CANTHO UNIVERSITY

TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
BỘ MÔN CÔNG NGHỆ THÔNG TIN

QUẢN TRỊ DỮ LIỆU - CT467

Chương 2: LƯU TRỮ VÀ CẤU TRÚC TẬP TIN

Biên soạn:

Ths. Nguyễn Thị Kim Yến

Ntkyen@ctu.edu.vn

File chỉ mục B^+ -cây - XÓA

Thao tác thực hiện:

1. Tìm nút lá chứa khóa cần xóa

2. Xóa khóa trong nút lá tìm được 1 trong các cách sau:

2.1 Trường hợp đơn giản, xóa khóa cây vẫn đúng cấu trúc

2.2 Trường hợp nút lá có ít hơn k khóa (trong đó, m = 2k+1), thực hiện tái cân bằng cây:

(a) Nếu 1 nút kề có nhiều hơn k khóa, chuyển một khóa sang

(b) Ngược lại, thực hiện gộp lại với một nút kề (cha sẽ giảm đi một nút con)

CANTHO UNIVERSITY

File chỉ mục B^+ -cây - XÓA

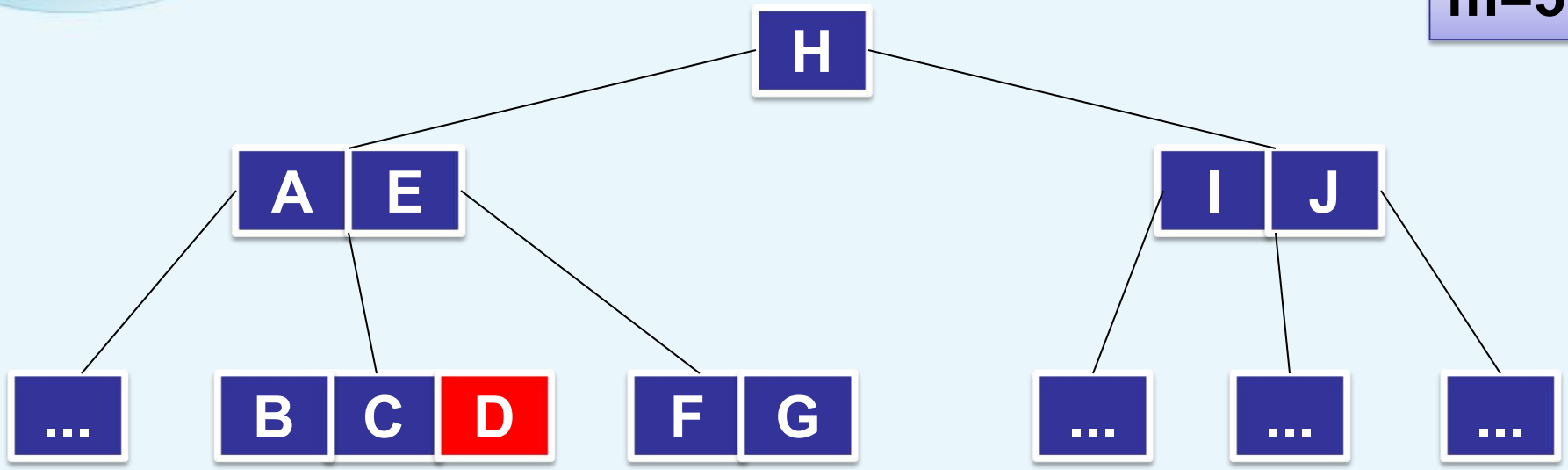
Thao tác thực hiện (tiếp theo):

