặp:

for i:=1 to n do b[a[i].key] := a[i]

Đây cũng là lệnh chính trong chương trình sắp xếp. Lệnh này lấy O(n) thời gian.

Các phần tử b[j] được gọi là các **bin** và phương pháp sắp xếp này được gọi là bin sort.

Trường hợp tổng quát

Là trường hợp có thể có nhiều phần tử có chung một giá trị khóa, chẳng hạn để sắp một mảng A có n phần tử mà các giá trị khóa của chúng là các số nguyên lấy giá trị trong khoảng 1..m với m <= n. Trong trường hợp này ta không thể sử dụng các phần tử của mảng B làm bin được vì nếu có hai phần tử của mảng A có cùng một khoá thì không thể lưu trữ trong cùng một bin.

Để giải quyết sự đụng độ này ta chuẩn bị một cấu trúc có m bin, mỗi bin có thể lưu trữ nhiều hơn một phần tử. Cụ thể là bin thứ j sẽ lưu các phần tử có khóa là j ($1 \leq j \leq m$) sau đó ta sẽ nối các bin lại với nhau để được một dãy các phần tử được sắp.

Cách tốt nhất là ta thiết kế mỗi bin là một danh sách liên kết của các phần tử mà mỗi phần tử có kiểu RecordType. Ta sẽ gọi kiểu của danh sách này là ListType.

Ta có thể tạo kiểu ListType bằng cách ghép RecordType với một con trỏ để trỏ tới phần tử kế tiếp.

Lấy B là một mảng kiểu Array[KeyType] of ListType. Như vậy B là mảng các bin, mỗi bin là một danh sách. B được đánh chỉ số bởi KeyType, như thế có ít nhất một bin cho mỗi giá trị khoá.

Ta vẫn sẽ phân phối phần tử a[i] vào bin b[j] nếu j = a[i].key. Dĩ nhiên mỗi bin b[j] có thể chứa nhiều phần tử của mảng A. Các phần tử mới sẽ được đưa vào cuối danh sách b[j].

Sau khi tất cả các phần tử của mảng A đã được phân phối vào trong các bin, công việc cuối cùng là ta phải nối các bin lại với nhau, ta sẽ được một danh sách có thứ tự. Ta sẽ dùng thủ tục concatenate(L1,L2) để nối hai danh sách L1, L2. Nó thay thế danh sách L1 bởi danh sách nối L1L2. Việc nối sẽ được thực hiện bằng cách gắn con trỏ của phần tử cuối cùng của L1 vào đầu của L2. Ta biết rằng để đến được phần tử cuối cùng của danh sách liên kết L1 ta phải duyệt qua tất cả các phần tử của nó. Để cho có hiệu quả, ta thêm một con trỏ nữa, trỏ đến phần tử cuối cùng của mỗi danh sách, điều này giúp ta đi thẳng tới phần tử cuối cùng mà không phải duyệt qua toàn bộ danh sách. Hình sau minh họa việc nối hai danh sách.

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED.

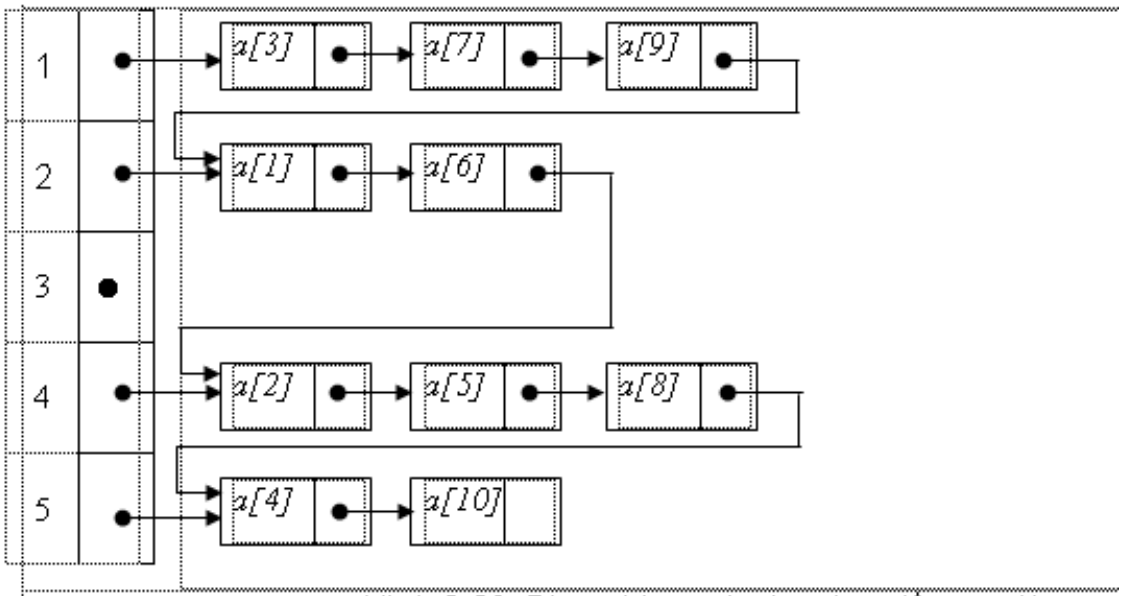
Hình 2-19: Nối các bin

Sau khi nối thì header và end của danh sách L2 không còn tác dụng nữa.

