Baocao CK 1 - Báo cáo

cấu trúc dữ liệu (Đại học Tôn Đức Thắng)

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN/ĐỒ ÁN CUỐI KÌ MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Báo cáo cuối kỳ tổng hợp môn Cấu trúc dữ liệu và giải thuật.

Người hướng dẫn: **GV. HỒ THỊ THANH TUYẾN**
Người thực hiện: **PHAN VĂN KHÁNH – 52000674**
Lớp **: 20050201**
Khoá **: 24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN/ĐỒ ÁN CUỐI KÌ MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Báo cáo cuối kỳ tổng hợp môn Cấu trúc dữ liệu và giải thuật.

Người hướng dẫn: **GV. HỒ THỊ THANH TUYẾN**
Người thực hiện: **PHAN VĂN KHÁNH – 52000674**
Lớp **: 20050201**
Khoá **: 24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

Em xin gửi lời cảm ơn sâu sắc tới cô Hồ Thị Thanh Tuyến và Thầy Dung Cẩm Quang đã hướng dẫn em môn Cấu trúc Dữ liệu và Giải thuật trong học kì vừa rồi. Em rất biết ơn thầy cô đã dạy hướng dẫn cho em những kiến thức chuyên môn rất thực tế để em áp dụng vào công việc sau này.

Em xin chân thành cảm ơn thầy cô!

ĐỒ ÁN ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của TS Nguyễn Văn A;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Phan Văn Khánh

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Khi tình hình dịch bệch còn diễn biến phức tạp, ngày trở lại trường học trực tiếp còn chưa có kế hoạch và hình thức thi môn Cấu trúc Dữ liệu và Giải thuật chuyển thành làm báo cáo cuối kì thay vì thi trực tiếp tại trường. Nắm được tình hình đó em đã xác định học nghiêm túc trong các buổi học online để không bỏ lỡ kiến thức. Khi mà kì học sắp kết thúc, nhận được đề tài báo cáo cuối kì, em đã bắt tay vào ôn tập kiến thức học trên lớp online và lớp thực hành.Sau một vài ngày ôn tập, em tìm hiểu các câu hỏi trong đề và xác định lượng kiến thức và chủ đề liên quan mà mình sẽ áp dụng để giải các câu này.Em đã tham khảo một số file sách trên mạng Internet và ghi chú lại những kiến thức trọng tâm, sau đó em đã giải đáp lần lượt các câu hỏi. Tuy ban đầu gặp phải rất nhiều khó khăn, nhưng kiên trì từng ngày em cũng tìm ra lời giải. Cuối cùng em cũng hoàn thành bài báo cáo và nộp đúng hạn.

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

f *Tần số của dòng điện và điện áp (Hz)*

p *Mật độ điện tích khối (C/m3)*

CÁC CHỮ VIẾT TẮT

CSTD Công suất tác dụng

MF Máy phát điện

BER Tỷ lệ bít lỗi

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

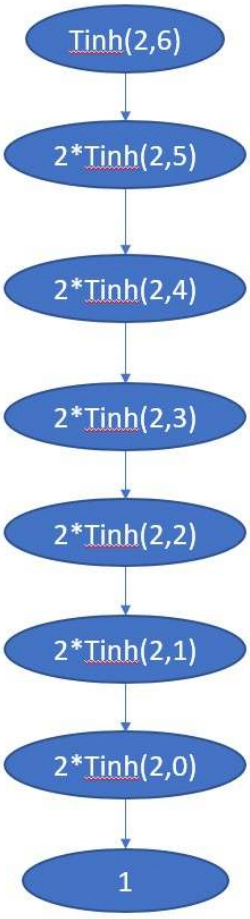
NỘI DUNG BÁO CÁO

CÂU 1: ĐỆ QUY

a)

- Nhóm 1: Giải bài toán x^y

Bài toán này thuộc dạng đệ quy tuyến tính, để tính được ta cần xác điều kiện dừng và công thức truy hồi.Hàm có 2 tham số truyền vào là x và y, điều kiện dừng là y=0 thì sẽ trả về 1,nếu y≠0 thì trả về tích của x và lời gọi hàm đệ quy.Sau mỗi lần gọi đệ quy thì y sẽ giảm đi 1 đơn vị.

