12/17/2018

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT DOCUMENT

Viet SacLo
08DHTH1

❖ - Phần 1 (4 điểm) gồm Chương 1 và 2: đánh giá độ phức tạp giải thuật, bài toán về giải thuật sắp xếp và tìm kiếm.

1. Giải thuật tìm kiếm nhị phân

```
public int binarySearch(int left,int right,int x){
    int mid;
    do{
        mid = (left+right)/2;
        if (arr[mid]==x) return mid;
        if (arr[mid]>x) right = mid-1;
        else left = mid+1;
    }while(left<=right);
    return -1;
}
```

2. Giải thuật heap sort

```
private void shift(int left,int right){
    int i=left,j=2*i,temp = arr[i];
    while(j<=right){
        if (j<right&&arr[j]<arr[j+1]) j++;
        if (temp>=arr[j]) break;
        arr[i] = arr[j];
        arr[j] = temp;
        i=j;
        j=2*i;
    }
}
private void createHeap(){
    for (int i=size/2;i>=0;i--)
        shift(i,size-1);
}
public void heapSort(){
    createHeap();
    int temp = arr[0];
    arr[0] = arr[size-1];
    arr[size-1] = temp;
    for (int i=size-2;i>0;i--){
        shift(0,i);
        temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
    }
}
```

3. Giải thuật Quick Sort

   Trong Slide

4. Hàm nhập danh sách n sinh viên vào mảng

```
public void nhapDSSV(){
    input = new Scanner(System.in);
    System.out.printf("Nhap so luong SV: ");
    this.size = input.nextInt();
    ds = new sinhVien[size];
    for (int i=0;i<this.size;i++){
        System.out.println("Nhap SV thu: "+(i+1));
        ds[i] = new sinhVien();
        ds[i].nhapSV();
        System.out.println("======================");
    }
}
```

5. Hàm sắp xếp danh sách sinh viên tang dần theo tên.

```
public void quicSort(int left,int right){
    int i = left,j = right,mid = (left+right)/2;
    do{
        int ss1 = ds[i].getTenSv().compareTo(ds[mid].getTenSv());
        while (ss1<0){
            i++;
            ss1 = ds[i].getTenSv().compareTo(ds[mid].getTenSv());
        }
        int ss2 = ds[j].getTenSv().compareTo(ds[mid].getTenSv());
        while (ss2>0){
            j--;
            ss2 = ds[j].getTenSv().compareTo(ds[mid].getTenSv());
        }
        if (i<j){
            sinhVien tam = ds[i];
            ds[i] = ds[j];
            ds[j] = tam;
        }
        i++;
        j--;
    }while(i<mid&&j>mid);
    if (left<j) quicSort(left,j);
    if (right>i) quicSort(i,right);
}
```

6. Hàm sắp xếp danh sách sinh viên tang dần theo điểm trung bình

```
public void shift(int dau,int cuoi){
    int i=dau,j = 2*i;
    sinhVien tam = ds[i];
    while(j<=cuoi){
        if (j<cuoi&&ds[j].getDiemTB()<ds[j+1].getDiemTB()) j++;
        if (ds[i].getDiemTB()>ds[j].getDiemTB()) break;
        ds[i] = ds[j];
        ds[j] = tam;
        i=j;
        j=2*i;
    }
}

public void taoHeap(){
    for (int i=size/2;i>=0;i--){
        shift(i, size-1);
    }
}

public void heapSort(){
    taoHeap();
    sinhVien tam = ds[0];
    ds[0] = ds[size-1];
    ds[size-1]= tam;
    for (int i = size-2;i>0;i--){
        shift(0, i);
        tam = ds[0];
        ds[0] = ds[i];
        ds[i] = tam;
    }
}
```

❖ Phần 2 (3 điểm) gồm chương 3: bài toán có sử dụng danh sách liên kết, stack, queue

1. Hàm insertAfterNode

```
public void them1NodeVaoSauNode(Node p, Node value) {
    if (value == null) {
        return;
    }
    if (header == null) {
        header = tailer = value;
        return;
    }
    if (p == null) {
        value.setNext(header);
        header = value;
    } else {
        value.setNext(p.getNext());
        p.setNext(value);
        if (p == tailer) {
            tailer = value;
        }
    }
}
```

2. Hàm deleteAfterNode

```
public Node xoa1NodeSauNode(Node q) {
    Node p = null;
    if (header == null) {
        return p;
    }
    if (q == null) {
        p = header;
        header = p.getNext();
        p.setNext(null);
    } else {
        p = q.getNext();
        if (p != null) {
            q.setNext(p.getNext());
            if (p == tailer) {
                tailer = q;
            }
            p.setNext(null);
        }
    }
    return p;
}
```

