

Họ và tên: Nguyễn Công Bình
 Mã số sinh viên: 19964
 Lớp: 64IT5

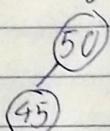
1. AVL: Bài thực hành số 4.

Câu 1:

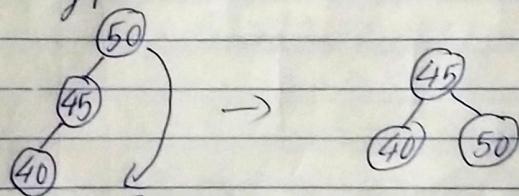
Bước 1: Tạo nút gốc 50, kiểm tra hệ số cân bằng = 0

(50)

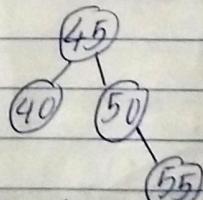
Bước 2: Thêm nút 45, hệ số cân bằng nút 50 bằng 1



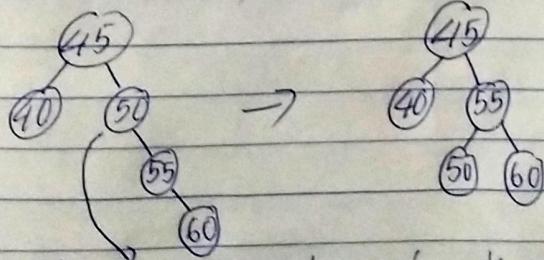
Bước 3: Thêm nút 40, hệ số cân bằng nút 50 bằng 2
 Nút cân bằng, cần xoay phải



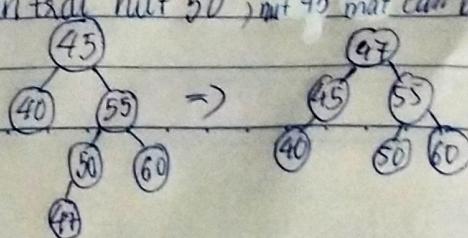
Bước 4: Thêm nút 55, kiểm tra nút 45, chưa mất cân bằng



Bước 5: Thêm nút 60, nút 50 sẽ mất cân bằng, cần xoay trái



Bước 6: Thêm nút 47 vào bên trái nút 50, nút 45 mất cân bằng



HẢI TIỀN

câu 2:

struct AvlNode {

int data;

AvlNode *Lchild;

AvlNode *Rchild;

int height;

{};

typedef struct AvlNode *tree_avl;

// Hỗn xoay trái:

void xoayTrai(tree_avl &root){

AvlNode *p;

p = root->Rchild;

root->Rchild = p->Lchild;

p->Lchild = root;

root->height = max(height((root->Lchild)), height((root->Rchild)) + 1);

p->height = max(height(p->Lchild), height(p->Rchild)) + 1; root = p;

{};

// Hỗn xoay phải:

void xoayPhai(tree_avl &root){

AvlNode *p;

p = root->Lchild;

root->Lchild = p->Rchild;

p->Rchild = root;

root->height = max(height((root->Lchild)), height((root->Rchild)) + 1);

p->height = max(height((p->Lchild)), height((p->Rchild)) + 1);

root = p;

{};

2. Heap

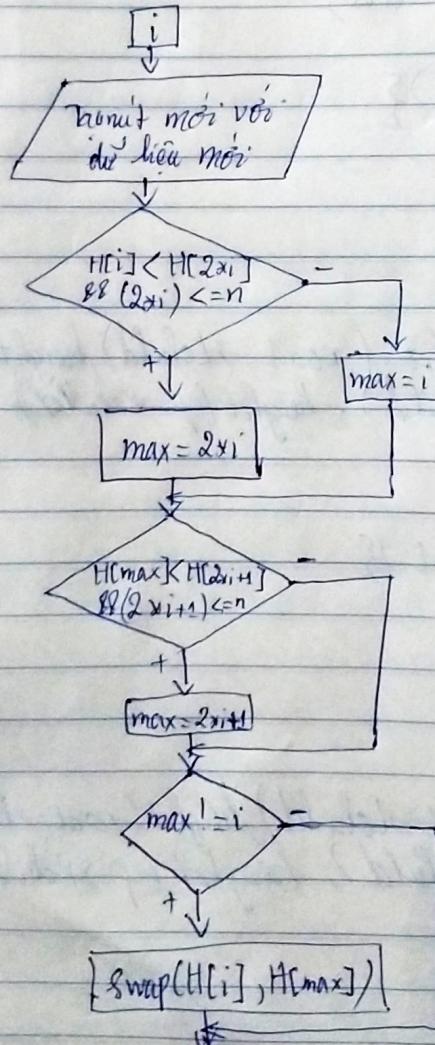
câu 1.