2.3 Trường hợp nút trung gian có ít hơn k khóa, lặp lại thao tác tái cân bằng cây như ở bước 2.1 và 2.2

3. Nếu thực hiện gộp đến nút gốc và nút gốc chỉ còn 1 nút con, thì cho nút con làm nút gốc mới

2.1 Cây không thay đổi cấu trúc: Xóa khóa D

m=5



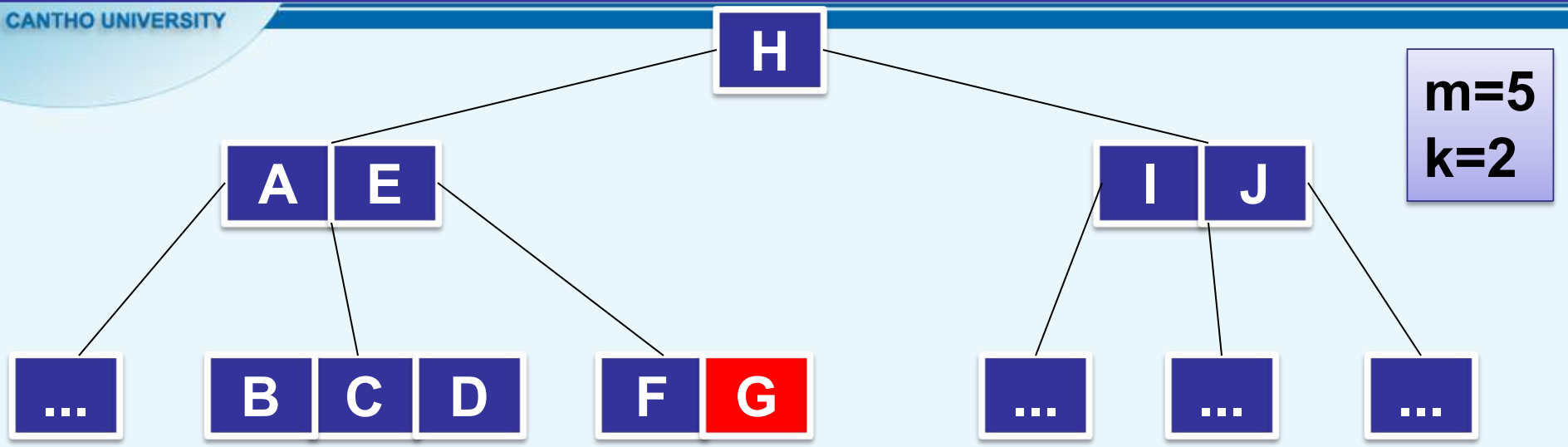
m = 2k+1 ➔ 5=2k+1
➔ 5 -1= 2k ➔ k=2

Số phần tử tối thiểu: (m-1)/2= 2

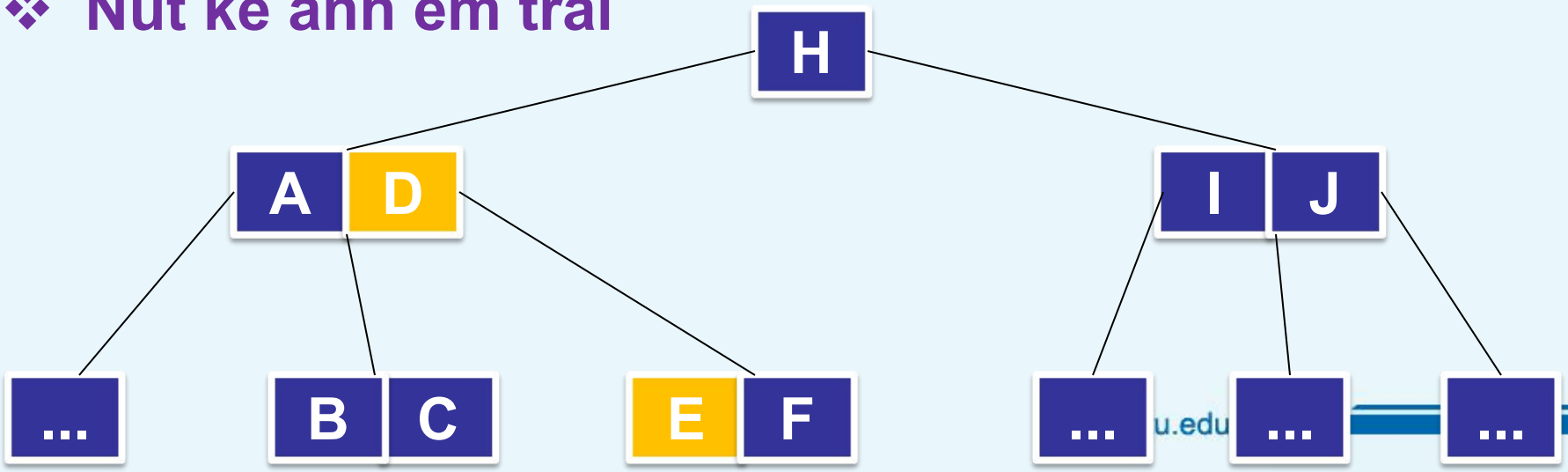
Số phần tử tối đa: m-1= 4

2.2 Nút lá có ít hơn k khóa thực hiện tái cân bằng cây

(a) Nút kề có nhiều hơn k khóa, chuyển 1 khóa sang

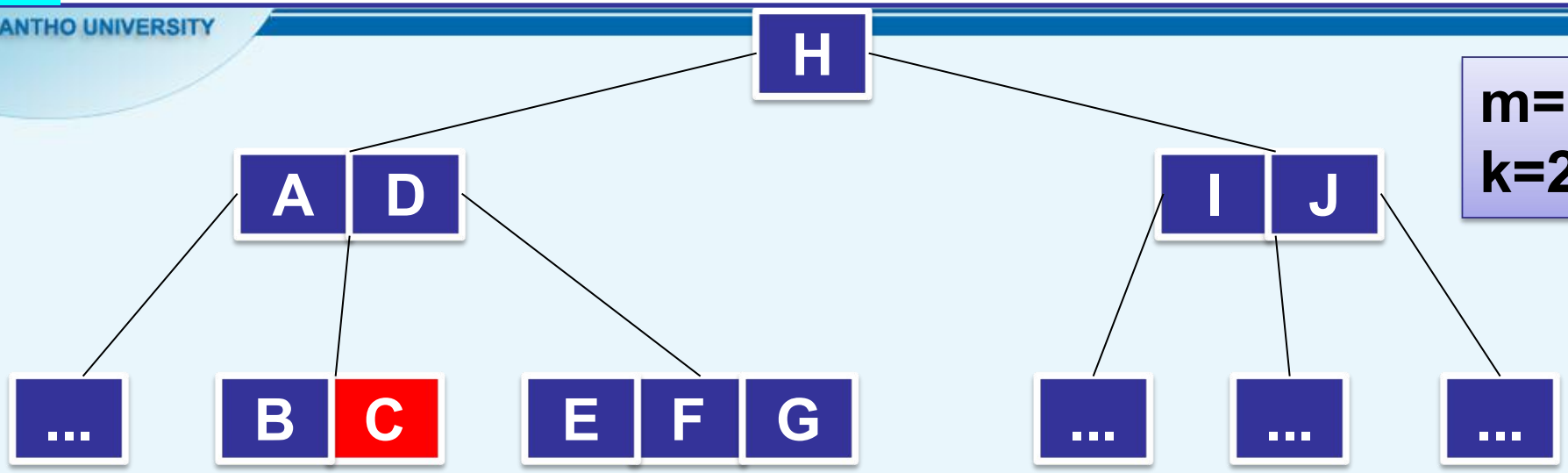