Ví dụ 2-8: Sắp xếp mảng A gồm 10 phần tử có khoá là các số nguyên có giá trị là các số 2, 4, 1, 5, 4, 2, 1, 4, 1, 5.

A	a[1]	a[2]	A[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
Khoá của A	2	4	1	5	4	2	1	4	1	5

Ta thấy các giá trị khoá nằm trong khoảng 1..5. Ta tổ chức một mảng B gồm 5 phần tử, mỗi phần tử là một con trỏ, trỏ đến một danh sách liên kết.



Hình 2-20: Binsort trong trường hợp tổng quát

Chương trình sử dụng cấu trúc danh sách liên kết làm các bin

```
VAR

a: ARRAY[1..n] OF RecordType;

b: ARRAY[keytype] OF ListType;

{Ta giả thiết keytype là kiểu miền con 1..m }

PROCEDURE BinSort;

VAR
```

```
Binsort

i:integer;

j: KeyType;

BEGIN

{1}FOR i:=1 TO n DO

Insert(A[i], END(B[A[i].key]), B[A[i}.key]);

{2}FOR j:= 2 TO m DO

Concatenate(B[1], B[j]);

END;
```

Phân tích Bin Sort

Bin sort lấy O(n) thời gian để sắp xếp mảng gồm n phần tử.

Trước hết thủ tục INSERT cần một thời gian O(1) để xen một phần tử vào trong danh sách. Do cách tổ chức danh sách có giữ con trỏ đến phần tử cuối cùng nên việc nối hai danh sách bằng thủ tục CONCATENATE cũng chỉ mất O(1) thời gian. Ta thấy vòng lặp {1} thực hiện n lần, mỗi lần tốn O(1) = 1 nên lấy O(n) đơn vị thời gian. Vòng lặp {2} thực hiện m-1 lần, mỗi lần O(1) nên tốn O(m) đơn vị thời gian. Hai lệnh {1} và {2} nối tiếp nhau nên thời gian thực hiện của BinSort là T(n) = O(max(n,m)) = O(n) vì $m \leq n$.

Sắp xếp tập giá trị có khoá lớn

Nếu m số các khoá không lớn hơn n số các phần tử cần sắp xếp, khi đó O(max(n,m)) thực sự là O(n). Nếu n > m thì T(n) là O(m) và đặc biệt khi $m = n^2$ thì T(n) là $O(n^2)$, như vậy Bin sort không tốt hơn các sắp xếp đơn giản khác.

Tuy nhiên trong một số trường hợp, ta vẫn có thể tổng quát hoá kĩ thuật bin sort để nó vẫn lấy O(n) thời gian.

Giả sử ta cần sắp xếp n phần tử có các giá trị khoá thuộc $0..n^2-1$. Nếu sử dụng phương pháp cũ, ta cần n^2 bin (từ bin 0 đến bin n^2-1) và do đó việc nối n^2 bin này tốn $O(n^2)$, nên bin sort lấy $O(n^2)$.

Để giải quyết vấn đề này, ta sẽ sử dụng n bin b[0], b[1],...b[n-1] và tiến hành việc sắp xếp trong hai kì.

Kì 1: Phân phối phần tử a[i] vào bin b[j] mà j = a[i].key MOD n.

Kì 2: Phân phối các phân tử trong danh sách kết quả của kỳ 1 vào các bin. Phần tử a[i] sẽ được phân phối vào bin b[j] mà j = a[i].key DIV n.

Chú ý rằng trong cả hai kỳ, ta xen các phần tử mới được phân phối vào cuối danh sách.

Ví dụ 2-9: Cần sắp xếp mảng gồm 10 phần tử có khoá là các số nguyên: 36, 9, 10, 25, 1, 8, 34, 16, 81 và 99.

Ta sử dụng 10 bin được đánh số từ 0 đến 9. Kì một ta phân phối phần tử a[i] vào bin có chỉ số a[i].key MOD 10. Nối các bin của kì một lại với nhau ta được danh sách có khóa là: 10, 1, 81, 34, 25, 36, 16, 8, 9, 99. Kì hai sử dụng kết quả của kì 1 để sắp tiếp. Phân phối phần tử a[i] vào bin có chỉ số a[i].key DIV 10. Nối các bin của kì hai lại với nhau ta được danh sách có thứ tự.

