Các vấn đề nâng cao về sắp xếp

Created by anhtt-fit@mail.hut.edu.vn
Updated by huonglt-fit@mail.hut.edu.vn

Các ứng dụng

- Tổ chức thư viện MP3
- Hiển thị kết quả xếp hạng của Google
- Liệt kê các mục thông tin theo trật tự thời gian
- Tìm điểm ở giữa
- TÌm các cặp giống nhau
- Tìm kiếm nhị phân trong CSDL
- Tìm các thư trùng lặp trong mailing list.
- Nén dữ liệu
- Đồ họa máy tính
- Sinh tin học
- Quản lý chuỗi
- Cân bằng tải trong máy tính song song
- . . .

Các thuật toán sắp xếp

Sắp xếp bên trong (Internal sorts)
- Insertion sort, selection sort, bubblesort, shaker sort.
  (…chèn, lựa chọn, nổi bọt, sàng)
- Quicksort, mergesort, heapsort, samplesort, shellsort.
  (…nhanh, trộn, vun đống, dựa mẫu, vỏ sò)
- Solitaire sort, red-black sort, splaysort, Dobosiewicz sort, psort, ...

Sắp xếp ngoài (External sorts)
- Poly-phase mergesort, cascade-merge, oscillating sort.
  (…trộn nhiều đoạn, thác nước, dao động)

Sắp xếp dựa trên cơ số (Radix sorts)
- Distribution, MSD, LSD.
- 3-way radix quicksort.

Sắp xếp song song (Parallel sorts)
- Bitonic sort, Batcher even-odd sort.
- Smooth sort, cube sort, column sort.
- GPUsort.

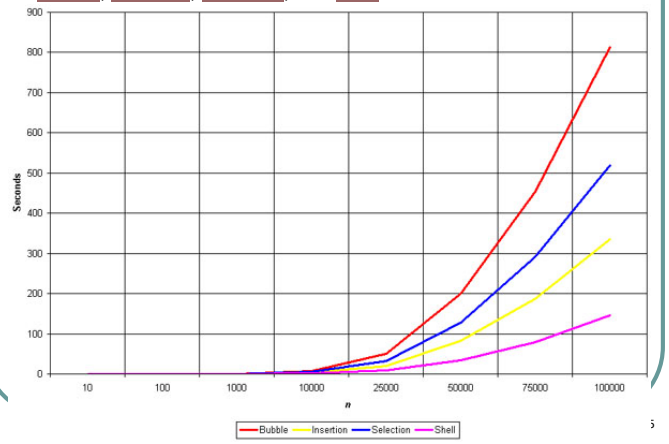
Tiêu chí lựa chọn thuật toán

- Ổn định?
- Nhiều khóa?
- Tất định (Deterministic)?
- Phù hợp với tất cả các dữ liệu (Keys all distinct)?
- Nhiều dạng khóa?
- Linked list hay arrays?
- Các bản ghi lớn hay nhỏ?
- Các file có trật tự ngẫu nhiên?
- Cần đảm bảo độ chính xác?

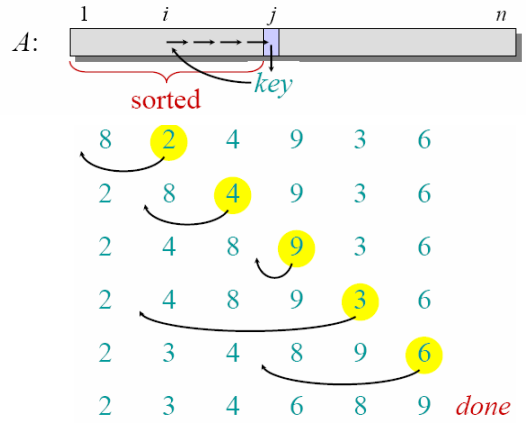
Mỗi thuật toán không thể phủ hết tất cả các tiêu chí

Thuật toán với độ phức tạp $O(n^2)$

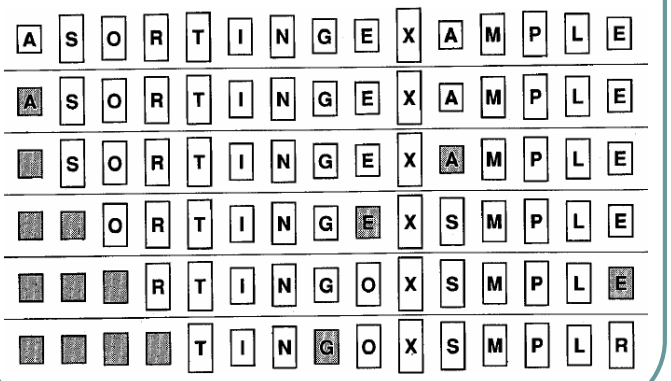
- bubble, insertion, selection, and shell sorts