- Nút kề anh em trái

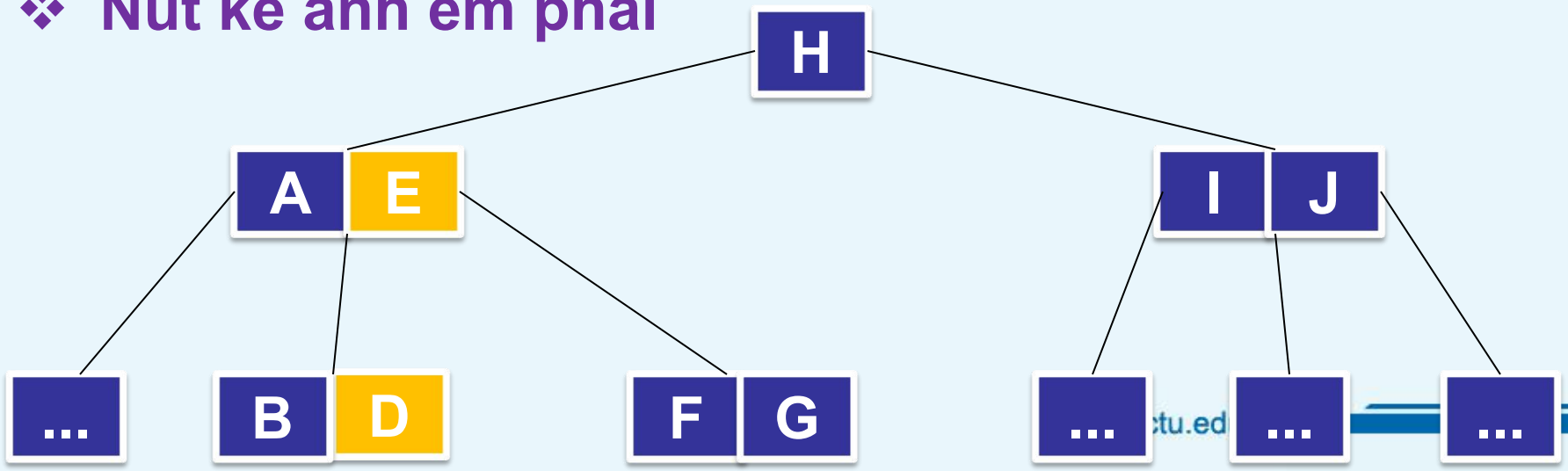


2.2 Nút lá có ít hơn k khóa thực hiện tái cân bằng cây

(a) Nút kề có nhiều hơn k khóa, chuyển 1 khóa sang

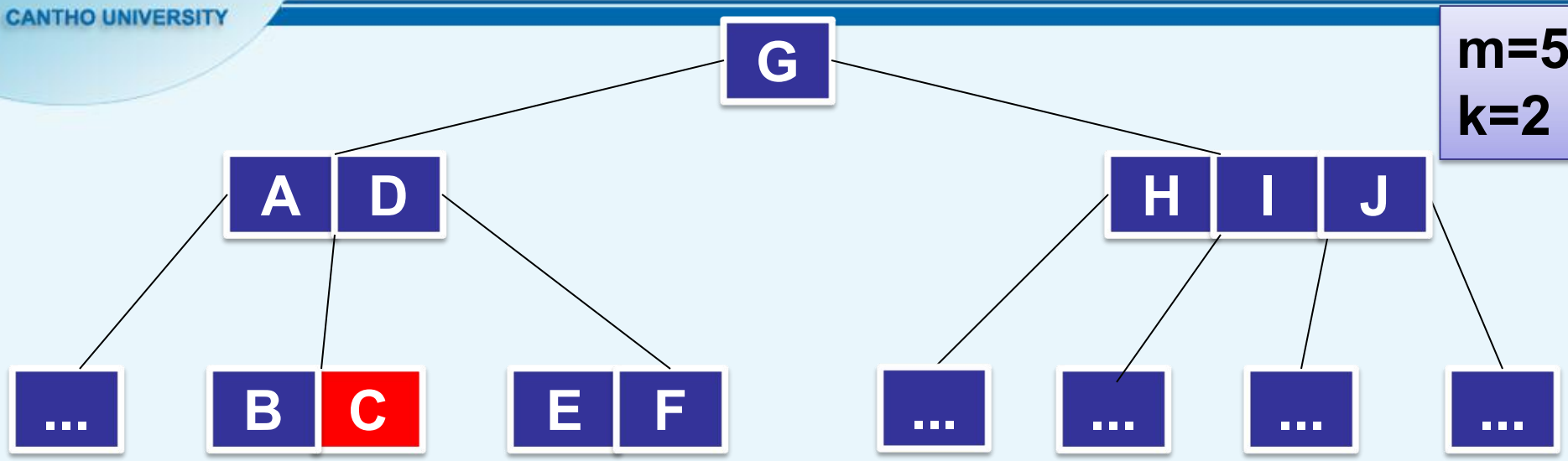


❖ Nút kề anh em phải

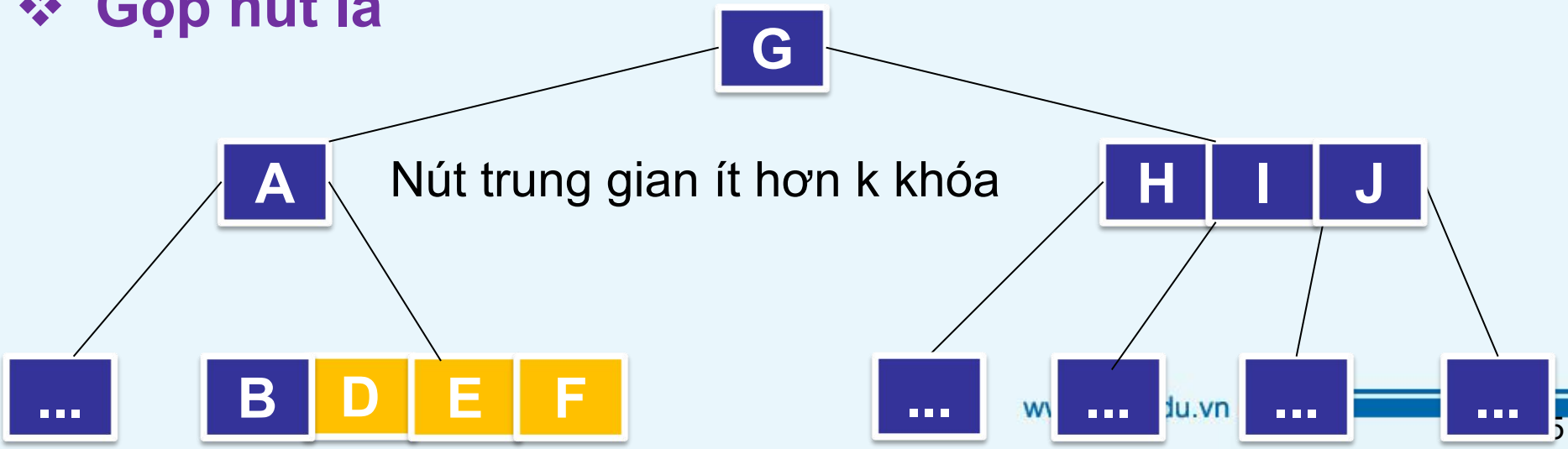


2.2 Nút lá có ít hơn k khóa thực hiện tái cân bằng cây

(b) Gộp lại với một nút kề (cha sẽ giảm đi một nút con)



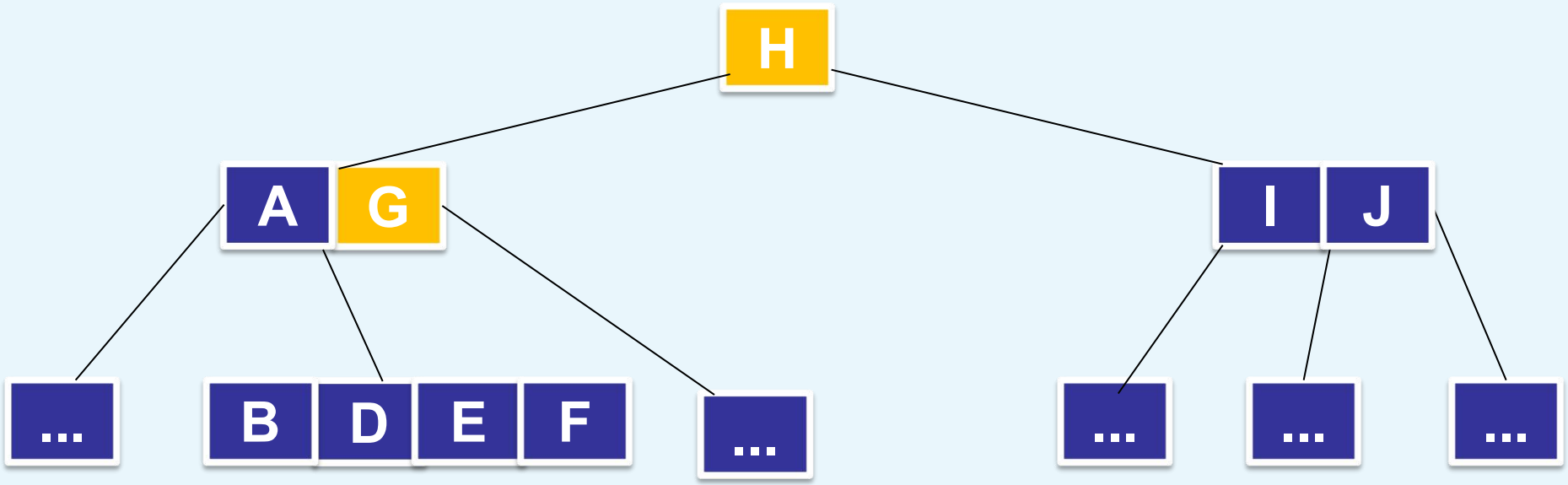
❖ **Gộp nút lá**



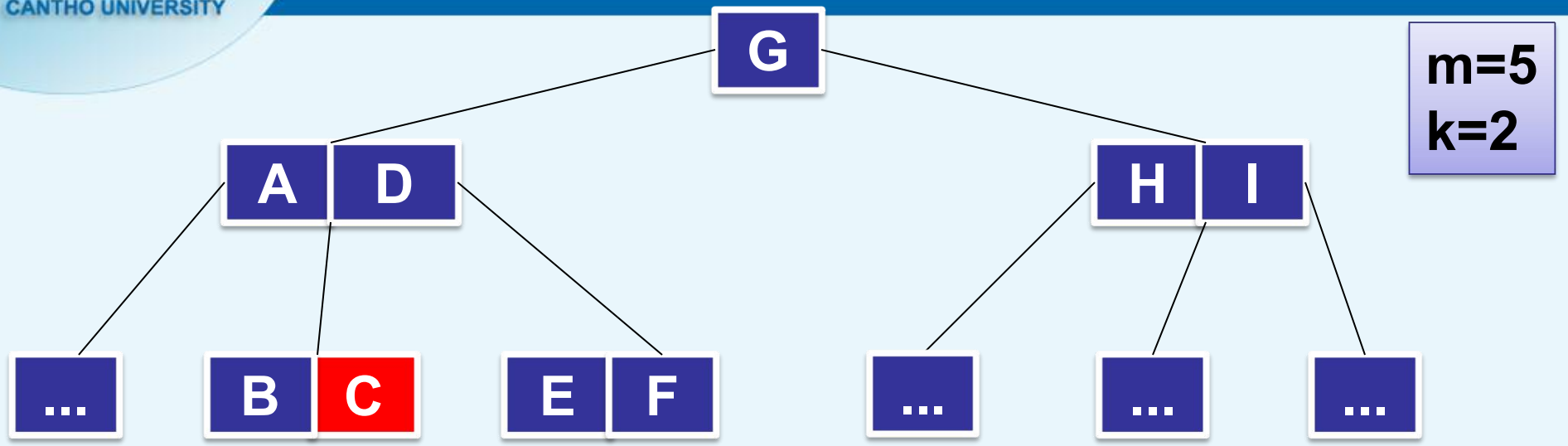
2.3 Trường hợp nút trung gian có ít hơn k khóa, lặp lại thao tác tái cân bằng cây như ở bước 2.1 và 2.2

