



Cấu Trúc Dữ Liệu Và Giải Thuật

---

Bài tập lớn 2

JJK RESTAURANT OPERATIONS  
(Phần 2 - Hồi tưởng)

---

nhóm thảo luận Code  
<https://www.facebook.com/groups/211867931379013>

Tp. Hồ Chí Minh, Tháng 11/2023



## Mục lục

<b>1</b>	<b>Phần 2 Mã hóa HuffTree</b>	<b>3</b>
1.1	Hàm buildHuff . . . . .	3
1.2	balanceTree . . . . .	4
1.3	balanceNode . . . . .	4
1.4	Test case . . . . .	4



# 1 Phần 2 Mã hóa HuffTree

## 1.1 Hàm buildHuff

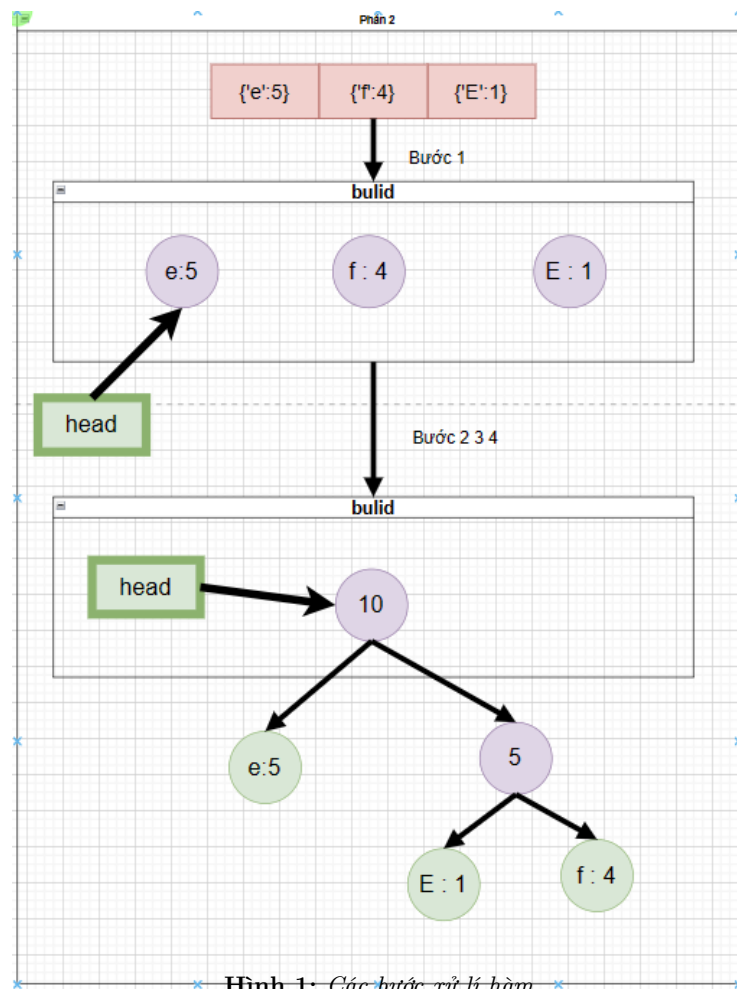
- **Đầu vào:** *freq* đã xử lý ở bước 1 theo chiều giảm dần
- **Trả về:** một cây đã được xử lý
- **Yêu cầu hàm:** làm trong file **buildHuff.cpp** trước
- **Hướng giải quyết theo các bước sau:** yêu cầu làm

**Bước 1:** chuyển *freq* thành *build* theo thứ tự 0 -> n, yêu cầu tạo ra một danh sách *node* theo chiều giảm dần giống *freq*

**Bước 2:** xử lý đến khi nào còn 1 *node* là *root* thì dừng, yêu cầu lấy 2 phần tử *min weight* trong *vector* để tạo thành một cây mới có *left* là node nhỏ nhất vừa tìm được, *right* là node nhỏ nhất thứ 2 vừa tìm được, **Chú ý cập nhật**  $\text{weight} = \text{left} \rightarrow \text{weight} + \text{right} \rightarrow \text{weight}$

**Bước 3:** tiến hành cân bằng *avl* nói rõ ở phần sau thông qua hàm *balanceTree*.

**Bước 4:** đưa node mới vào *build* vẫn đảm bảo được luôn giảm dần như ban đầu, **chú ý nếu bằng nhau thì xem như node mới luôn lớn hơn các node bằng giá trị *weight* trong *build* -> ý là xếp nó gần *head* hơn**



Hình 1: Các bước xử lý hàm



## 1.2 balanceTree

- **Đầu vào:** `Node* node`, `int& count` `node` là cây cần cân bằng, `count` số lượng các bước đã tiến hành quay. Nếu `count >= 3` thì sẽ không cân bằng nữa
- **Trả về:** trả về `root` của cây vừa được cân bằng
- **Yêu cầu hàm:** làm trong file `buildHuff.cpp` trước
- **Hướng giải quyết theo các bước sau:** yêu cầu làm

**Bước 1:** xét trường hợp phải dừng không tiến hành xử lý nữa

**Bước 2:** Tiến hành cân bằng `balanceNode` cho `root` của cây

**Bước 3:** Tiến hành cân bằng `balanceTree` cho `left` của cây

**Bước 4:** Tiến hành cân bằng `balanceTree` cho `right` của cây

## 1.3 balanceNode

- Hàm này chỉ là cân bằng `AVL` thôi yêu cầu đọc trong file lý thuyết  
<https://drive.google.com/drive/folders/1DoF0V6fcs0jT6Q550GS6Nk9QUSNzUD05>
- **Chú ý:**
  1. trường hợp LL -> tính là count được cộng thêm 1
  2. trường hợp RR -> tính là count được cộng thêm 1
  3. trường hợp RL -> tính là count được cộng thêm 2
  4. trường hợp LR -> tính là count được cộng thêm 2
  5. nếu trường hợp LL cũng là LR -> xét như Trường hợp LL
  6. nếu trường hợp RL cũng là RR -> xét như Trường hợp RR
  7. chú ý vì RL LR tính 2 lần nên khi count = 2 mà phải xử lý 1 lần quay này thì dừng không xử lý lần quy tiếp theo
  8. Tính `height` dung đệ quy

## 1.4 Test case

khuyến khích các bạn đóng góp test case 😊 😊 😊.

<https://drive.google.com/file/d/1gJJ3CMqcmL6dQBnFYsKSGqXavpdq1Vjj/view?usp=sharing>



nhóm thảo luận Code

<https://www.facebook.com/groups/211867931379013>

CHÚC CÁC EM HỌC TỐT

