Giới thiệu

Các thuật toán sắp xếp

Nội dung trình bày

- *Tiếp cận sắp xếp đơn giản*
  - *Sắp xếp chọn*
  - *Sắp xếp chèn*
  - *Sắp xếp nổi bọt*
- *Tiếp cận sắp xếp độ phức tạp O(nlog(n))*
  - *Sắp xếp theo phân đoạn (Quick sort)*
  - *Sắp xếp hòa nhập*
  - *Sắp xếp vung đống*
- Một số tiếp cận khác
  - Sắp xếp theo cơ số
  - Sắp xếp hòa nhập file lớn

Sắp xếp đếm - countingsort

- Bài toán
  - Có n phần tử cần sắp xếp là kiểu nguyên
  - Các giá trị của phần tử không lớn hơn giá trị k
- Có cách nào đó xác định được nhanh nhất phần tử x đầu vào sẽ xác định tại vị trí nào trong danh sách đầu ra
- Ví dụ:
  - Nếu kiểm tra được có 17 phần tử bé hơn phần tử x, vậy x sẽ được bắt đầu tại vị trí 18
  - Dùng một mảng trung gian để đếm vị trí xuất hiện của x trong dãy đầu ra

Sắp xếp đếm - countingsort (t)

- Đếm thông qua thống kê số lần xuất hiện của các phần tử
- Cộng dồn để xác định vị trí xuất hiện cuối của phần tử tiếp theo trong danh sách

- Phân phối các giá trị theo vị trí đã xác định

Sắp xếp đếm – countingsort (t)

- Thuật toán – countingsort (t)
- Input: dãy A[0..N-1] nguyên, miền giá trị [0..k]
- Output: dãy B[0..N] đã được sắp xếp

```
1. for(i=0->k)
        c[i]=0;
2. for(i=0->N-1)
        c[a[i]]++;
3. for(i=1->k)
        c[i]=c[i-1]+c[i];
```

Sắp xếp đếm – countingsort (t)

- Thuật toán – countingsort (t)

4. for(i=N-1->0)
    a. b[c[a[i]]-1]=a[i];
    b. c[a[i]]--;

Sắp xếp đếm – countingsort (t)

- Thử nghiệm

		0	1	2	3	4	5	6	7	8	9
i	a	3	1	7	8	2	6	9			
	c	0	0	0	0	0	0	0	0	0	0
	c	0	1	1	1	0	0	1	1	1	1
	c	0	1	2	3	3	3	4	5	6	7

Sắp xếp đếm – countingsort (t)

- Thử nghiệm

		0	1	2	3	4	5	6	7	8	9
i	a	3	1	7	8	2	6	9			
6	b							9			
	c	0	1	2	3	3	3	4	5	6	6
5	b				6			9			
	c	0	1	2	3	3	3	3	5	6	6
4	b		2		6			9			
	c	0	1	1	3	3	3	3	5	6	6
3	b		2		6		8	9			
	c	0	1	1	3	3	3	3	5	5	6
2	b		2		6	7	8	9			
	c	0	1	1	3	3	3	3	4	5	6
1	b	1	2		6	7	8	9			
	c	0	0	1	3	3	3	3	4	5	6
0	b	1	2	3	6	7	8	9			
	c	0	0	1	2	3	3	3	4	5	6

Sắp xếp đếm – countingsort (t)

- Đánh giá độ phức tạp
  - Số phép toán chỉ số: 2*k
  - Số phép toán gán: N
  - Độ phức tạp thuật toán: O(N+K)
  - Độ phức tạp bộ nhớ: thêm mảng b, và mảng c trung gian.

Sắp xếp cơ số - radixsort

- Ý tưởng thuật toán
  - Nếu xem các số nguyên là tập hợp các con số
  - Sắp xếp theo số bên trái cùng (most significant) thành các nhóm
  - Sau đó tiến hành sắp trong nội bộ nhóm với các chỉ số tiếp theo về bên phải
  - Lặp hết các vị trí được dãy đã sắp xếp
  - Ý tưởng này rất là hay nhưng sẽ khó khăn là có rất nhiều nhóm nhỏ rất khó để kiểm soát
  - Giải pháp: sắp xếp từ phải sang trái (least fignificant)

Sắp xếp cơ số - radixsort (t)

- Ý tưởng thuật toán
  - Dựa trên việc tách các số, có thể sử dụng thuật toán sắp xếp đếm (countingsort) để sắp xếp trên các chỉ số
  - Như thế hệ số k sẽ là 9 (0->9) cho mỗi lần thực hiện sắp xếp
  - Sẽ thực hiện trên d là số chữ số của các số tham gia sắp xếp

Sắp xếp cơ số - radixsort (t)

- Thuật toán radix
- Input: A[0..N-1] kiểu nguyên, d số chữ số lớn nhất
- Ouput: A[..N-1] đã sắp xếp

1. for(i=1->d)
   countingsortex(A[0..N-1],i)

Sắp xếp cơ số - radixsort (t)

- countingsortex(A[0..N-1],k)
  - Mở rộng của countingsort với việc phân phối dựa trên giá trị của các con số tại vị trí thứ k

Sắp xếp cơ số - radixsort (t)