- Thực hiện gộp ở nút cha

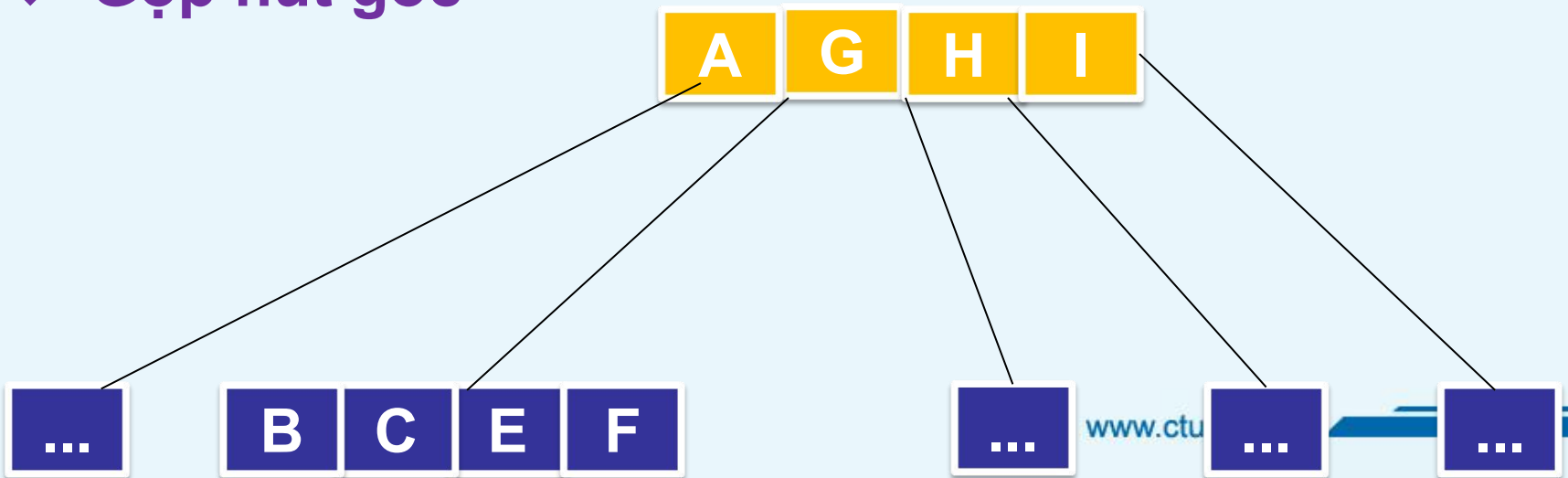
m=5
k=2



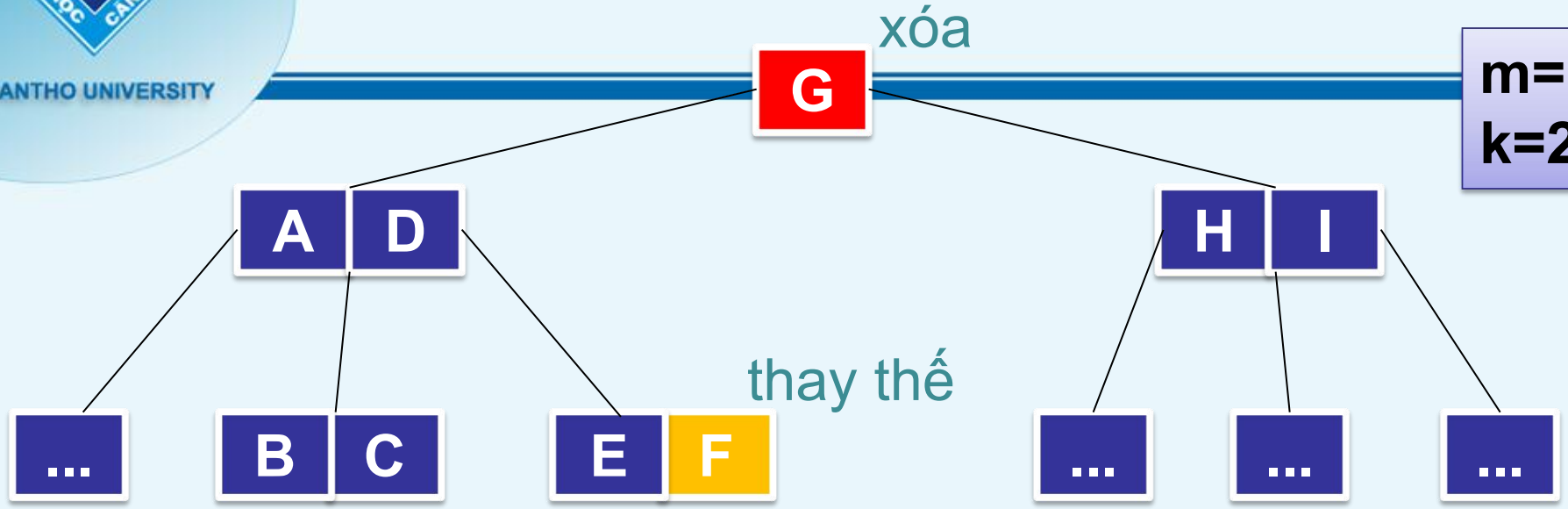
3. Gộp đến nút gốc và nút gốc chỉ còn 1 nút con, thì cho nút con làm nút gốc mới



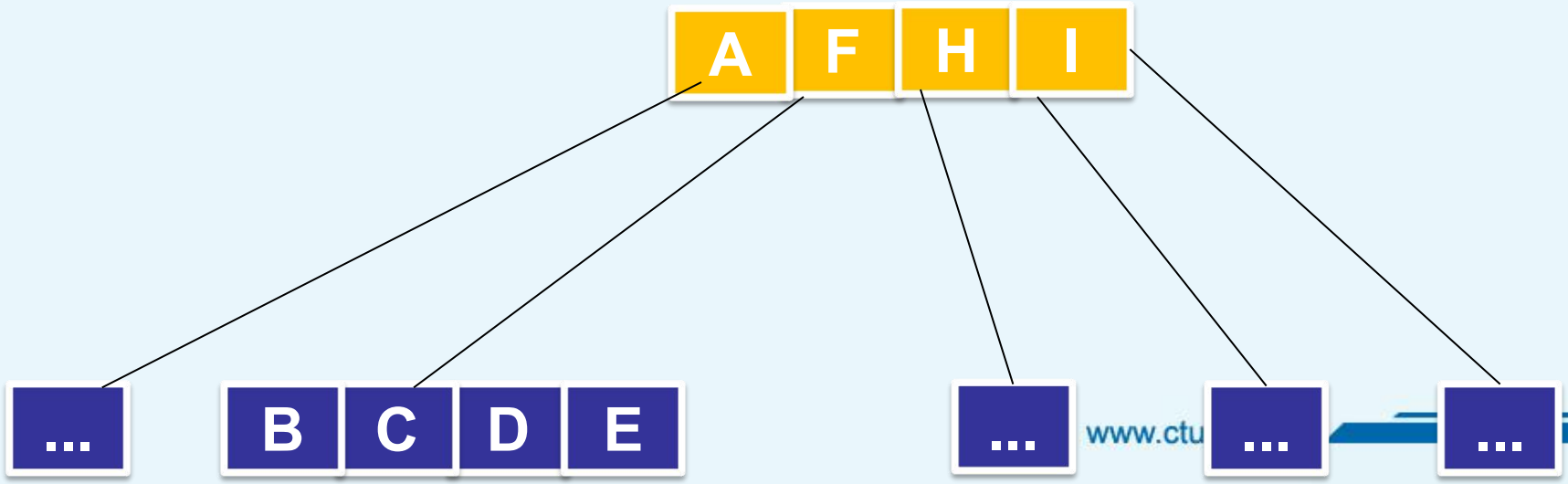
❖ Gộp nút gốc



Trường hợp xóa nút gốc



❖ **Quay lại bước 3: Gộp nút gốc**



6. Chỉ mục và băm

- **Chỉ mục**
  - Chỉ mục được sắp
  - Chỉ mục B+-cây
- **Băm**
  - Băm tĩnh
  - Băm động