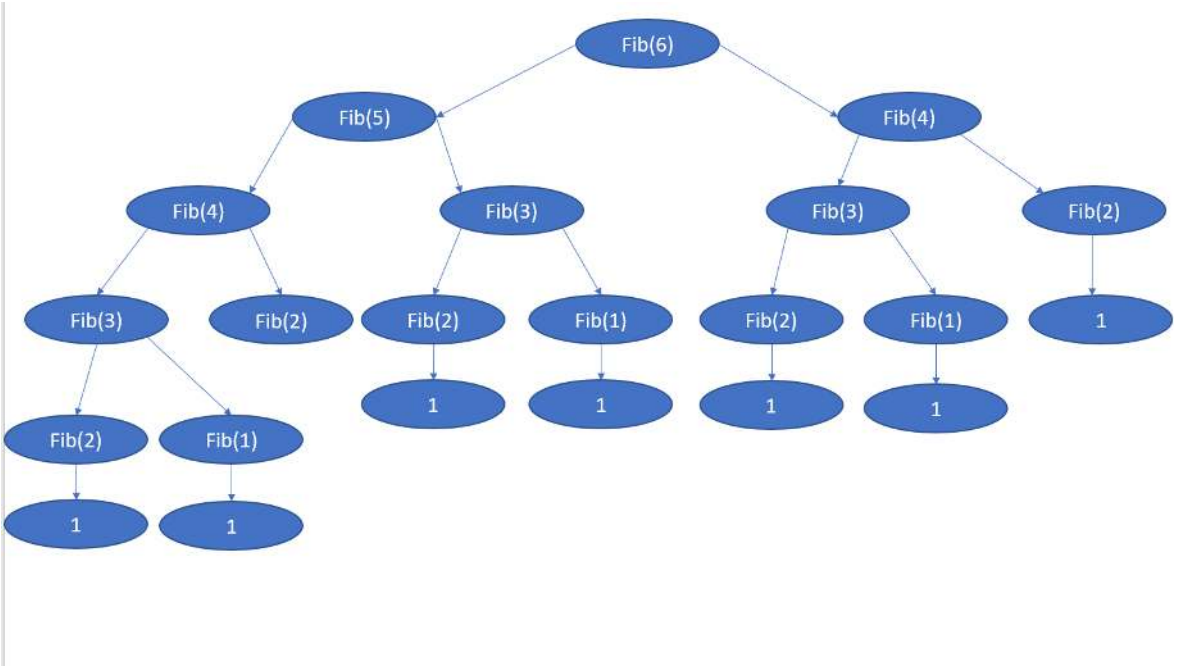


HÌNH 1.1

- Nhóm 2: Giải bài toán tính số fibonacci thứ n

Bài toán này thuộc dạng đệ quy nhị phân, ta cũng cần xác định điều kiện dừng và công thức truy hồi. Với bài toán tìm số fibonacci, ta biết số(n)= số(n-1) +số(n-2),

mà trong dãy số fibonacci có 2 số hạng đầu tiên ngoại lệ nên ta xác định điều kiện dừng là nếu n≤2 thì sẽ trả về 1,nếu không sẽ trả về tổng của 2 số trước đó.



HÌNH 1.2

b) Thực hành

```
public class Dequy {
    public static int Tinh(int x,int y){
        if(y==0) return 1;
        return x*Tinh(x, y-1);
    }
    public static int fib(int n){
        if(n<=2) return 1;
        return fib(n-1)+fib(n-2);
    }

    public static void main(String args[]){
        System.out.println(Tinh(2,6));
        System.out.println(fib(6));
```

```
    }
  }
```

CÂU 2: SẮP XẾP

a={10, 3, 99, 27, 45, 68, 17, 31, 55, 82}

Sắp xếp mảng theo thứ tự tăng dần.

a)

- Bubble Sort

```
public static void Bubblesort(int a[],int n){
      for(int i=0;i<n;i++){
         for(int j=i;j<n-1;j++){
            if(a[j]>a[j+1]){
               int temp = a[j];
               a[j] = a[j+1];
               a[j+1]= temp;
            }
         }
      }
   }
```

Với Bubble Sort với 2 tham số truyền vào là một mảng và kích thước mảng , ta tạo 2 vòng lặp for để so sánh 2 phần tử liền kề nhau trong mảng nếu phần tử trước lớn hơn phần tử sau thì đổi vị trí 2 phần tử này, nếu không thí tiếp tục vòng lặp:

a={10, 3, 99, 27, 45, 68, 17, 31, 55, 82}

-a[0]=10, a[1]= 3, a[0] >a[1]⇒đổi vị trí

a={3,10,99,27,45,68,17,31,55,82}

-a[1]=10, a[2]=99, a[1]<a[2]⇒giữ nguyên

-a[2]=99, a[3]=27,a[2]>a[3]⇒đổi vị trí
a={3,10,27,99,45,68,17,31,55,82}

-a[3]=99, a[4]=45,a[3]>a[4]⇒đổi vị trí
a={3,10,27,45,99,68,17,31,55,82}

-a[4]=99, a[5]=68,a[4]>a[5]⇒đổi vị trí
a={3,10,27,45,68,99,17,31,55,82}

-a[5]=99, a[6]=17,a[5]>a[6]⇒đổi vị trí
a={3,10,27,45,68,17,99,31,55,82}

-a[6]=99, a[7]=31,a[6]>a[7]⇒đổi vị trí
a={3,10,27,45,68,17,31,99,55,82}

-a[7]=99, a[8]=55,a[7]>a[8]⇒đổi vị trí
a={3,10,27,45,68,17,31,55,99,82}

-a[8]=99, a[9]=82,a[8]>a[9]⇒đổi vị trí
a={3,10,27,45,68,17,31,55,82,99}

Kết thúc vòng lắp for với biến chạy j, ta tiếp tục với vòng lặp for biến chạy i. Vậy khi kết thúc vòng lặp ta sẽ được kết quả là
a={3, 10, 17, 27, 31, 45, 55, 68, 82, 99}

- Selection Sort

```
public static void Selectionsort(int a[],int n){
        for(int i=0;i<n-1;i++){
            int min = i;
            for(int j=i+1;j<n;j++){
                if(a[min] > a[j])
```

```
                min = j;
            }
            int temp = a[min];
            a[min] = a[i];
            a[i] = temp;
        }
    }
```

Với Selection Sort với 2 tham số truyền vào là một mảng và kích thước mảng , ban đầu ta gán cho phần tử đầu tiên của mảng là min rồi ta so sánh với các phần tử trong mảng.Nếu có phần tử nào nhỏ hơn min thì ta gán nó bằng min.kết thúc vòng lặp for biến chạy j ta đổi chỗ min với phần tử i đang xét.
a={10, 3, 99, 27, 45, 68, 17, 31, 55, 82}

- min=i=0,j=i+1=1
-a[min]=a[0]=10>a[j]=a[1]=3⇒min =1
-a[min]=3,a[2]=99,a[min]<a[2]⇒min không đổi
-a[min]<a[3]⇒min không đổi
-a[min]<a[4]⇒min không đổi
-a[min]<a[5]⇒min không đổi
-a[min]<a[6]⇒min không đổi
-a[min]<a[7]⇒min không đổi
-a[min]<a[8]⇒min không đổi
-a[min]<a[9]⇒min không đổi
-kết thúc vòng lặp biến chạy đổi vị trí min với a[i]
a={3,10,99,27,45,68,17,31,55,82}
-tiếp tục vòng lặp i=1, min=1, mảng không đổi
-i=2,min=2
a={3,10,17,27,45,68,99,31,55,82}