Insertion sort

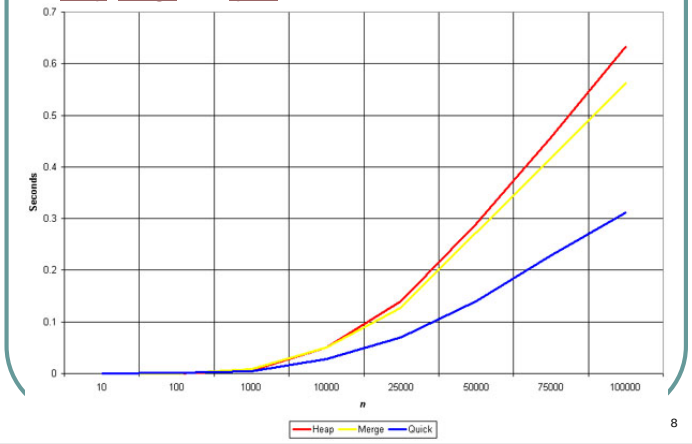


Selection sort

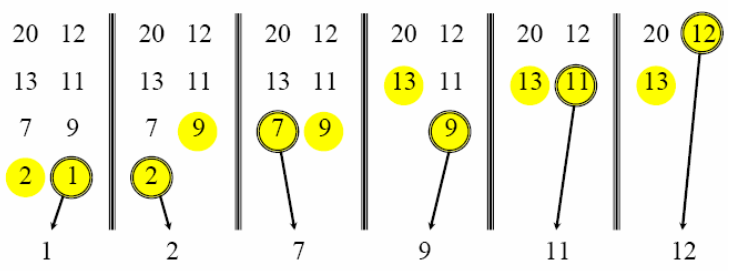


Thuật toán với độ phức tạp $O(n\log_2 n)$

- heap, merge, and quick sorts

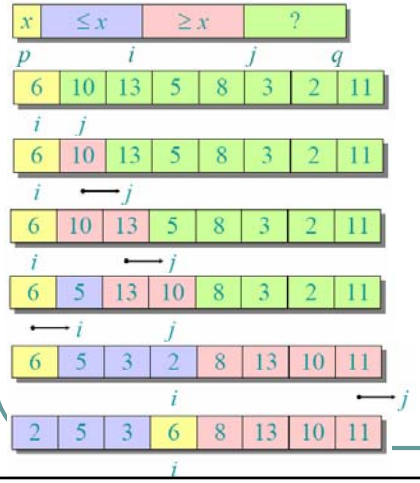


Merge sort



9

Quicksort



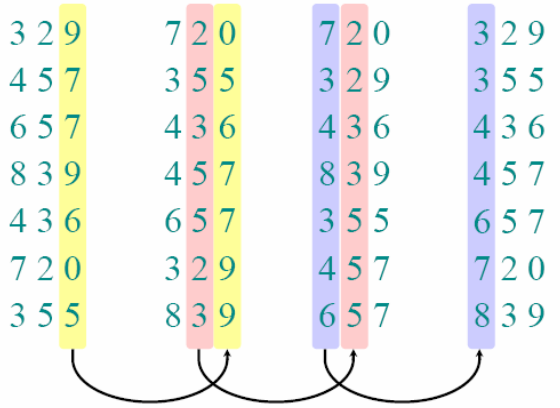
Quicksort(6,…,11)

Quicksort(2,…,5)
Quicksort(8,…,11)

Another demo:
demo-partition.ppt

10

Radix sort - Complexity O(n)



11

Case study 1

Bài toán:

- Xếp 1 file lớn chứa các bản ghi nhỏ có trật tự ngẫu nhiên.

Ví dụ:

- Xử lý các bản ghi giao dịch của 1 công ty điện thoại.

Lựa chọn thuật toán:

1. Quicksort: phù hợp
2. Insertion sort: No ($O(N^2)$ với các file có trật tự ngẫu nhiên)
3. Selection sort: No ($O(N^2)$)

12

Case study 2

Bài toán

- Xếp 1 file lớn gần như đã theo trật tự.

Ví dụ

- Sắp lại 1 CSDL lớn sau khi có một vài thay đổi nhỏ.

Lựa chọn thuật toán:

1. Quicksort: nhiều khả năng No (phép chèn đơn giản và nhanh hơn)
2. Insertion sort: YES (thời gian tuyến tính (O(N)) với phần lớn dữ liệu đã được sắp xếp)
3. Selection sort: No ($O(N^2)$)

13

Case study 3

Bài toán

- Xếp 1 file gồm nhiều bản ghi với các khóa nhỏ.

Ví dụ: tổ chức lại các file MP3

Lựa chọn thuật toán:

1. Mergesort: nhiều khả năng No (phép lựa chọn đơn giản và nhanh hơn)
2. Insertion sort: No (quá nhiều phép đổi chỗ)
3. Selection sort: YES (O(N) với một số giả thiết hợp lý)

Ví dụ: 5,000 bản ghi, mỗi bản ghi có 2 triệu byte với các khóa 100 byte

- Chi phí cho các phép so sánh: 100 x 5000^2 / 2 = 1.25 billion
- Chi phí đổi chỗ: 2,000,000 x 5,000 = 10 trillion
- Mergesort chậm hơn log (5000) lần

14

Các khóa lặp lại (Duplicate keys)

Mục đích của dạng sắp xếp này là nhóm các bản ghi với cùng khóa.