6.2 Băm (hashing)

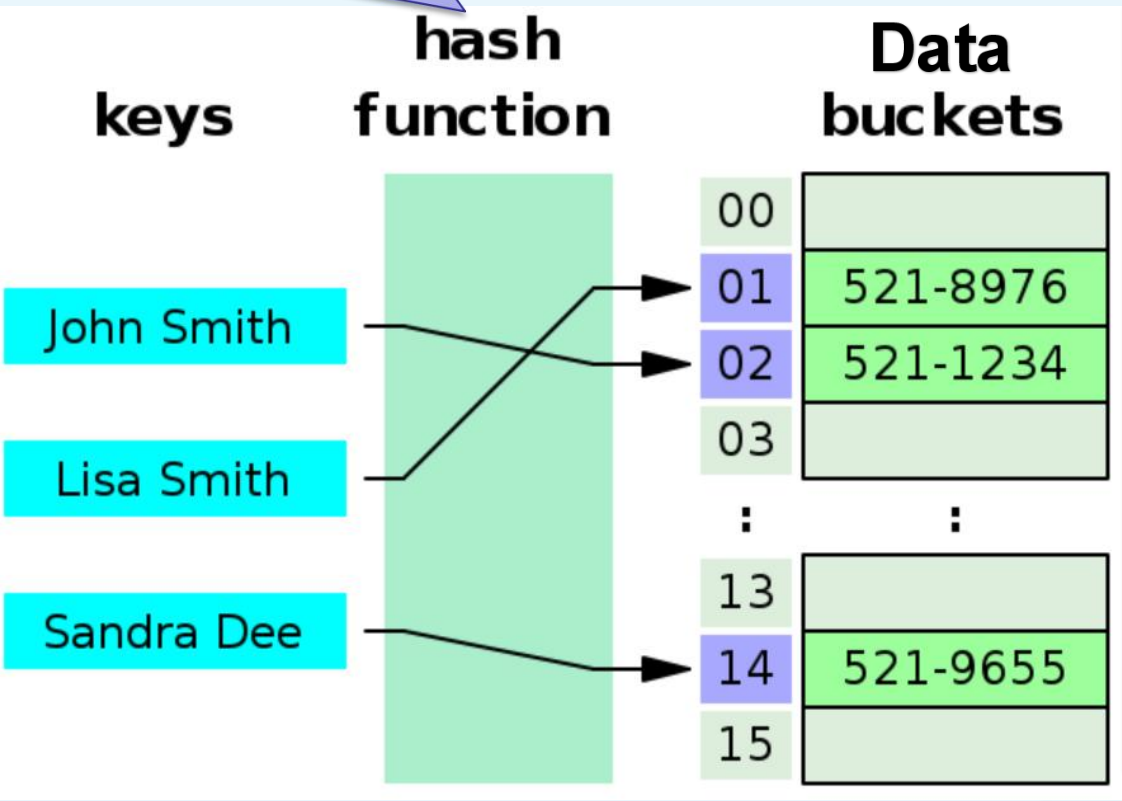
- Hạn chế của cơ chế tổ chức file tuần tự là:
  - Phải truy cập cấu trúc chỉ mục để định vị dữ liệu
  - Hoặc phải sử dụng tìm kiếm nhị phân
  - Tốn nhiều thời gian cho các thao tác I/O
- **Kỹ thuật băm** giúp bỏ qua các thao tác truy xuất cấu trúc Index. DL được lưu trữ dưới dạng các khối dữ liệu
- **Có 2 loại băm:** Băm tĩnh (static hashing) và Băm động (dynamic hashing)

CANTHO UNIVERSITY

❖ Các thuật ngữ trong phương pháp băm

Hàm băm: 1 hàm ánh xạ tất cả các tập hợp khóa tìm kiếm trỏ đến địa chỉ nơi các bản ghi được lưu trữ

Khóa: 1 thuộc tính hoặc tập hợp các thuộc tính giúp xác định duy nhất 1 hàng trong 1 bảng



Vùng nhớ dữ liệu: vị trí bộ nhớ nơi các bản ghi được lưu trữ

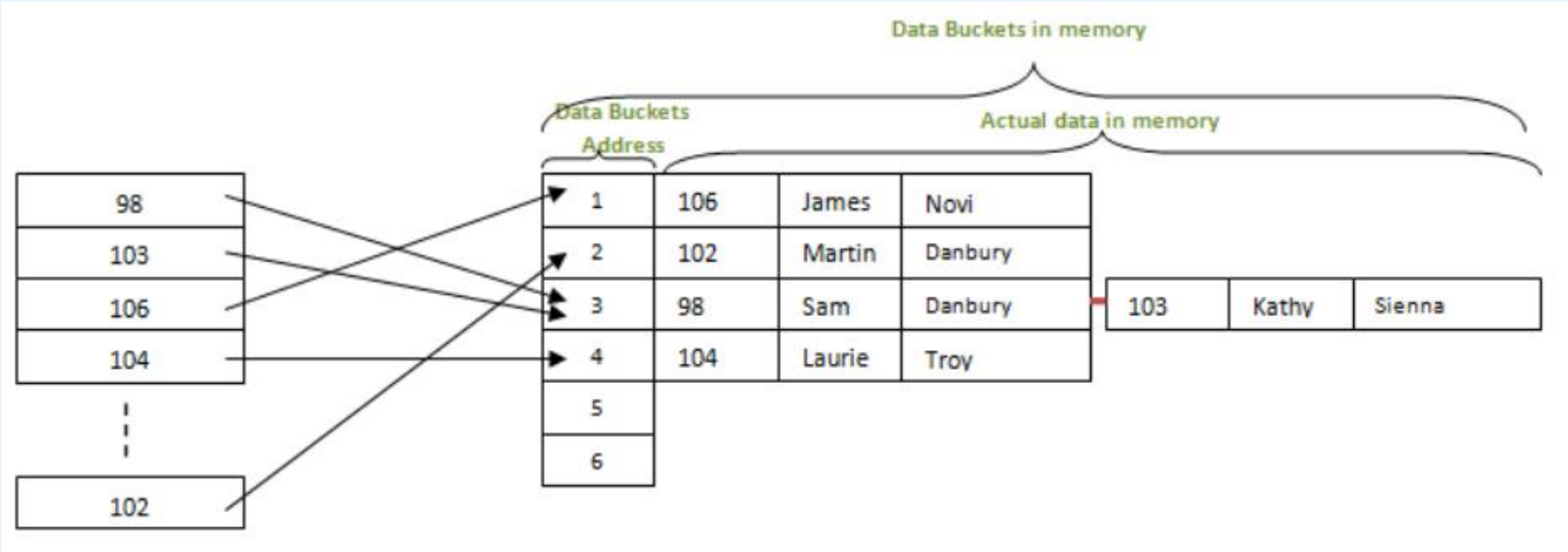
❖ Lựa chọn hàm băm (hash function)

- Một hàm băm tốt phải thỏa mãn các điều kiện sau:
  - Giảm thiểu va chạm
  - Tính toán nhanh
  - Các giá trị khóa được phân bổ đều trong bảng
  - Xử lý được các loại khóa có kiểu dữ liệu khác nhau

6.2 Băm (hashing)

6.2.1 Băm tĩnh

- Địa chỉ của một vùng nhớ dữ liệu khi được tính toán thì sẽ luôn giống nhau
- **Ví dụ:** hàm băm là mod (5) để xác định địa chỉ của khối dữ liệu