(1) Để một cây lèo thành max heap:

- Nó là 1 cây nhị phân đầy đủ (hoàn chỉnh)

- Cadao trong mỗi nút cha phải lớn hơn hoặc bằng giá trị các nút con trái, phải! (nút cha là nút có giá trị lớn nhất của max heap)

(2) Sơ đồ khái thuật toán biểu diễn các bước xác định max heap

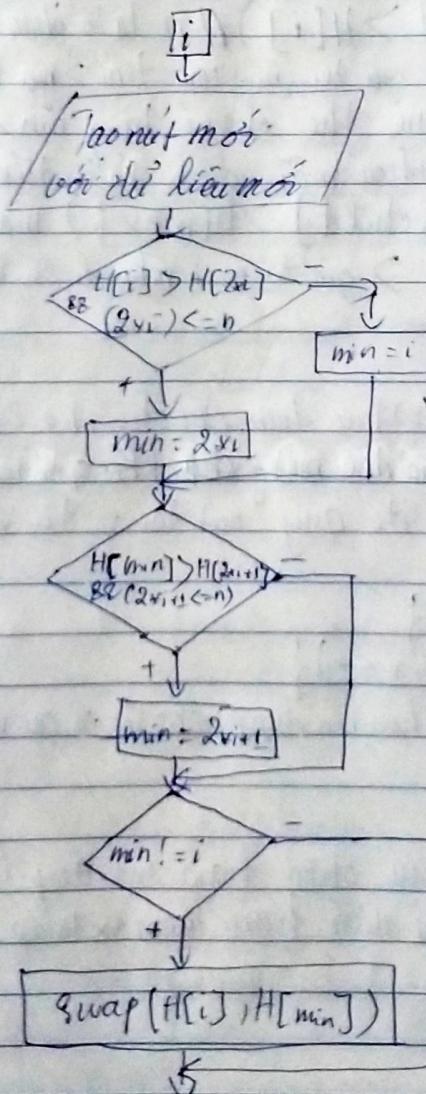


② Đổi một cây nhị phân thành minheap:

Nó là 1 cây nhị phân đầy đủ (hoàn chỉnh)

Giai thừa ở cột trái phải phải lớn hơn hoặc bằng giá trị nút cha
(Nút cha có giá trị nhỏ nhất trong cây min heap)

③ Số độ khởi thuật toán biểu diễn xác suất xác định minheap



Câu 2:

```

void maxHeap(int H[], int i) {
    int max; // chỉ số của phần tử lớn nhất trong bộ 3 nút hiện thời, mìn
    // con trái và phải của nút hiện thời là (i, 2*i, 2*i+1)
    int Lchild = 2*i; Rchild = 2*i+1; // vì chỉ của nút con bên trái, bên phải
    if [Lchild <= n && H[Lchild] > H[i]] // i/n là số phần tử trong mảng
        max = Lchild; // khi giá trị nút con trái > giá trị nút cha hiện thời
    // và nó bằng max
    else max = i; // không thỏa mãn điều kiện trên
    if (Rchild <= n && H[Rchild] > H[max]) max = Rchild;
    // khi giá trị nút con phải > giá trị nút mìn hiện tại, cùn nó là
    // cho nó bằng max.
    if (max != i) {
        swap(H[i], H[max]); // thực hiện đổi chỗ cho 2 phần tử này
        // và mìn đang xét khác với mìn trước
        maxHeap(H, max); // gọi đệ quy nút tài và thi mới
    }
}

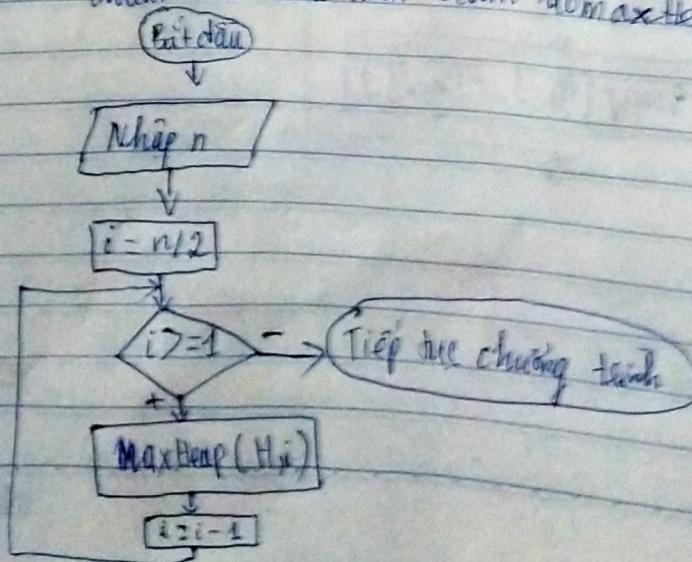
```

```

void TaomaxHeap (int H[]) {
    for (int i = n/2; i >= 1; i--) {
        MaxHeap(H, i); // áp dụng hàm maxHeap cho tất cả các nút từ nút cuối
    }
}

```

- ② Sơ đồ khái niệm thuật toán biểu diễn hàm maxHeap (hình ở phần 1 của)
- ③ Sơ đồ khái niệm thuật toán biểu diễn hàm TaomaxHeap



Tieng 6

NO _____
DATE _____

câu 3:

void minHeap (int H[], int i)