-i=3,min=3, mảng không đổi

-i=4,min=4
a={3,10,17,27,31,68,99,45,55,82}

-i=5,min=5
a={3,10,17,27,31,45,99,68,55,82}

-i=6,min=6
a={3,10,17,27,31,45,55,68,99,82}

-i=7,min=7
a={3,10,17,27,31,45,55,68,82,99}
-i=8,min=8, mảng không đổi

Vậy khi kết thúc vòng lặp ta sẽ được kết quả là
a={3, 10, 17, 27, 31, 45, 55, 68, 82, 99}

b)

- Merge Sort

```
void merge(int arr[], int l, int m, int r) {
        // Tìm kích thước của 2 mảng con để merged
        int n1 = m - l + 1;
        int n2 = r - m;
        // Tạo mảng tạm
        int L[] = new int[n1];
        int R[] = new int[n2];
        // Copy dữ liệu vào mảng tạm
        for (int i = 0; i < n1; ++i)
           L[i] = arr[l + i];
        for (int j = 0; j < n2; ++j)
           R[j] = arr[m + 1 + j];
```

```
    // Merge các mảng tạm lại
    // Chỉ mục ban đầu của 2 mảng con
    int i = 0, j = 0;
    // Chỉ mục ban đầu của mảng phụ được hợp nhất
    int k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    // Sao chép các phần tử còn lại của L[] nếu có
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    // Sao chép các phần tử còn lại của R[] nếu có
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
```

Với Merge sort với 4 tham số truyền vào là một mảng ,l là index đầu mảng,m là index giữa mảng và r là index cuối mảng, ta sẽ tách mảng ban đầu thành các mảng

con tới khi các mảng con có kích thước là 1 thì ta sẽ gộp các mảng con lại với thứ tự tăng dần.

a={10, 3, 99, 27, 45, 68, 17, 31, 55, 82}

-tách a thành a1={10,3,99,27,45} và a2={68,17,31,55,82}

-tách a1 thành a3={10,3,99} và a4={27,45}

-tách a2 thành a5={68,17,31} và a6={55,82}

-tách a3 thành b1={10,3} và b2={99}

-tách a4 thành c1={27} và c2={45}

-tách a5 thành d1={68,17} và d2={31}

-tách a6 thành e1={55} và e2={82}

-tách b1 thành f1={10} và f2={3}

-tách d1 thành g1={68} và g2={17}

-gộp f1 và f2 ta được mảng mới có thứ tự m={3,10}, gộp t với b2 ta được mảng mới m={3,10,99}

-gộp c1 và c2 ta được mảng mới n={27,45}

-gộp g1 và g2 ta được mảng mới o={17,68}, gộp o với d2 ta được mảng mới o={17,31,68}

-gộp e1 và e2 ta được mảng mới p={55,82}

-gộp m và n ta được mảng mới t={3,10,27,45,99}

-gộp o và p ta được mảng mới u={17,31,55,68,82}

Vậy khi gộp t và u ta được mảng mới có thứ tự sắp xếp tăng dần

v={3, 10, 17, 27, 31, 45, 55, 68, 82, 99}

CÂU 3: STACK

a)

-Các biểu thức toán học thường được biểu diễn ở dạng trung tố (Infix), nhưng để có thể tính toán được các biểu thức đó trên máy tính thì cần biểu diễn ở dạng hậu tố (Postfix).

-Để chuyển từ Infix sang Postfix ta sử dụng stack:

-Ta duyệt các kí tự trong chuối Infix truyền vào :

-Nếu kí tự đang xét là toán hạng thì ta đưa vào chuỗi Postfix

-Nếu kí tự đang xét là toán tử hay các dấu ngoặc mở thì ta đưa nó vào stack.Tuy nhiên ,các toán tử * hoặc / sẽ có độ ưu tiên cao hơn + và - ,nên khi xét đến + hoặc - mà đỉnh stack là * hoặc / thì ta sẽ pop() nó ra thêm vào chuối Postfix trước, sau đó đưa toán tử đang xét vào stack.

-Khi xét hết các kí tự trong chuỗi Infix mà stack chưa rỗng thì ta pop hết ra thêm vào chuỗi Postfix đến khi stack rỗng.

-vd: Infix="[(9-7)*2+3*4-8]/2"

Infix	Stack	Postfix
[	[	
(	[(	
9	[(	9
-	[(-	9
7	[(-	97
)	[(	97-
	[	97-
*	[*	97-
2	[*	97-2
+	[	97-2*
	[+	97-2*
3	[+	97-2*3
*	[+*	97-2*3
4	[+*	97-2*34
-	[+	97-2*34*
	[+-	97-2*34*
8	[+-	97-2*34*8
]	[+	97-2*34*8-

	[	97-2*34*8-+
		97-2*34*8-+
/	/	97-2*34*8-+
2	/	97-2*34*8-+2
		97-2*34*8-+2/

Vậy chuỗi Postfix chuyển từ Infix là : 97-2*34*8-+2/

b)

Để tính toán từ biểu thức từ chuỗi Postfix ta cũng sử dụng stack:

-Ta duyệt các kí tự trong chuỗi Postfix từ trái qua phải

-Nếu kí tự đang xét là toán hạng ta sẽ thêm vào stack

-Nếu kí tự đang xét là toán tử ta sẽ pop() 2 giá trị trên đỉnh của stack ra thực hiện phép toán. Sau khi tính xong ta lại đẩy kết quả phép toán trở lại stack

-Khi duyệt xong ta sẽ pop() từ stack ra và giá trị đó là kết quả của biểu thức cần tính.