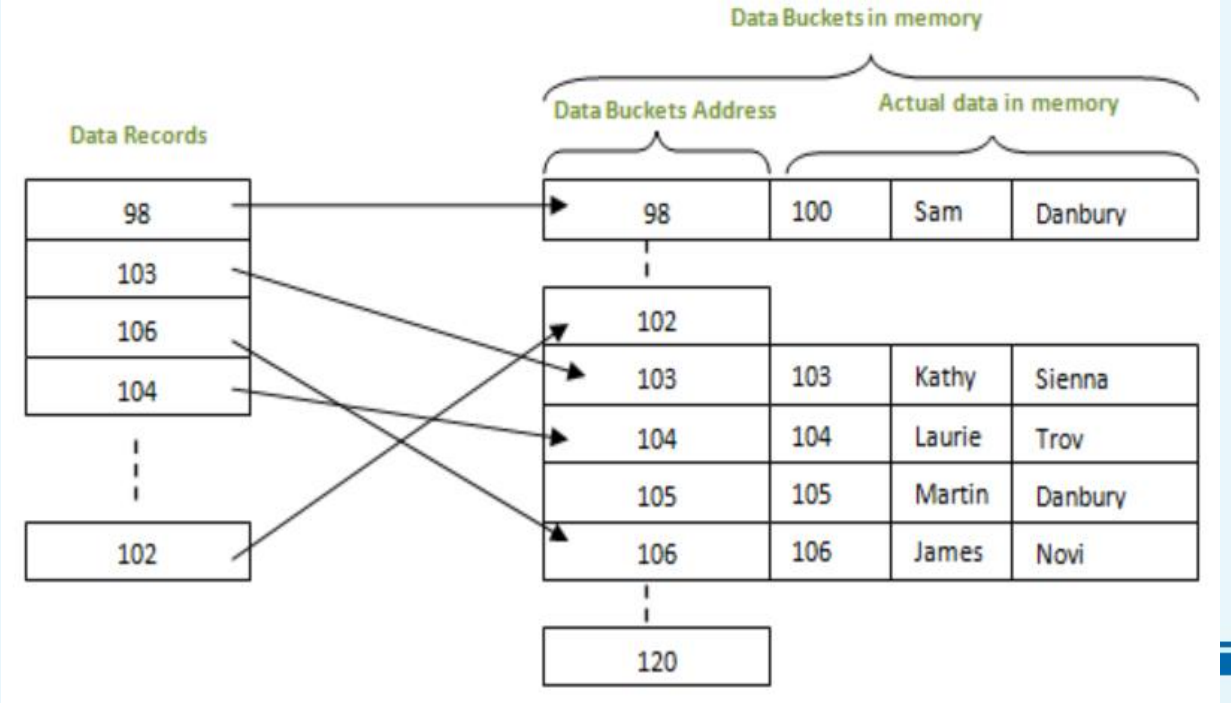


CANTHO UNIVERSITY

❖ Phương pháp tổ chức file băm

- Là nơi dữ liệu được lưu trữ tại các khối dữ liệu có địa chỉ được tạo ra bằng cách sử dụng hàm băm.
- Hàm băm có thể sử dụng bất kỳ giá trị cột nào để tạo địa chỉ.

Sử dụng khóa chính để tạo chỉ mục băm - địa chỉ của khối dữ liệu



❖ Phương pháp tổ chức file băm (tt)

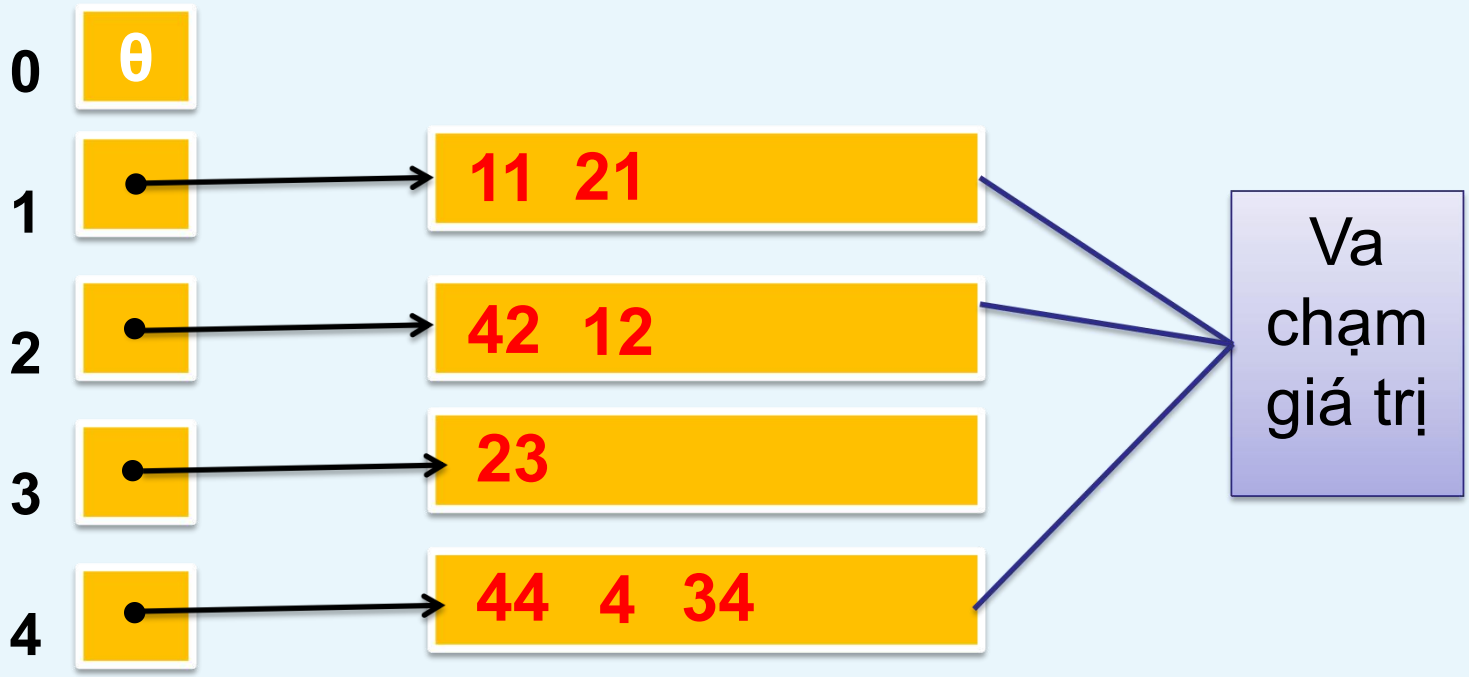
- Trong kỹ thuật băm, người ta cho đầu vào là một khoá tìm kiếm, hàm băm trên khoá này sẽ cho ra giá trị là vị trí của khối đĩa chứa mẫu tin
- Nếu gọi:
  - K: tập tất cả các giá trị khóa tìm kiếm
  - B: tập tất cả các địa chỉ bucket
  - h: hàm băm
- Thì h: **K → B**

Một số phương pháp xây dựng hàm băm

1. **Chia lấy số dư:** h(k) = k % m, với m là 1 số nguyên tố
2. **Nhân:** h(k) = một số chữ số ở "giữa" giá trị k^2
3. **Phân đoạn:** phân ra thành nhiều đoạn bằng nhau
   - Tách: căn thẳng lề các đoạn trái or phải ➔ Tính tổng
   - Gấp: gấp các đoạn lại theo đường biên tương tự như gấp giấy