- Thuật toán – countingsortex (t)
- Input: dãy A[0..N-1] nguyên, cột tìm chỉ số k
- Output: dãy B[0..N] đã được sắp xếp

1. for(i=0->9)
   c[i]=0;
2. for(i=0->N-1)
   c[a[i]Θk]++;
3. for(i=0->9)
   c[i]=c[i-1]+c[i];

Sắp xếp cơ số - radixsort (t)

- Thuật toán – countingsortex (t)

4. for(i=N-1->0)
    a. b[c[a[i]Θk]-1]=a[i];
    b. c[a[i]Θk]--;
5. for(i=0->N-1)
    a[i]=b[i];

// Θk: lấy giá trị tại cột k của số

Sắp xếp cơ số - radixsort (t)

- Thử nghiệm

	1	2	3	4
1423	2342	1423	3234	1423
4325	3532	3423	4325	2342
4329	3432	4325	5326	3234
2342	6342	5326	4329	3423
3423	1423	4329	2342	3432
3234	3423	3532	6342	3532
5433	5433	3432	6343	3538
3532	6343	5433	4354	4325
3538	3234	3234	1423	4329
3432	4354	3538	3423	4354
6342	4325	2342	3432	5326
6343	5326	6342	5433	5433
5326	3538	6343	3532	6342
4354	4329	4354	3538	6343

Sắp xếp cơ số - radixsort (t)

- Đánh giá độ phức tạp
  - Dựa trên phép toán của thuật toán countingsort nhân với d – số chữ số
  - Độ phức tạp thuật toán O(dn)
  - Với trường hợp d là bé, hữu hạn thì được coi là O(n)
  - Độ phức tạp bộ nhớ: Thêm mảng b trung gian.

Sắp xếp cơ số - radixsort (t)

- Phát triển
  - Có thể phát triển để sắp xếp với các khóa sắp xếp là chuỗi có độ dài xác định
  - COW, DOG, SEA, …

Sắp xếp ở bộ nhớ ngoài

- Bài toán
  - Sắp xếp với dữ liệu được lưu trên bộ nhớ ngoài (ổ đĩa)
  - Khối lượng dữ liệu lớn (không thể đồng thời đọc vào bộ nhớ được)
- Các đặc trưng:
  - Không thể đọc vào bộ nhớ máy tính đồng thời
  - Thời gian thực thao tác đọc ghi đĩa chậm
- Mục tiêu
  - Hạn chế đọc ghi đĩa
  - Tận dụng tối đa bộ nhớ

Sắp xếp ở bộ nhớ ngoài

- Giải pháp
  - Đọc tối đa dữ liệu lên bộ nhớ, sử dụng thuật toán sắp xếp hiệu quả trên bộ nhớ (heapsort, quicksort, radixsort)
  - Ghi dữ liệu đã được sắp xếp thành các file tạm (mỗi khối thành một file tạm)
  - Đọc một phần nhỏ các file đã sắp xếp, tiến hành sắp xếp trộn, và tiếp tục đọc các phần đến hết
  - Trong trường hợp số lượng khối (file) trung gian lớn các khối nhỏ đọc lên để so sánh trộn không hiệu quả thì sẽ chia thành nhiều nhóm và trộn trên mỗi nhóm trước, sau đó trộn lại

Sắp xếp ở bộ nhớ ngoài

- Ví dụ
  - Có 110 mb dữ liệu
  - Bộ nhớ có thể sử dụng là 10mb
- Thực hiện
  - Đọc mỗi khối 10mb tiến hành sắp xếp với thuật toán quicksort, và ghi thành các file t01 -> t11
  - Mỗi file t01->t11 đọc mỗi khối 10/11 mb, tiến hành sắp xếp trộn với 11 dòng (kiểm tra trên 11 dòng cùng trộn vào)
  - File kết quả là file đã trộn hết t01->t11

Sắp xếp ở bộ nhớ ngoài

- Ví dụ
  - Có 110000 mb dữ liệu
  - Bộ nhớ có thể sử dụng là 10mb
- Thực hiện
  - Đọc mỗi khối 10mb tiến hành sắp xếp với thuật toán quicksort, và ghi thành các file t00001 -> t11000
  - Do mỗi lần đọc bộ nhớ sẽ dựa trên khối 64kb là thuận tiện trong lúc 10mb/11000 ~ 1kb việc đọc như vậy là chưa tối ưu khả năng đọc dữ liệu

Sắp xếp ở bộ nhớ ngoài

- Vì thế có thể xây dựng 100 khối trung gian để từ t00001 ->t00100 thành t001, sẽ có t001 -> t110
- Sau đó tiến hành trộn t001->t110 thành file đầu ra

Bài tập trên lớp

- Triển khai thuật toán trộn trên bộ nhớ ngoài

Nội dung trình bày

- Sắp xếp đếm countingsort
- Sắp xếp theo cơ số radixsort
- Sắp xếp trên bộ nhớ ngoài

Bài tập

- Cài đặt thuật toán trên ngôn ngữ lập trình và chạy thử
- Thử nghiệm các thuật toán sắp xếp để đạt được dãy không tăng với các bộ dữ liệu sau
- 5342 5435 7634 7632 3432 3232 3433 4534
- 5342 5342 5342 5342 5342 5342 5342 5342