-vd: Postfix= “97-2*34*8-+2/”

xét 9:

-Stack.push(9)

-Stack:9

Stack	9		

xét 7:

-Stack.push(7)

-Stack :9,7

Stack	9	7	

xét -:

-a=stack.pop()=7

-b=stack.pop()=9

-c=b-a=9-7=2

-Stack.push(c)

-Stack:2

Stack	2		

xét 2:

-Stack.push(2)

-Stack :2,2

Stack	2	2	

xét *:

-a=Stack.pop()=2

-b=Stack.pop()=2

-c=a*b=2*2=4

-Stack.push(c)

-Stack:4

Stack	4		

xét 3:

-Stack.push(3)

-Stack.4,3

Stack	4	3	

xét 4:

-Stack.push(4)

-Stack :4,3,4

Stack	4	3	4

xét *:

-a=Stack.pop()=4

-b=Stack.pop()=3

-c=a*b=4*3=12

-Stack.push(c)

-Stack:4,12

Stack	4	12	

xét 8:

-Stack.push(8)

-Stack:4,12,8

Stack	4	12	8

xét -:

-a=Stack.pop()

-b=Stack.pop()

-c=b-a=12-8=4

-Stack.push(c)

-Stack:4,4

Stack	4	4	

xét +:

-a=Stack.pop()

-b=Stack.pop()

-c=a+b=4+4=8

-Stack,push(c)

-Stack :8

Stack	8		

xét 2:

-Stack.push(2)

-Stack :8,2

Stack	8	2	

xét /:

-a=Stack.pop()

-b=Stack.pop()

-c=b/a=8/2=4

-Stack.push(c)

Stack:4

Stack	4		

Vậy kết quả của biểu thức tính từ chuỗi Postfix “97-2*34*8-+2/” là: 4

CÂU 4: CẤU TRÚC DỮ LIỆU DANH SÁCH VÀ CÂY

- Lí thuyết

a) Thêm một sinh viên mới

Mảng	Linked List	AVL
-Khi biết kích thước của mảng ta có thể thêm một sinh viên mới vào vị trí bất kì với điều kiện là mảng chưa đầy. -Trường hợp tối ưu nhất là thêm một sinh viên vào cuối mảng,lúc này độ phức tạp của thuật toán là: O(1)	-Với các hàm addFirst, addLast,addIndex ta có thể thêm sinh viên mới vào vị trí bất kì và có thể mở rộng kích thước. -Tuy nhiên trừ trường hợp addFirst, những trường hợp còn lại ta phải duyệt các để đến vị trí cần thêm, trường hợp tệ nhất là addLast sẽ có độ phức tạp là: O(n)	-ta có thể thêm một sinh viên mới dựa vào điều kiện cân bằng của cây và điều kiện của các Node. - Các Node đã sắp xếp theo một trật tự nhất định nên khi duyệt ta sẽ không duyệt tất cả các phần tử trong cây, độ phức tạp là: O(logn)

b) Xóa một sinh viên

Mảng	Linked List	AVL
-Để xóa một sinh viên khỏi mảng, ta phải duyệt mảng đến vị trí cần xóa. -Sau khi xóa, ta phải cập nhật lại mảng và kích thước. Trường hợp tệ nhất độ phức tạp là :O(n)	-Giống với thêm một sinh viên ta cũng có các hàm xóa như removeLast, removeAfter, trong các hàm này ta phải duyệt để đến được vị trí cần xóa,trường hợp tệ nhất là removeLast có độ phức tạp là: O(n)	-Để xóa một sinh viên (Node) ta cũng phải duyệt cây đến vị trí cần xóa, hoán đổi vị trí node đó với node nhỏ nhất nhánh bên phải của node đó.Hoán đổi xong ta xóa node, sau khi xóa ta phải cập nhật lại chiều cao các node và kiểm tra điều kiện cân bằng. Độ phức tạp là : O(logn)

c) Tìm kiếm sinh viên

Mảng	Linked List	AVL
-Để tìm kiếm 1 sinh viên ta phải duyệt các phần tử trong mảng và kiểm tra.Độ phức tạp là: O(n)	-Để tìm kiếm 1sinh viên ta sẽ phải duyệt các Node và kiểm tra. -Độ phức tạp là: O(n)	-Để tìm kiếm 1 sinh viên (Node) trong cây, ta sẽ phải duyệt cây và kiểm tra các Node. -Độ phức tạp là: O(logn)

- Thực hành
  - Sinh viên

```
public class Sinhvien {
    private String id;
    private String name;
    private double grade;
    private String address;
    private String sdt;
    public Sinhvien(String id, String name,double grade,String
address,String sdt){
        this.id = id;
        this.name= name;
        this.grade = grade;
        this.address = address;
        this.sdt= sdt;
    }
    public String getId(){
        return this.id;
    }
    public String getName(){
        return this.name;
    }
    public double getGrade(){
        return this.grade;
    }
    public String getAddress(){
        return this.address;
    }
    public String getSdt(){
        return this.sdt;
    }
    public String toString(){
        return this.id+" "+this.name+" "+this.grade+"
"+this.address+" "+this.sdt;
    }
}
```