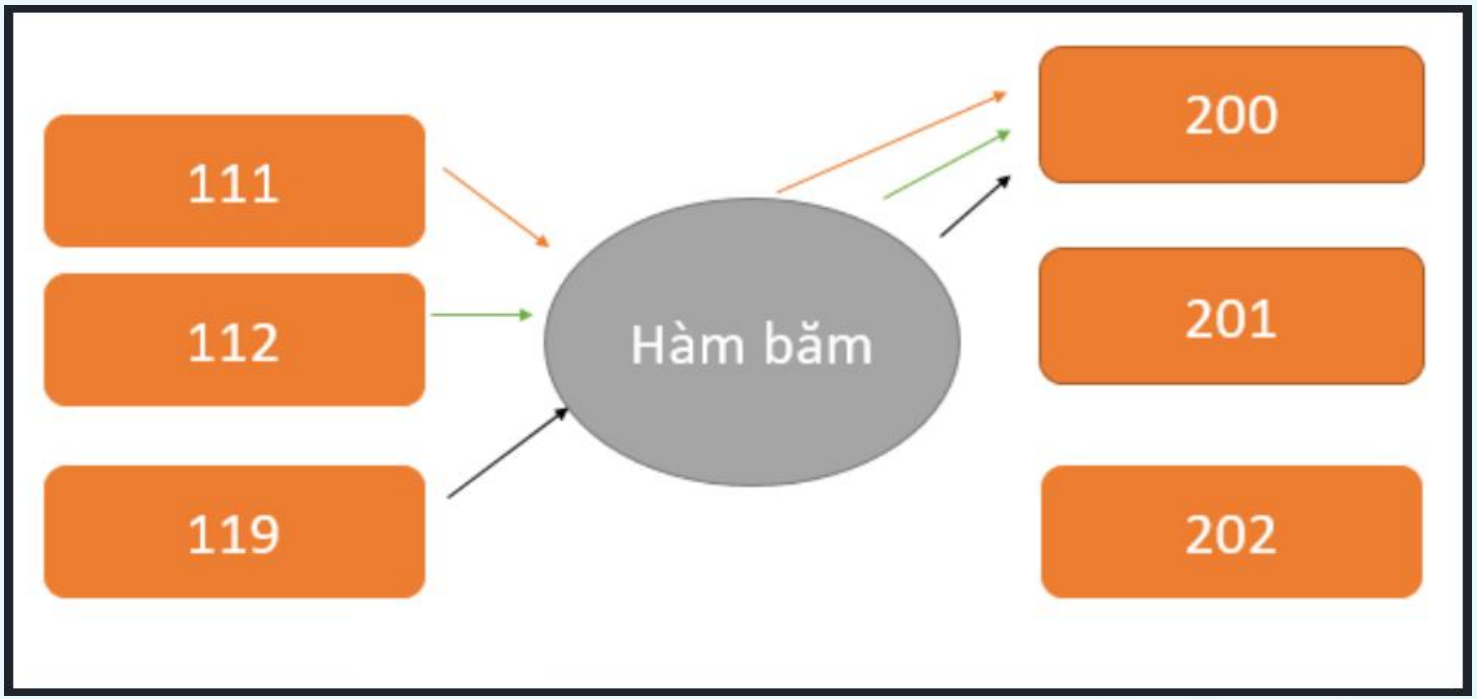
❖ Xây dựng bảng băm - Hash table

- Ví dụ: Giả sử có hàm h(k) = k%5
- Có các giá trị key: **11, 21, 44, 23, 42, 4, 34, 12**



Va chạm giá trị băm

- Là trạng thái khi các kết quả băm từ hai hoặc nhiều DL trong tập DL, ánh xạ vào cùng một vị trí trong bảng băm



Một số kỹ thuật giải quyết sự va chạm

- Direct Chaining (tạo dây chuyền): Ứng dụng danh sách liên kết
  - Separate chaining (chuỗi riêng biệt)
- Open Addressing (định địa chỉ mở): Dựa trên cấu trúc mảng
  - Linear probing (thăm dò tuyến tính)
  - Quadratic probing (thăm dò bình phương)
  - Double hashing (băm kép)

6.2 Băm (hashing)

6.2.1 Băm tĩnh (tt)

Băm mở

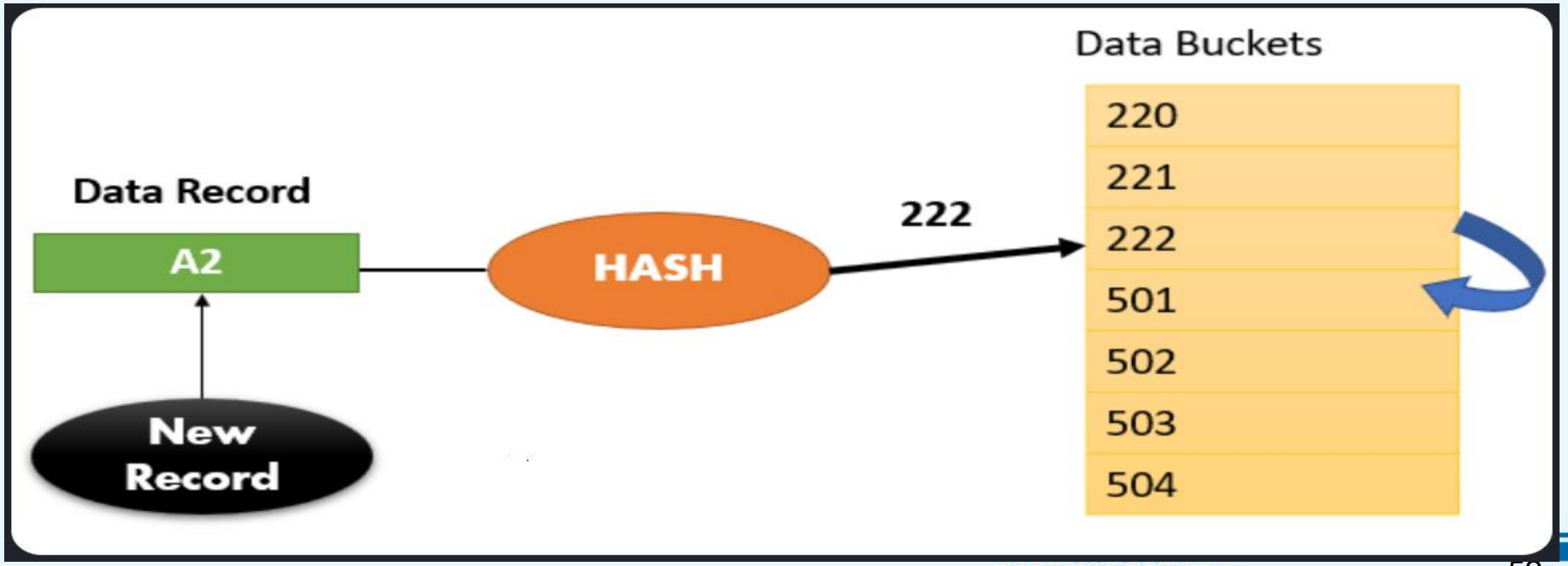
- Khắc phục bucket tràn
- Thăm dò tuyến tính

Băm đóng

- Khắc phục chuỗi tràn
- Chuỗi riêng biệt

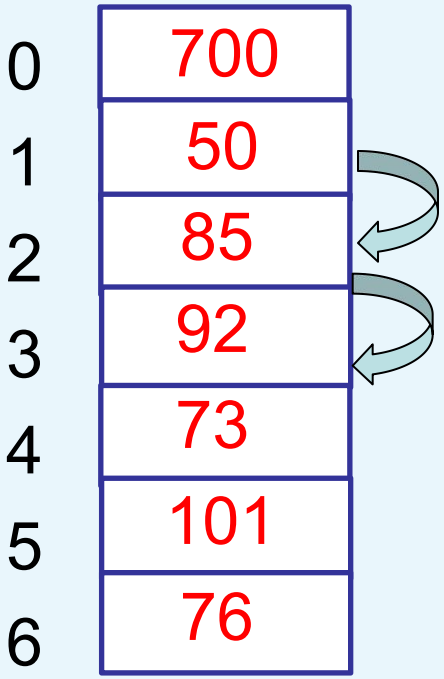
❖ Phương pháp băm mở

Băm mở: Thay vì ghi đè vùng nhớ dữ liệu cũ, thì vùng nhớ dữ liệu tiếp theo được sử dụng để nhập bản ghi mới (phương pháp thăm dò tuyến tính: tìm kiếm ô kế tiếp)