Kì một		Kì hai					
Bin				Bin			
0	10			0	1	8	9
1	1	81		1	10	16	
2				2	25		
3				3	34	36	
4	34			4			
5	25			5			
6	36	16		6			
7				7			
8	8			8	81		
9	9	99		9	99		

Hình 2-21: Sắp xếp theo hai kỳ

Theo sự phân tích giải thuật Bin Sort thì mỗi kì lấy O(n) thời gian, hai kì này nối tiếp nhau nên thời gian tổng cộng là O(n).

Chứng minh giải thuật đúng

Để thấy tính đúng đắn của giải thuật ta xem các các giá trị khóa nguyên từ 0 đến n^2-1 như các số có hai chữ số trong hệ đếm cơ số n. Xét hai số K = s.n + t (lấy K chia cho n được s , dư t) và L = u.n + v trong đó s, t, u, v là các số 0..n-1. Giả sử K < L, ta cần chứng minh rằng sau 2 kì sắp thì K phải đứng trước L.

Vì K < L nên s ≤ u. Ta có hai trường hợp là s < u và s = u.

Trường hợp 1: Nếu s < u thì K đứng trước L trong danh sách kết quả vì trong kì hai, K được sắp vào bin b[s] và L được sắp vào bin b[u] mà b[s] đứng trước b[u]. Chẳng hạn trong ví dụ trên, ta chọn K = 16 và L = 25. Ta có K = 1 x 10 + 6 và L = 2 x 10 + 5 (s = 1, t = 6, u = 2 và v = 5; s < u). Trong kì hai, K = 16 được sắp vào bin 1 và L = 25 được sắp vào bin 2 nên K = 16 đứng trước L = 25.

Trường hợp 2: Nếu s = u thì t < v (do K < L). Sau kì một thì K đứng trước L, vì K được sắp vào trong bin b[t] và L được sắp vào trong bin b[v]. Đến kì hai, mặc dù cả K và L đều được sắp vào trong bin b[s], nhưng K được xen vào trước L nên kết quả là K đứng trước L. Chẳng hạn trong ví dụ trên ta chọn K = 34 và L = 36. Ta có K = 3 x 10 + 4 và L = 3 x 10 + 6. Sau kì một thì K = 34 đứng trước L = 36 vì K được sắp vào bin 4 còn L được sắp vào bin 6. Trong kì hai, cả K và L đều được sắp vào bin 3, nhưng do K được xét trước nên K đứng trước L trong bin 3 và do đó K đứng trước L trong kết quả cuối cùng.

Chú ý:Từ chứng minh trên ta thấy để sắp các phần tử có khóa là các số nguyên (hệ đếm cơ số 10) từ 0 đến 99 ta dùng 10 bin có chỉ số từ 0 đến 9. Để sắp các phần tử có khóa là các số nguyên từ 0 đến 9999 ta dùng 100 bin có chỉ số từ 0 đến 99...

TỔNG KẾT CHƯƠNG 2

Các giải thuật sắp xếp đơn giản có giải thuật đơn giản nhưng kém hiệu quả về mặt thời gian. Tất cả các giải thuật sắp xếp đơn giản đều lấy $O(n^2)$ để sắp xếp n mẩu tin.

Các giải thuật QuickSort và HeapSort đều rất hiệu quả về mặt thời gian (độ phức tạp O(nlogn)), do đó chúng thường được sử dụng trong thực tế, nhất là QuickSort.

BinSort chỉ sử dụng được cho dữ liệu đặc biệt.

BÀI TẬP CHƯƠNG 2

Bài 1: Sắp xếp mảng gồm 12 phần tử có khóa là các số nguyên: 5, 15, 12, 2, 10, 12, 9, 1, 9, 3, 2, 3 bằng cách sử dụng:

1. Sắp xếp chọn.
2. Sắp xếp xen.
3. Sắp xếp nổi bọt.
4. QuickSort.
5. HeapSort (Sắp thứ tự giảm, sử dụng mô hình cây và sử dụng bảng).

Bài 2: Viết thủ tục sắp xếp trộn (xem giải thuật thô trong chương 1).

Bài 3: Viết lại hàm FindPivot để hàm trả về giá trị chốt và viết lại thủ tục QuickSort phù hợp với hàm FindPivot mới này.

Bài 4: Có một biến thể của QuickSort như sau: Chọn chốt là khóa của phần tử nhỏ nhất trong hai phần tử có khóa khác nhau đầu tiên. Mảng con bên trái gồm các phần tử có khóa nhỏ hơn hoặc bằng chốt, mảng con bên phải gồm các phần tử có khóa lớn hơn chốt. Hãy viết lại các thủ tục cần thiết cho biến thể này.

Bài 5: Một biến thể khác của QuickSort là chọn khóa của phần tử đầu tiên làm chốt. Hãy viết lại các thủ tục cần thiết cho biến thể này.

Bài 6: Hãy viết lại thủ tục PushDown trong HeapSort bằng giải thuật đệ quy.

Bài 7: Hãy viết lại thủ tục PushDown trong HeapSort để có thể sắp xếp theo thứ tự tăng.