- Linked List:

```
public class Node {
    private Sinhvien value;
    private Node pNext;
    public Node(Sinhvien s){
        this.value = s;
        this.pNext = null;
    }
    public Node(Sinhvien s, Node n){
        this.value = s;
        this.pNext = n;
    }
    public Sinhvien getValue(){
        return this.value;
    }
    public void setValue(Sinhvien s){
        this.value = s;
    }
    public Node getNext(){
        return this.pNext;
    }
    public void setNext(Node n){
        this.pNext = n;
    }
}

public class LinkedList {
    private Node head;

    public LinkedList(){
        head=null;
    }
    public Node getHead(){
        return this.head;
    }
    public void addLast(Sinhvien s){
```

```
        if(head==null){
            head = new Node(s,head);
        }else{
            Node tmp = head;
            Node del = new Node(s,null);
            while(tmp.getNext() != null){
                tmp = tmp.getNext();
            }
            tmp.setNext(del);
            del.setNext(null);
        }
    }
    public boolean removeAt(int index){
        if(index==0) head= head.getNext();
        int i=0;
        Node tmp = head;
        while(i != index){
            tmp = tmp.getNext();
            if(tmp==null) return false;
            i++;
        }
        if(head == null) return false;
        else{
            int dem=0;
            while(dem+1 !=index && tmp != null){
                tmp=tmp.getNext();
                dem++;
            }
            Node del = tmp.getNext();
            tmp.setNext(del.getNext());
            del.setNext(null);
        }
        return true;
    }
    public boolean search(String id){
        Node tmp = head;
        while(tmp!=null){
            if(tmp.getValue().getId().equals(id)) return true;
```

```
        }
        return false;
    }
    public void print(){
        if(head == null) System.out.println("List is empty");
        Node temp = head.getNext();
        System.out.println("List: "+head.getValue());
        while(temp != null){
            System.out.println(" -> "+temp.getValue().toString());
            temp = temp.getNext();
        }
        System.out.println();
    }
}

import java.util.*;
public class Main {
    public static void main(String args[]){
        Sinhvien s1 = new Sinhvien("52000674","Khanh",8.1,"Thai
Binh","0128672");
        Sinhvien s2 = new Sinhvien("52000612","Kiet",8.0,"Da
Lat","2582568");
        Sinhvien s3 = new Sinhvien("42005679","Van", 6.8,
"Hue","0306022");
        Sinhvien s4 = new Sinhvien("H1900144","Duy",8.3 , "Gia
Lai","03060267");
        Sinhvien s5 = new Sinhvien("H1800389","Sang",9.2 , "Kien
Giang","0695858");
        Sinhvien s6 = new Sinhvien("72001366","Long", 8.7, "Ca
Mau","0478748");
        Sinhvien s7 = new Sinhvien("62000149","Kien", 6.7, "Binh
Thuan","0101406");
        Sinhvien s8 = new Sinhvien("11910343","Hoa",7.8 ,"Lam
Dong","0366746");
        Sinhvien s9 = new Sinhvien("H1800072","Hoa", 7.3,"Bao
Loc","0123986");
        Sinhvien s10 = new Sinhvien("H1900346","Khiem", 8.8,"Phu
Yen","0555654");
```

```
        Sinhvien s11 = new Sinhvien("32001013","Hanh",5.6 ,"Quang
Nam","0674234");
        Sinhvien s12 = new Sinhvien("11900345","Tran",7.7 ,"Tra
Vinh","0664573");
        Sinhvien s13 = new Sinhvien("H5200777","Huynh",8.9 ,"Dong
Thap","0989879");
        Sinhvien s14 = new Sinhvien("41901231","Quang",8.1 ,"Da
Lat","0231441");
        Sinhvien s15 = new Sinhvien("72101344","Phat",6.9 ,"Bac
Ninh","0567431");

        LinkedList list = new LinkedList();
        list.addLast(s1);
        list.addLast(s2);
        list.addLast(s3);
        list.addLast(s4);
        list.addLast(s5);
        list.addLast(s6);
        list.addLast(s7);
        list.addLast(s8);
        list.addLast(s9);
        list.addLast(s10);
        list.addLast(s11);
        list.addLast(s12);
        list.addLast(s13);
        list.addLast(s14);
        list.addLast(s15);
        list.print();
        list.removeAt(7);
        list.removeAt(4);
        System.out.println("Sau khi xoa: ");
        list.print();
        System.out.println("Ket qua tim sinh vien co id la 52000674:
"+list.search("52000674"));

    }
}
```

- AVL:

```
public class Node {
    private Sinhvien value;
    private Node left;
    private Node right;
    private int height;

    public Node(Sinhvien s){
        this.value = s;
        this.left =this.right = null;
        this.height=0;
    }
    public Sinhvien getValue(){
        return this.value;
    }
    public void setValue(Sinhvien s){
        this.value= s;
    }
    public Node getLeft(){
        return this.left;
    }
    public void setLeft(Node l){
        this.left=l;
    }
    public Node getRight(){
        return this.right;
    }
    public void setRight(Node r){
        this.right=r;
    }
    public int getHeight(){
        return this.height;
    }
    public void setHeight(int h){
        this.height=h;
    }
}
```

```
public class AVL {
    private Node root;
    public AVL(){
        this.root=null;
    }
    public Node getRoot(){
        return root;
    }
    public void setRoot(Node root){
        this.root=root;
    }
    public int height(Node node){
        if(node ==null){
            return -1;
        }
        return node.getHeight();
    }
    private int checkBalance(Node x){
        return height(x.getLeft())-height(x.getRight());
    }
    private Node balance(Node x){

        return x;
    }
    private Node rotateLeft(Node x){
        if(x==null) return null;
        Node y=x.getRight();
        x.setRight(y.getLeft());
        y.setLeft(x);

x.setHeight(1+Math.max(height(x.getLeft()),height(x.getRight())));

y.setHeight(1+Math.max(height(y.getLeft()),height(y.getRight())));
        return y;
    }
    private Node rotateRight(Node x){
        if(x==null) return null;
        Node y=x.getLeft();
```

```
        x.setLeft(y.getRight());
        y.setRight(x);

x.setHeight(1+Math.max(height(x.getLeft()),height(x.getRight())));

y.setHeight(1+Math.max(height(y.getLeft()),height(y.getRight())));
        return y;
    }
    private int priority(String s1,String s2){
        char[] ch1 = s1.toCharArray();
        char[] ch2 = s2.toCharArray();
        for(int i=0; i<ch1.length;i++){
            int a= ch1[i];
            int b= ch2[i];
            if(a>b) return 1;
        }
        return 0;
    }
    private Node Insert(Node node, Sinhvien s){
        if(node==null){
            node = new Node(s);
            return node;
        }
        int a= s.getId().compareTo(node.getValue().getId());
        if(a==0) return node;
        if(a<0){
            node.setLeft(Insert(node.getLeft(),s));
        }else{
            node.setRight(Insert(node.getRight(),s));
        }
        node.setHeight(1+Math.max(height(node.getLeft()),
height(node.getRight())));

        int balance = checkBalance(node);
        if(balance > 1){
            int b =
s.getId().compareTo(node.getLeft().getValue().getId());
            if(b >0){
```

```
                node.setLeft(rotateLeft(node.getLeft()));
            }
            return rotateRight(node);
        }else if(balance < -1){
            int c =
s.getId().compareTo(node.getRight().getValue().getId());
            if(c < 0){
                node.setRight(rotateRight(node.getRight()));
            }
            return rotateLeft(node);
        }
        return node;
    }
    public void insert(Sinhvien s){
        this.root=Insert(this.root, s);
    }
    private Node search(Node x, String id){
        if(x==null){
            return null;
        }
        int a = id.compareTo(x.getValue().getId());
        if(a<0){
            return search(x.getLeft(),id);
        }else if(a>0){
            return search(x.getRight(),id);
        }else{
            return x;
        }
    }
    public String search(String id){
        return search(this.root,id).getValue().toString();
    }
    private void LNR(Node node){
        if(node!= null){
            LNR(node.getLeft());
            System.out.println(node.getValue().toString());
            LNR(node.getRight());
        }
```

```
    }
    public void LNR(){
        LNR(this.root);
    }
    private Node min(Node x){
        if(x.getLeft()==null)
            return x;
        else
            return min(x.getLeft());
    }
    private Node deletemin(Node x){
        if(x.getLeft()==null)
            return x.getRight();
        x.setLeft(deletemin(x.getLeft()));
        return x;
    }
    private Node delete(Node node,String id){
        if(node ==null)
            return null;
        int a=id.compareTo(node.getValue().getId());
        if(a<0)
            node.setLeft(delete(node.getLeft(),id));
        else if(a>0)
            node.setRight(delete(node.getRight(),id));
        else{
            if(node.getLeft()==null )
                return node.getRight();
            if(node.getRight()==null)
                return node.getLeft();

            node.setValue(min(node.getRight()).getValue());
            node.setRight(deletemin(node.getRight()));
        }
        if(node==null) return null;


node.setHeight(1+Math.max(height(node.getLeft()),height(node.getRight
()));
```

```
        int balance = checkBalance(node);
        if(balance > 1 && checkBalance(node.getLeft()) >= 0)
            return rotateRight(node);

        if(balance > 1 && checkBalance(node.getLeft()) < 0){
            node.setLeft(rotateLeft(node.getLeft()));
            return rotateRight(node);
        }

        if(balance < -1 && checkBalance(node.getRight()) <= 0)
            return rotateLeft(node);

        if(balance < -1 && checkBalance(node.getRight()) > 0){
            node.setRight(rotateRight(node.getRight()));
            return rotateLeft(node);
        }

        return node;
    }
    public void delete(String id){
        delete(this.root,id);
    }
}

public class Main {
    public static void main(String args[]){
        AVL tree = new AVL();
        Sinhvien s1 = new Sinhvien("52000674","Khanh",8.1,"Thai
Binh","0128672");
        Sinhvien s2 = new Sinhvien("52000612","Kiet",8.0,"Da
Lat","2582568");
        Sinhvien s3 = new Sinhvien("42005679","Van", 6.8,
"Hue","0306022");
        Sinhvien s4 = new Sinhvien("H1900144","Duy",8.3 , "Gia
Lai","03060267");
        Sinhvien s5 = new Sinhvien("H1800389","Sang",9.2 , "Kien
Giang","0695858");
```