- Sắp dân số theo tuổi
- Tìm các điểm thẳng hàng
- Loại bỏ các thư trùng lặp trong mailing list.
- Sắp xếp các người xin việc theo trường đã theo học.

Đặc điểm của loại ứng dụng này:

- File lớn
- Có số lượng nhỏ các khóa

Mergesort với duplicate keys: ~O(N lg N)

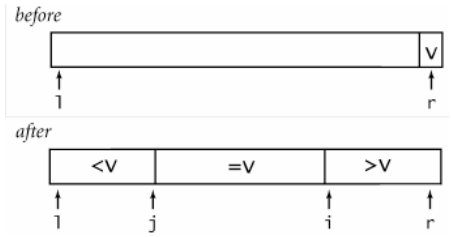
Quicksort với duplicate keys: $O(N^2)$ trừ khi việc phân chia dừng khi các khóa giống nhau.

15

Phân chia 3 phần (3-Way Partitioning)

Chia dữ liệu thành 3 phần:

- Các phần tử giữa i và j được đưa vào phần v
- Không có phần tử nào >v ở bên trái i
- Không có phần tử nào <v ở bên phải j



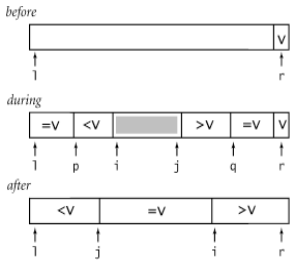
16

Cài đặt

Phân chia 3 phần (Bentley-McIlroy): Chia dữ liệu thành 4 phần:

- Không có phần tử nào >v ở bên trái i
- Không có phần tử nào <v ở bên phải j
- Các phần tử =v ở bên trái p
- Các phần tử =v ở bên phải q

Sau đó, trộn các phần tử =v vào phần trung tâm



17

Code

```
void sort(int a[], int l, int r) {
   if (r <= l) return;
   int i = l-1, j = r;
   int p = l-1, q = r;
   while(1)    {
        while (a[++i] < a[r]);
        while (a[r] < a[--j]) if (j == l) break;
        if (i >= j) break;
        exch(a, i, j);
        if (a[i]==a[r]) exch(a, ++p, i);
        if (a[j]==a[r]) exch(a, --q, j);
   }
   exch(a, i, r);
   j = i - 1;
   i = i + 1;
   for (int k = l ; k <= p; k++) exch(a, k, j--);
   for (int k = r-1; k >= q; k--) exch(a, k, i++);
   sort(a, l, j);
   sort(a, i, r);
}
```

18

Demo

- demo-partition3.ppt

19

Bài tập 1

- Viết 2 thuật toán kiểu quick sort theo:
  - 2-way partitioning
  - 3-way partitioning
- Tạp 2 mảng phân biệt cho 1 triệu số ngẫu nhiên có giá trị trong khoảng 1÷10.
- So sánh thời gian sắp xếp các số sử dụng 2 thuật toán trên.

20

Tổng quan thuật toán sắp xếp quick sort

```
void qsort(
     void *buf,
     size_t num,
     size_t size,
     int (*compare)(void const *, void
  const *)
);
```

- Hàm qsort() sắp *buf* (chứa các phần tử *num*, mỗi phần tử có kích thước là *size*).
- Hàm *compare* so sánh các phần tử trong buf. *compare* trả về giá trị <0 nếu tham số thứ nhất < tham số thứ 2, = 0 nếu tham số thứ nhất = tham số thứ 2, >0 nếu tham số thứ nhất > tham số thứ 2.

21

Ví dụ

```
int int_compare(void const* x, void const *y) {
  int m, n;
  m = *((int*)x);
  n = *((int*)y);
  if ( m == n ) return 0;
  return m > n ? 1: -1;
}
void main()
{
  int a[20], n;
  /* input an array of numbers */
  /* call qsort */
  qsort(a, n, sizeof(int), int_compare);
}
```

22

Hàm con trỏ

- Khai báo hàm con trỏ:
  - int (*pf) (int);
- Khai báo hàm
  - int f(int);
- Gán hàm cho hàm con trỏ:
  - pf = &f;
- Gọi hàm qua con trỏ:
  - ans = pf(5); // tương đương với ans = f(5)
- Trong qsort(), *compare* là hàm con trỏ để tham chiếu đến phép so sánh 2 phần tử

23

Bài tập 2

- Làm cách nào sử dụng qsort() để sắp 1 mảng theo trật tự giảm dần?
- Viết hàm quick sort (sử dụng thuật toán 3-way partitioning).
- Sử dụng hàm trên để sắp các kiểu dữ liệu khác nhau (số nguyên, các bản ghi số điện thoại, …)

24