TRƯỜNG ĐẠI HỌC AN GIANG
KHOA CÔNG NGHỆ THÔNG TIN

PHÂN TÍCH THIẾT KẾ GIẢI THUẬT

Giảng viên phụ trách:
NGUYỄN THÁI DƯ

Chương 3 SẮP XẾP NGOẠI

- Sắp thứ tự ngoại là sắp thứ tự trên tập tin
- Vì sao phải sắp xếp trên tập tin?
- Các phương pháp SX ngoại
  - **Phương pháp trộn Run**
  - **Phương pháp trộn tự nhiên**
  - **Phương pháp trộn đa lối cân bằng** (Balanced multiway merging)

Sắp xếp nội và sắp xếp ngoại

- **Sắp xếp nội:**
  - Sắp xếp dữ liệu trên RAM
  - Tốc độ truy xuất ngẫu nhiên cao
  - Dựa trên việc hoán đổi
  - Lượng dữ liệu cần sx nhỏ (vừa với RAM)
- **Sắp xếp ngoại:**
  - Sắp xếp dữ liệu trên đĩa (Disk)
  - Tốc độ tru xuất tuần tự cao (vẫn thấp hơn nhiều so với RAM)
  - Dựa trên thao tác trộn
  - Lượng dữ liệu cần sx lớn

1. Phương pháp trộn Run

- Run là một dãy liên tiếp các phần tử đã có thứ tự
- Ví dụ về Run: 2 4 7 12 50 40 60
- Chiều dài của Run chính là số phần tử trong Run
- Trong ví dụ trên có 2 run có độ dài lần lượt là 5 và 2
- Mỗi phần tử của dãy chính là 1 run có độ dài bằng 1

1. Phương pháp trộn Run

- Việc tạo ra một run mới từ 2 run ban đầu gọi là trộn run (merge).
- Run được tạo từ hai run ban đầu là một dãy các phần tử đã được sắp thứ tự.

1. Phương pháp trộn Run

Mô tả bài toán

- Dữ liệu vào: tập tin f0 cần sắp xếp
- Dữ liệu ra: tập tin f0 đã được sắp xếp
- f1, f2 là hai tập tin phụ dùng để sắp xếp

1. Phương pháp trộn Run

- Giả sử các phần tử trên f0 là:

24 12 67 33 58 42 11 34 29 31

- Khởi tạo f1, f2 rỗng

Bước 1:

- Thực hiện **phân bố** m=1 phần tử lần lượt từ f0 vào f1 và f2:

f1: 24 67 58 11 29
f2: 12 33 42 34 31

- **Trộn** f1, f2 thành f0:

f0: 12 24 33 67 42 58 11 34 29 31

1. Phương pháp trộn Run

Bước 2:

- **Phân bố** m=2*m=2 phần tử lần lượt từ f0 vào f1 và f2:

f0: 12 24 33 67 42 58 11 34 29 31
f1: 12 24 42 58 29 31
f2: 33 67 11 34

- **Trộn** f1, f2 thành f0:

f0: 12 24 33 67 11 34 42 58 29 31
f1: 12 24 42 58 29 31
f2: 33 67 11 34

Bước 3:

- Tương tự bước 2, **phân bố** m=2*m=4 phần tử lần lượt từ f0 vào f1 và f2, kết quả thu được như sau:

f0: 12 24 33 67 11 34 42 58 29 31

f1: 12 24 33 67 29 31

f2: 11 34 42 58

- **Trộn** f1, f2 thành f0:

f0: 11 12 24 33 34 42 58 67 29 31

Bước 4:

- **Phân bố** m=2*m=8 phần tử lần lượt từ f0 vào f1 và f2:

f0: 11 12 24 33 34 42 58 67 29 31

f1: 11 12 24 33 34 42 58 67

f2: 29 31

- **Trộn** f1, f2 thành f0:

f0: 11 12 24 29 31 33 34 42 58 67

Bước 5:

Lặp lại tương tự các bước trên, cho đến khi chiều dài m của run cần phân bổ lớn hơn chiều dài n của f0 thì dừng.

◆**Thuật toán tổng quát**

[**B1**] m = 1

[**B2**] Chia xoay vòng dữ liệu của file f0 cho f1 và f2, mỗi lần m phần tử, cho đến khi file f0 hết

[**B3**] Trộn từng cặp m phần tử của f1 và f2 tạo thành dãy mới 2m phần tử (được sắp) trên f0

[**B4**] m = 2*m

[**B5**] Nếu (m < N) thì Quay lại bước [B2]

Ngược lại **Kết thúc thuật toán**


```
m = 1
while (m < số phần tử của fo)
{
   Chia(Distribute) m phần tử của fo lần lượt
     cho f1, f2
   Trộn(Merge) f1, f2 lần lượt vào fo
   M = M * 2
}
```

2. Phương pháp trộn tự nhiên

- Trong phương pháp trộn đã trình bày ở trên, giải thuật chưa tận dụng được chiều dài cực đại của các run trước khi phân bổ; do vậy, việc tối ưu thuật toán chưa được tận dụng.