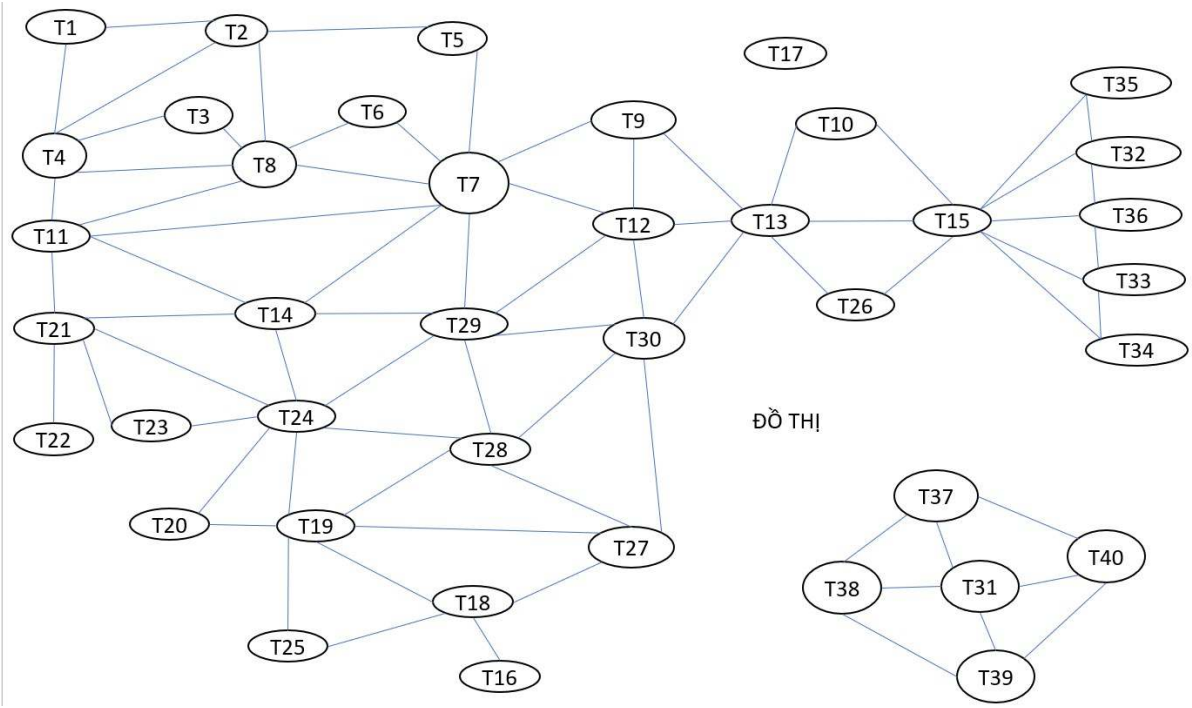
```
        Sinhvien s6 = new Sinhvien("72001366","Long", 8.7, "Ca
Mau","0478748");
        Sinhvien s7 = new Sinhvien("62000149","Kien", 6.7, "Binh
Thuan","0101406");
        Sinhvien s8 = new Sinhvien("11910343","Hoa",7.8 ,"Lam
Dong","0366746");
        Sinhvien s9 = new Sinhvien("H1800072","Hoa", 7.3,"Bao
Loc","0123986");
        Sinhvien s10 = new Sinhvien("H1900346","Khiem", 8.8,"Phu
Yen","0555654");
        Sinhvien s11 = new Sinhvien("32001013","Hanh",5.6 ,"Quang
Nam","0674234");
        Sinhvien s12 = new Sinhvien("11900345","Tran",7.7 ,"Tra
Vinh","0664573");
        Sinhvien s13 = new Sinhvien("H5200777","Huynh",8.9 ,"Dong
Thap","0989879");
        Sinhvien s14 = new Sinhvien("41901231","Quang",8.1 ,"Da
Lat","0231441");
        Sinhvien s15 = new Sinhvien("72101344","Phat",6.9 ,"Bac
Ninh","0567431");
        tree.insert(s1);
        tree.insert(s2);
        tree.insert(s3);
        tree.insert(s4);
        tree.insert(s5);
        tree.insert(s6);
        tree.insert(s7);
        tree.insert(s8);
        tree.insert(s9);
        tree.insert(s10);
        tree.insert(s11);
        tree.insert(s12);
        tree.insert(s13);
        tree.insert(s14);
        tree.insert(s15);
        tree.LNR();
        tree.delete("42005679");
        tree.delete("62000149");
```

```
        tree.delete("72001366");
        tree.delete("H1900346");
        System.out.println("Sau khi xoa: ");
        tree.LNR();
        System.out.println("ket qua search id 52000674:
"+tree.search("52000674"));
    }
}
```