Ví dụ

- Xét hàm băm "**key mod 7**"
- dãy key = {50, 700, 76, 85, 92, 73, 101}



85 mod 7 => dư 1

92 mod 7 => dư 1

Xung đột thêm ô kế tiếp

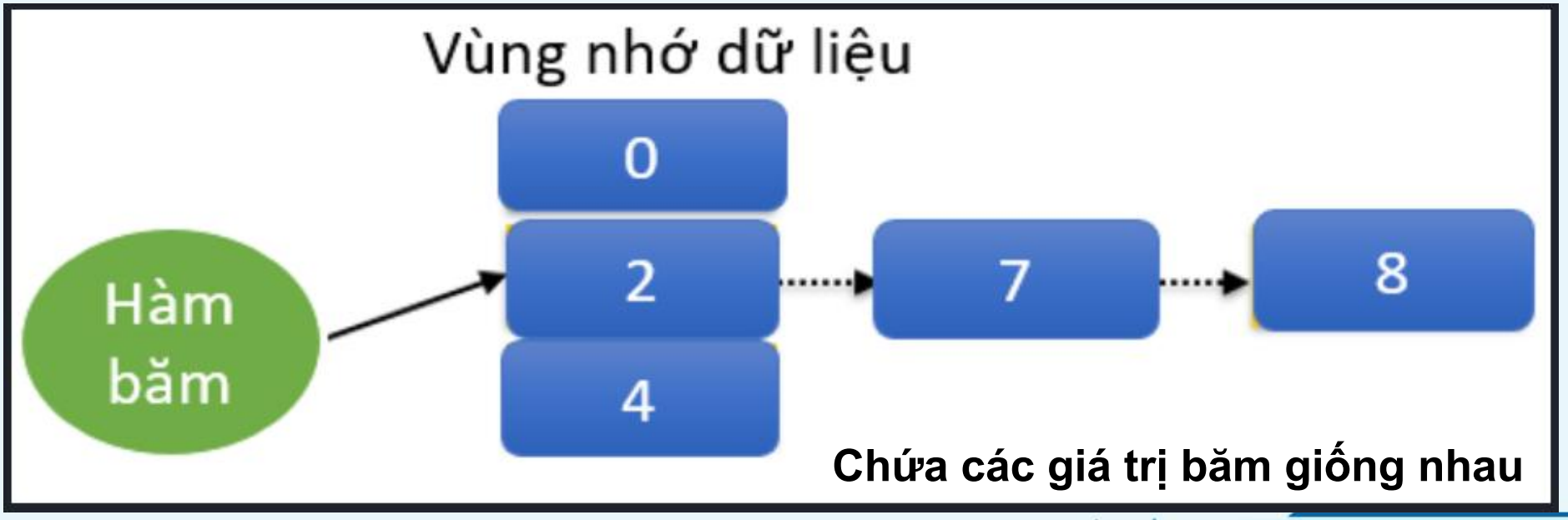
Xung đột thêm ô kế tiếp

73 mod 7 => dư 3

101 mod 7 => dư 3

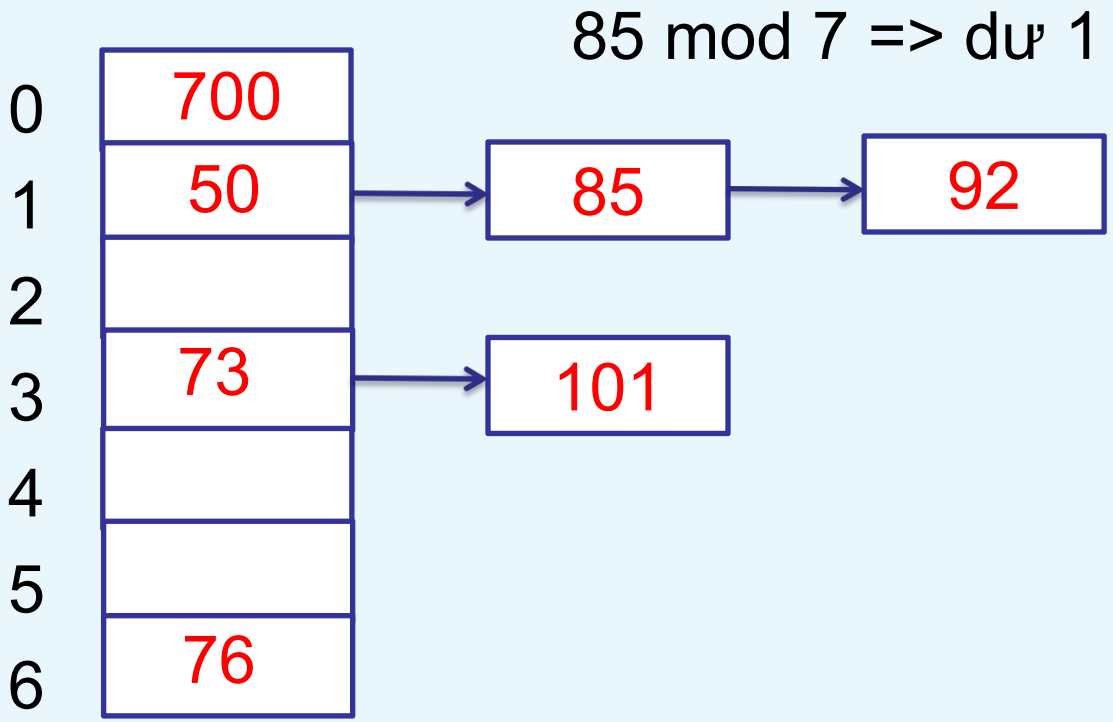
❖ Phương pháp băm đóng

Băm đóng: Khi các vùng nhớ dữ liệu đầy, một vùng nhớ mới sẽ được cấp phát cho cùng một hàm băm và kết quả sẽ được liên kết sau vùng nhớ trước đó (chuỗi tràn)



Ví dụ

- Xét hàm băm "key mod 7"
- dãy key = {50, 700, 76, 85, 92, 73, 101}



6.2 Băm (hashing)

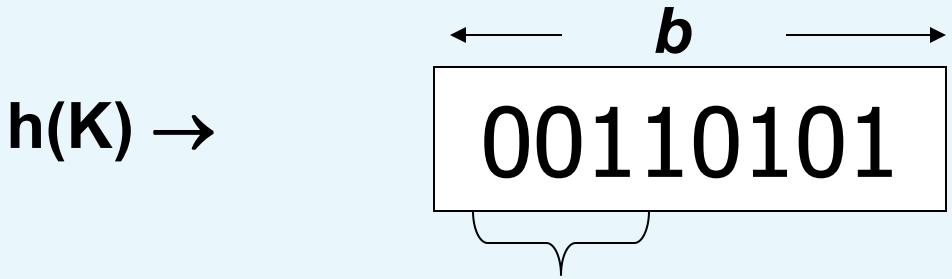
6.2.2 Băm động

Chúng ta đối phó như thế nào khi file dữ liệu phình to quá khả năng quản lý của bảng băm?

- Nếu tràn ⇒ Nới rộng, chọn hàm băm mới và tổ chức lại bảng băm ⇒ Tốn chi phí
- Dùng cơ chế băm động
  - Bảng băm tự nới rộng nhưng hàm băm không đổi
  - Kích thước của giá trị băm tăng tuyến tính theo sự lớn lên của dữ liệu

Băm động - Ý tưởng

Chỉ sử dụng i bits trong tổng số b bits chiều dài của kết quả hàm băm h



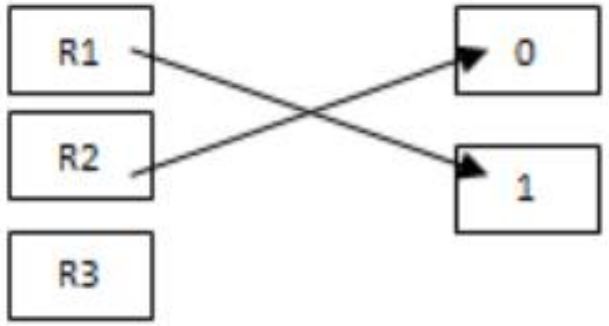
sử dụng *i* → tăng theo kích thước của file dữ liệu….

Ví dụ

Sử dụng 1 bit

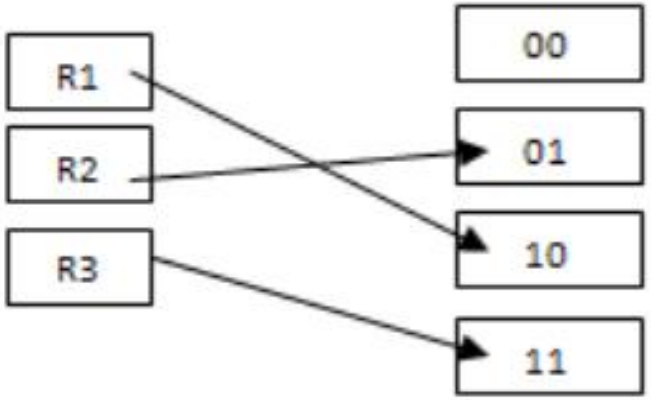
h (R1) →100100
h (R2) →010110
h (R3) →110110

Không có không gian cho R3



Sử dụng 2 bit

h (R1) →100100
h (R2) →010110
h (R3) →110110



HẾT CHƯƠNG 2