{

int min, // chỉ số của phần tử nhỏ nhất trong bộ 3 mảng

// mảng con trái và phải của mảng hiện thời là C, 2xi, 2xi+1

int Lchild = 2xi, // vì trái của mảng con bên trái

int Rchild = 2xi+1; // vì phải của mảng con bên phải

if (Lchild <= n and H[Lchild] < H[i])

min = Lchild; // n là số phần tử trong mảng

if khi vị trí mảng con trái <= n và giá trị của nó < giá trị mảng hiện tại

nhưng không

else min = i; // ngược lại với điều kiện trên

if (Rchild <= n and H[Rchild] < H[min]) min = Rchild;

if khi vị trí mảng con phải <= n và giá trị của nó < giá trị mảng hiện tại

cho nó bằng min

if (min != i) {

swap (H[i], H[min]); // thực hiện để cho chỗ của phần tử này

và vị trí đang xét khác vị trí min */

minHeap (H, min); // gọi để quy nạp mảng sau vị trí mới.

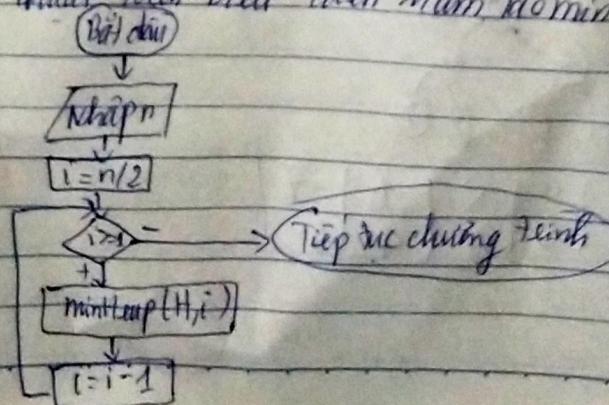
void TaoMinHeap (int H[]) {

for (int i = n/2; i > -1; i--) {

minHeap (H, i); // áp dụng hàm minHeap cho tất cả các mảng từ mảng

(1) Sắp xếp thuật toán minHeap (hình ảnh phần 2 bài 1)

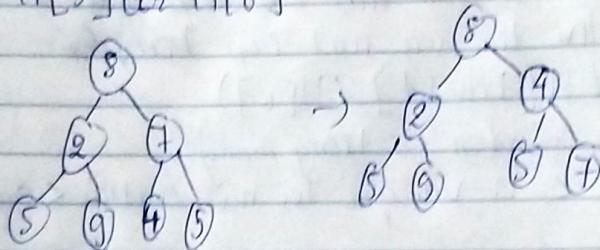
(2) Sắp xếp thuật toán biểu diễn hàm TaoMinHeap



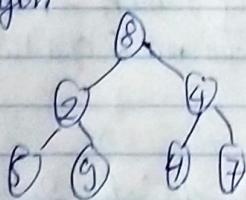
Câu 4:

i	1	2	3	4	5	6	7
H[i]	8	9	7	5	9	4	5

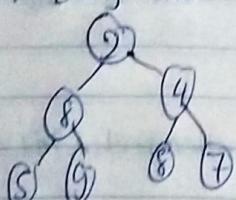
khi $i = n/2 = 3$ xét bộ 3 nút斐波那契 3, 6, 7 (min nút 6)
 đối chéo $H[3] \cup H[6]$



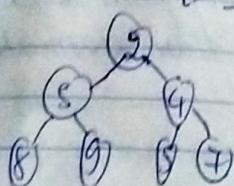
khi $i = 2$, $H[2] = 5$. Xem 3 mảng thứ 2, 4, 5 (mảng số 2),
giữ nguyên



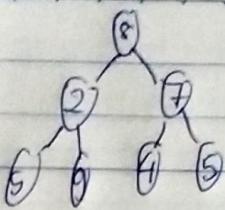
khi $i = 1$, $H[1] = 8$ (xem bài 3 nút thứ 1, 2, 3 (màn hình))
 ⇒ cần chia $H[1]$ với $H[2]$:



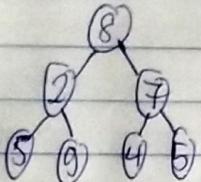
khi $H[2] = 8$, xem bộ 3 nút 2, 4, 5 (min ở nút 4)
 \Rightarrow đổi chỗ $H[2]$ với $H[4]$.



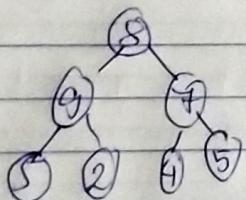
b) $i_{H(1)} \{ 1 | 8 | 2 | 7 | 5 | 9 | 4 | 5 | 7 \}$



Với $i=3$, xét bộ 3 nút 3, 6, 7 \rightarrow max nút 3 \Rightarrow giữ nguyên.



Với $i=2$, xét bộ 3 nút 2, 4, 5 \rightarrow max nút 5 \rightarrow đổi chỗ $H[5]$ với $H[2]$.



Với $i=1$, xét nút 1, 2, 3 \rightarrow max nút 2 \rightarrow đổi chỗ $H[2]$ với $H[1]$