- Đặc điểm cơ bản của phương pháp trộn tự nhiên là tận dụng độ dài "tự nhiên" của các run ban đầu; nghĩa là, thực hiện việc trộn các run có độ dài cực đại với nhau cho đến khi dãy chỉ bao gồm một run -> dãy đã được sắp thứ tự.

2. Phương pháp trộn tự nhiên

Lặp Cho đến khi dãy cần sắp chỉ gồm duy nhất một run.

Phân bố:

Phân bố F0 vào F1 và F2 theo các run tự nhiên

Trộn:

Trộn các run của F1 và F2 vào F0

Quá trình này sẽ tiếp tục cho đến khi số run của F0 là 1 thì dừng

2. Phương pháp trộn tự nhiên

- **Giải thuật**

While (Số Run của F0 >1)

{

Phân bố F0 vào F1, F2 theo các Run tự nhiên.

Trộn các Run của F1, F2 vào F0.

}

- [Distribute] Chia xoay vòng dữ liệu của F0 cho F1 và F2, mỗi lần 1 run cho đến khi file F0 hết.
- [Merger] Trộn từng cặp run của F1 và F2 tạo thành run mới trên F0.

2. Phương pháp trộn tự nhiên

- F0: 1 2 9 8 7 6 5
- Bước 1:
  - F1: 1 2 9 7 5
  - F2: 8 6
  - F0: 1 2 8 9 6 7 5
- Bước 2:
  - F1: 1 2 8 9 5
  - F2: 6 7
  - F0: 1 2 6 7 8 9 5
- Bước 3:
  - F1: 1 2 6 7 8 9
  - F2: 5
  - F0: 1 2 5 6 7 8 9
- Bước 4: Dừng vì F0 chỉ có 1 Run

3. Phương pháp trộn đa lối cân bằng

- Thuật toán sắp xếp ngoài cần 2 giai đoạn: Phân phối và trộn.
  - Giai đoạn nào làm thay đổi thứ tự?
  - Chi phí cho giai đoạn phân phối?
- Rút ra kết luận:
  - Thay vì thực hiện 2 giai đoạn, ta chỉ cần thực hiện 01 giai đoạn trộn.
    - Tiết kiệm ½ chi phí Copy.
    - Cần số lượng file trung gian gấp đôi.

3. Phương pháp trộn đa lối cân bằng

- B1: Gọi tập nguồn S = {f1, f2, …, fn}
  Gọi tập đích D = {g1, g2, …, gn}

  Chia xoay vòng dữ liệu của file F0 cho các file thuộc tập nguồn, mỗi lần 1 Run cho tới khi F0 hết.
- B2: Trộn từng bộ Run của các file thuộc tập nguồn S, tạo thành Run mới, mỗi lần ghi lên các file thuộc tập đích D.
- B3: Nếu (số Run trên các file của D > 1) thì:
  - Hoán vị vai trò tập nguồn (S) và tập đích (D).
  - Quay lại B2

Ngược lại kết thúc thuật toán.

3. Phương pháp trộn đa lối cân bằng

- Ví dụ: **Cho dãy số sau**

3 5 2 7 12 8 4 15 20 1 2 8 23 7 21 27

- **Nhập :**

f0 :3 5 2 7 12 8 4 15 20 1 2 8 23 7 21 27

- **Xuất :**

f0: 1 2 2 3 4 5 7 7 8 8 12 15 20 21 23 27

3. Phương pháp trộn đa lối cân bằng

- **Nhập :**

f0 :3 5 2 7 12 8 4 15 20 1 2 8 23 7 21 27

Bước 0: đặt m = 3

Bước 1:

Phân phối các run luân phiên vào f[1], f[2], f[3]

f1: 3 5 4 15 20

f2: 2 7 12 1 2 8 23

f3: 8 7 21 27

3. Phương pháp trộn đa lối cân bằng

Bước 2:

-Trộn các run của f[1], f[2], f[3] và luân phiên phân phối vào các file g[1], g[2], g[3]

g1: 2 3 5 7 8 12

g2: 1 2 4 7 8 15 20 21 23 27

g3:

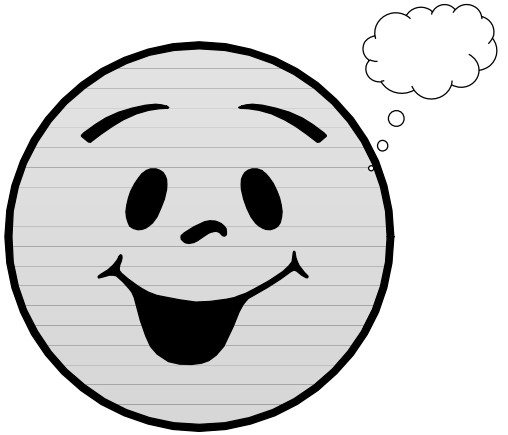
- Do số run sau khi trộn >1 nên tiếp tục trộn run từ g[1], g[2], g[3] vào ngược trở lại f[1], f[2], f[3]

f1: 1 2 2 3 4 5 7 7 8 8 12 15 20 21 23 27

f2:

f3:

- Do số run trộn = 1 nên kết thúc thuật toán



Cảm ơn !