# CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

## Bài tập Thực hành #5

Nguyễn Quản Bá Hồng (1411103)

19/12/2015

# Đề bài:

Bài thực hành gồm có 5 bài tập nhỏ và mỗi bài tập gồm 5 bộ test. Nội dung các bài tập thực hành là xây dựng cây nhị phân tìm kiếm (NPTK) với dữ liệu đầu vào là các số nguyên sau đó thực hiện các yêu cầu tương ứng:

- Bài 1: Đếm số phần tử của cây NPTK.
- Bài 2: Đếm số node lá của cây NPTK.
- Bài 3: Đếm số node là cha của node lá.
- Bài 4: Tính trung bình cộng giá trị các node có hai node con.
- Bài 5: Tìm giá trị của node trên cây gần nhất với trung bình cộng giá trị các node trên cây.

**Lưu ý:**

- Quy định về cách đọc file input và ghi file output giống như các BTTH trước.
- Các số nguyên đọc từ file và đưa vào cây NPTK theo thứ tự từ trái sang phải.
- Các số nguyên trong file data.txt cách nhau bằng 1 khoảng trắng.
- Nếu không có node nào có hai node con trên cây nhị phân thì kết quả của bộ test đó sẽ là NaN.
- Nếu tìm được hơn một giá trị của các node trên cây đều gần nhất với trung bình cộng giá trị các node trên cây thì sẽ đưa tất cả các giá trị đó vào kết quả.

***Ví dụ:***

Các giá trị node của cây: 3 10 2 11.
Giá trị trung bình cộng của các giá trị của các node là 6.5.
Ta thấy giá trị 3 và 10 đều cách giá trị 6.5 một khoảng 3.5.
Vậy kết quả của bài tập sẽ là: 3 10.

# Bài làm

Bao gồm 3 file như sau: 1411103_defs.h, 1411103_sub.cpp, 1411103.cpp. Nội dung cụ thể từng file như sau:

1. **1411103_defs.h**

   **1.1. Cấu trúc ifndef**

   Sử dụng cấu trúc sau để tổ chức các tệp thư viện sao cho chúng có thể được kết nối nhiều lần vào 1 chương trình nguồn mà không gây lỗi.

```
#ifndef DEFS_H_
#define DEFS_H_
#endif /* DEFS_H_ */
```

   **1.2. Khai báo các thư viện cần sử dụng**

   Trong bài thực hành này, chúng ta cần sử dụng các thư viện nhập xuất chuẩn của C/C++, thư viện các hàm toán học, thư viện xử lý file, thư viện xử lý chuỗi

```
#include <iostream>
#include <cmath>
#include <fstream>                                                    // thư
viện xử lý file của C++
#include <string>
```

   **1.3. Khai báo các hằng số cần sử dụng**

```
#define INPUT_FILE  "data.txt"                                  // file lấy dữ
liệu: data.txt
#define OUTPUT_FILE  "output\\1411103.txt"              // xuất ra file
1411103.txt nằm trong thư mục output, thư mục output nằm chung với các file.cpp
.h
#define Error 846252.1410232
#define MAX 100
```

   **1.4. Định nghĩa kiểu dữ liệu node**

   1 node trong cây bao gồm các thành phần: dữ liệu (số nguyên), node con trái, node con phải

```
struct node {
int data;
node* left;
node* right;
};
```

   **1.5. Định nghĩa kiểu dữ liệu cây nhị phân tìm kiếm**

   Chứa 1 con trỏ tới node gốc của cây nhị phân tìm kiếm

```
struct bstree{
       node* root;
};
```

   **1.6. Các hàm xử lý trên cây nhị phân tìm kiếm**

   Cần sử dụng các hàm có chức năng như sau:

   - Kiểm tra xem cây nhị phân có rỗng hay không.

     ```
     int isEmpty(bstree* bst);
     ```

   - Khởi tạo cây nhị phân tìm kiếm

     ```
     void initBSTree(bstree* &bst);
     ```

   - Duyệt cây theo 3 cách.

     + Duyệt cây theo thứ tự trước (node – left – right )

     ```
     void NLR(bstree* bst);
     ```

     + Duyệt cây theo thứ tự giữa (left- node – right )

     ```
     void LNR(bstree* bst);
     ```

+ Duyệt cây theo thứ tự sau (left – right – node)

```
void LRN(bstree* bst);
```

- Tìm kiếm node chứa dữ liệu x trong cây nhị phân tìm kiếm

```
node* searchNode(bstree* bst, int x);
```

- Chèn node chứa dữ liệu x vào cây nhị phân tìm kiếm

```
int insertNode(bstree* &bst, int x);
```

- Xóa node chứa dữ liệu x trong cây nhị phân tìm kiếm

```
int deleteNode(bstree* &bst, int x);
```

- Tìm phần tử nhỏ nhất bên phải (sử dụng trong thao tác xóa node)

```
void searchStandFor(node* &p, node* &q);
```

- Kiểm tra 1 node có phải là node lá hay không.

```
int isLeaf(node* N);
```

- Tính chiều cao của cây nhị phân tìm kiếm

```
int getHighofTree(bstree* bst);
```

- Hủy 1 cây nhị phân tìm kiếm

```
void removeTree(bstree* &bst);
```

- Đếm số lượng phần tử của cây nhị phân tìm kiếm

```
int countNode(bstree* bst);
```

- Đếm số lượng node lá trong cây nhị phân tìm kiếm

```
int countLeaf(bstree* bst);
```

- Đếm số lượng node cha mẹ của cây nhị phân tìm kiếm

```
int countParent(bstree* bst);
```

- Tính giá trị trung bình các node có 2 node con

```
float averageNodeHave2Node(bstree* bst);
```

- Tính giá trị trung bình tất cả các node của cây

```
float averageNode(bstree* bst);
```

- Tìm node có dữ liệu gần nhất với số cho trước

```
void findNear(bstree* bst, int arr[], int &size);
```

- Đếm số lượng các node có 2 node con

```
int countNodeHave2Node(bstree* bst);
```

- Hàm dành riêng cho bài 4

```
void bai4(bstree* bst);
```

**2. 1411103_sub.cpp**

**2.1. Viết các hàm đã được khai báo trong file header 1411103_defs.h**

Viết các hàm có chức năng sau:

- Kiểm tra cây nhị phân tìm kiếm rỗng
- Khởi tạo cây nhị phân tìm kiếm.
- Duyệt cây theo thứ tự node-left-right
- Duyệt cây theo thứ tự left-node-right
- Duyệt cây theo thứ tự sau
- Tìm kiếm node chứa số nguyên x cho trước
- Thêm 1 node có dữ liệu x vào cây nhị phân tìm kiếm
- Hủy 1 node có dữ liệu x trong cây nhị phân tìm kiếm
- Tìm phần tử nhỏ nhất bên phải
- Kiểm tra 1 node có phải là node lá hay không
- Tính chiều cao của cây nhị phân tìm kiếm
- Hủy 1 cây nhị phân tìm kiếm
- Đếm số lượng phần tử của cây

- Đếm số node lá của cây
- Đếm số node là cha của node của cây
- Đếm số node có 2 node con
- Tính trung bình các node có 2 node con
- Xử lý bài số 4
- Tính trung bình tất cả các node của cây nhị phân tìm kiếm
- Tìm node có giá trị gần nhất với giá trị trung bình cộng của tất cả các node

3. **1411103.cpp**

   Các thao tác xử lý file, lấy dữ liệu từ file data.txt, xử lý bằng các hàm đã được cài đặt ở file 1411103_sub.cpp rồi xuất ra file 1411103.txt nằm trong thư mục output.