CÂU 5: ĐỒ THỊ

- Lí thuyết

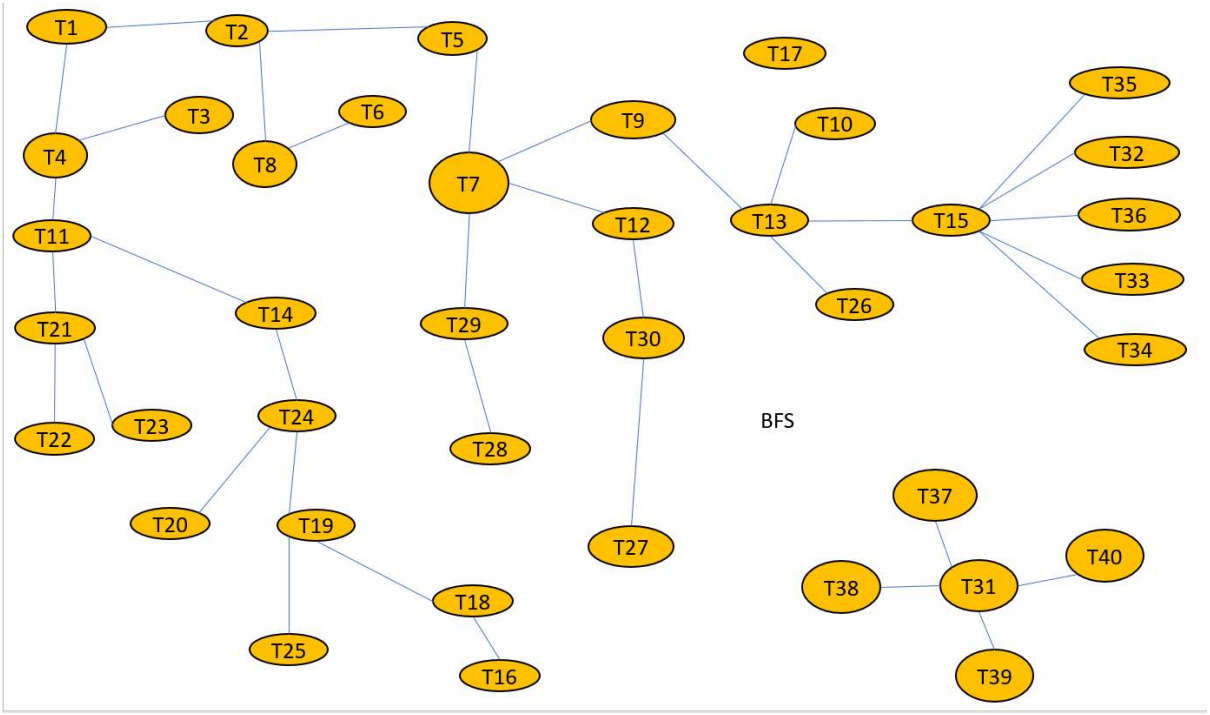
a) Đồ thị:



HÌNH 5.1

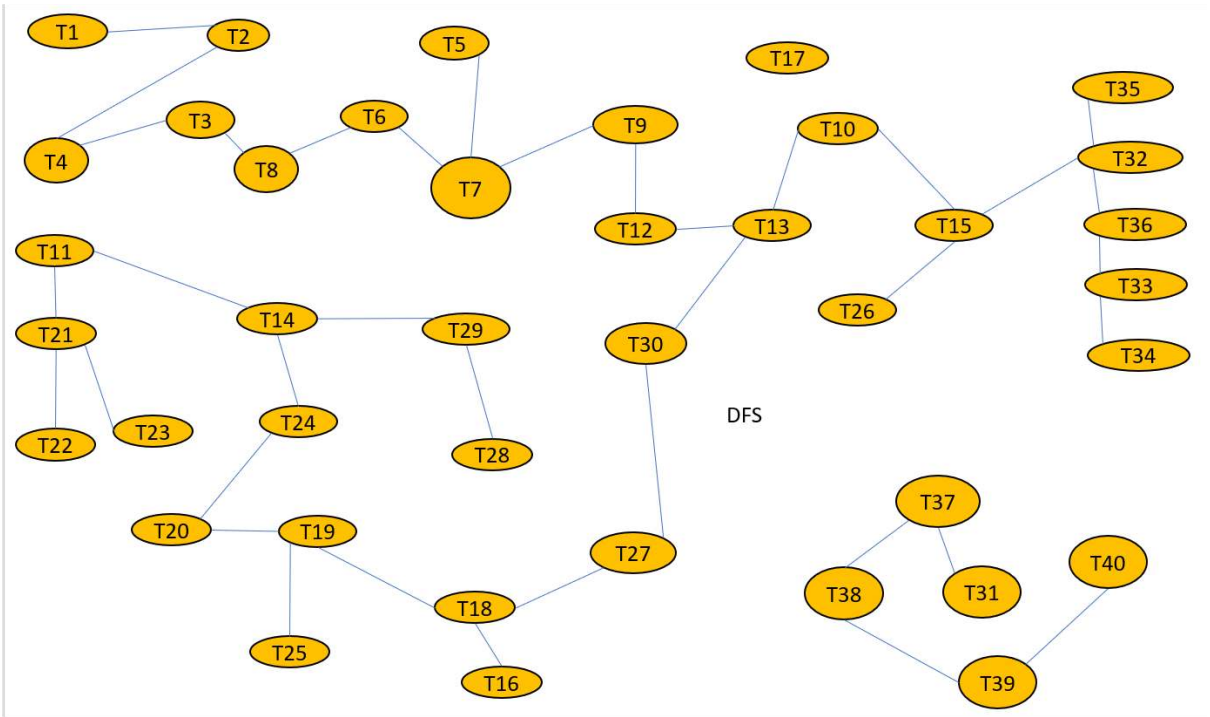
b) Bắt đầu từ T1, ưu tiên nhỏ trước:

- BFS: T1, T2, T4, T5, T8, T3, T11, T7, T6, T14, T21, T9, T12, T29, T24, T22, T23, T13, T30, T28, T19, T20, T10, T15, T26, T27, T18, T25, T32, T33, T34, T35, T36, T16, T17, T31, T37, T38, T39, T40



HÌNH 5.2

- DFS: T1, T2, T4, T3, T8, T6, T7, T5, T9, T12, T13, T10, T15, T26, T32, T35, T36, T33, T34, T30, T27, T18, T16, T19, T20, T24, T14, T11, T21, T22, T23, T29, T28, T25, T17, T31, T37, T38, T39, T40



HÌNH 3.3

- Thực hành

a) Đọc file,vẽ đồ thị bằng danh sach cạnh

```
public Graph(String peoplepath,String linkpath){
        edges = new ArrayList<Link>();
        list = new ArrayList<People>();
        readPeopleFile(peoplepath);
        buildGraph(linkpath);
    }
    public void printGraph(){
        for(int i=0;i<edges.size();i++){
            System.out.println(edges.get(i).toString());
        }
    }
    public boolean readPeopleFile(String filepath){
        try{
            File f = new File(filepath);
            Scanner sc = new Scanner(f);
            while(sc.hasNextLine()){
                String s = sc.nextLine();
                String[] bien = s.split(",");
                People p = new
People(bien[0],bien[1],Integer.parseInt(bien[2]),bien[3],bien[4]);
                list.add(p);
            }
            sc.close();
        }catch(FileNotFoundException e){
            System.out.println("File not found");
            return false;
        }
        return true;
    }
    public void addEdge(People p1,People p2){
        edges.add(new Link(p1,p2));
    }
    public boolean buildGraph(String filepath){
        try{
            File f= new File(filepath);
            Scanner sc = new Scanner(f);
            while(sc.hasNextLine()){
```

```
                String s = sc.nextLine();
                String[] bien = s.split(",");
                People p1 = new People();
                People p2 = new People();
                for(People a : list){
                    if(a.getId().equals(bien[0]))
                        p1=a;
                }
                for(People b : list){
                    if(b.getId().equals(bien[1]))
                        p2=b;
                }
                addEdge(p1,p2);
            }
            sc.close();
        }catch(FileNotFoundException e){
            System.out.println("File not found");
            return false;
        }
        return true;
    }
```

b) Phương thức trả về danh sách bạn bè của một người dùng truyền vào

```
public ArrayList<People> listFriends(People p){
        ArrayList<People> list1= new ArrayList<People>();
        for(Link l : edges){
            if(l.getHead().getId().equals(p.getId())){
                list1.add(l.getLast());
            }
        }
        //in ra man hinh
        System.out.println("Danh sach ban cua "+p.getId()+":");
        for(People p1 : list1){
            System.out.println(p1.toString());
        }
        return list1;
    }
```

c) Phương thức xác định hai tài khoản truyền vào có bao nhiêu bạn chung

```
public int sameFriends(People p1,People p2){
        int count=0;
        ArrayList<People> list1= new ArrayList<People>();
        ArrayList<People> list2= new ArrayList<People>();
        list1= listFriends(p1);
        list2= listFriends(p2);
        for(People n : list1){
            for(People m : list2){
                if(n.getId().equals(m.getId()))
                    count++;
            }
        }
        //in ra man hinh
        System.out.println("So ban chung cua "+p1.getId()+" va
"+p2.getId()+" la: "+count);
        return count;
    }
```

-

TÀI LIỆU THAM KHẢO

[1] https://drive.google.com/drive/folders/1xp_vdYTNOStOrh2ONe-tM9yyMYmrdZ_9

[2] Introduction to Algorithm,3rd edition – ThomasH.Cormen

[3] The Art of Computer Programming, 1st Edition- Donald E.Knuth

[4] Data Structure and Algorithm in java, 2nd Edition- Robert Lafore

[5] https://niithanoi.edu.vn/merge-sort.html

[6] https://ichi.pro/vi/tim-de-quy-so-fibonacci-thu-n-94470777113526

[7] https://www.stdio.vn/giai-thuat-lap-trinh/ung-dung-stack-bieu-thuc-hau-to-postfix-uqu1Q

[8] https://tek4.vn/khoa-hoc/cau-truc-du-lieu-va-giai-thuat/cay-avl-va-cac-thao-tac

[9] Algorithm, 4th Edition – Robert Sedgewick, Kevin Wayne

[10] Data Structures and Algorithms Made Easy – Narasimha Karumanchi

[11] Algorithms in C, 3rd Edition – Robert Sedgewick