3. Các hàm trong simple List

```
// ham tim node
public Node sercher(int x) {
    for (Node i = header; i != null; i = i.getNext()) {
        if (i.getValue() == x) {
            return i;
        }
    }
    return null;
}

// ham them 1 ndoe vao dau danh sach lien ket
public void them1NodeVaoDauDanhSach(Node value) {
    them1NodeVaoSauNode(null, value);
}

// ham xoa 1 node vao cuoi danh sach lien ket
public Node xoa1Node_CuoiDS() {
    Node pre = null;
    Node tam = header;
    while (tam != tailer) {
        pre = tam;
        tam = tam.getNext();
    }
    return xoa1NodeSauNode(pre);
}

// ham them 1 node o cuoi danh sach
public void them1NodeCuoiDS(Node value) {
    them1NodeVaoSauNode(tailer, value);
}

// ham xuat tat ca node trong danh sach lien ket
public void xuatAll() {
    for (Node tam = header; tam != null; tam = tam.getNext()) {
        System.out.print(tam.getValue() + "\t");
    }
}

// ham xoa 1 node o dau danh sach;
public void xoaCaiDau() {
    xoa1NodeSauNode(null);
}
```

```
    // ham them so luong node lon
    public void them_N_Node() {
        Scanner input = new Scanner(System.in);
        System.out.printf("Nhap So Luong Node: ");
        int n = input.nextInt();
        for (int i = 0; i < n; i++) {
            System.out.println("Nhap Node thu: " + (i + 1));
            Node tam = new Node(input.nextInt());
            them1NodeCuoiDS(tam);
        }
    }

    // ham xoa phan tu chan
    public void xoaChan() {
        Node p = header;
        if (p == null) {
            return;
        }
        if (p.getValue() % 2 == 0) {
            xoa1NodeSauNode(null);
        } else {
            for (Node i = header.getNext(); i != null; i =
i.getNext()) {
                if (i.getValue() % 2 == 0) {
                    xoa1NodeSauNode(p);
                    i = p;
                } else {
                    p = p.getNext();
                }
            }
        }
    }

    // ham in tat ca so am trong danh sach
    public void lietKeAm() {
        for (Node i = header; i != null; i = i.getNext()) {
            if (i.getValue() < 0) {
                System.out.println(i.getValue());
            }
        }
    }

    // ham tim so am lon nhat
    public Node timAmMax() {
        Node x = null;
```

```
        for (Node i = header; i != null; i = i.getNext()) {
            if (i.getValue() < 0) {
                x = i;
                break;
            }
        }
        if (x != null) {
            for (Node i = x; i != null; i = i.getNext()) {
                if (i.getValue() < 0 && i.getValue() > x.getValue()) {
                    x = i;
                }
            }
        }
        return x;
    }

    // ham hoan vi
    public void hoanVi(Node node1, Node node2) {
        int temp = node1.getValue();
        node1.setValue(node2.getValue());
        node2.setValue(temp);
    }

    // ham sap xep tang dan
    public void sortAccWithInterchange() {
        for (Node i = header; i != null; i = i.getNext()) {
            for (Node j = i.getNext(); j != null; j = j.getNext()) {
                if (i.getValue() > j.getValue()) {
                    hoanVi(i, j);
                }
            }
        }
    }

    // ham sap xep danh sach giam dan bang selectionSort
    public void selectionSort() {
        Node i, j, temp = null;
        for (i = header; i != null; i = i.getNext()) {
            int max = i.getValue();
            for (j = i.getNext(); j != null; j = j.getNext()) {
                if (j.getValue() > max) {
                    max = j.getValue();
                    temp = j;
                }
            }
```

```
            if (temp != null) {
                hoanVi(i, temp);
                temp = null;
            }
        }
    }

    // ham xuat kieu 2
    public void xuatDanhSach() {
        Node temp = header;
        if (temp == null) {
            System.out.println("Danh Sach Rong");
        } else {
            while (temp != null) {
                System.out.println(temp.getValue());
                temp = temp.getNext();
            }
        }
    }

    // ham tra ve so luong node
    public int sizeOfList() {
        int size = 0;
        Node temp = header;
        if (temp == null) {
            return size;
        } else {
            while (temp != null) {
                size++;
                temp = temp.getNext();
            }
        }
        return size;
    }

    // hien thi ba phan tu co gia tri lon nhat trong danh sach lien
ket
    public void hienThi_3PT_Max() {
        selectionSort();
        Node temp = header;
        for (int i = 0; i < 3; i++) {
            if (temp != null) {
                System.out.println(temp.getValue());
                temp = temp.getNext();
            }
```

```
    }
}

// ham sap xep
public void sapXepTang(boolean giaTri) {
    if (giaTri) {
        sortAccWithInterchange();
    } else {
        selectionSort();
    }
}

// chan so 2 vao sau Node co gia tri chan
public void chen2_VaoSauChan() {
    if (header == null) {
        return;
    }
    for (Node i = header; i != null; i = i.getNext()) {
        if (i.getValue() % 2 == 0) {
            them1NodeVaoSauNode(i, new Node(2));
            i = i.getNext();
        }
    }
}

// xoa tat ca cac so am trong danh sach lien ket cach 2
public void xoaTatCaAm() {
    if (header == null) {
        return;
    }
    Node NPre = null, i, tam = null;
    while (header.getValue() < 0) {
        xoa1NodeSauNode(null);
    }
    for (i = header; i != null; i = i.getNext()) {
        if (i.getValue() < 0) {
            xoa1NodeSauNode(NPre);
            i = NPre;
        } else {
            NPre = i;
        }
    }
}

// xoa tat ca phan tu trung nhau chi giu lai 1 phan tu
```

```
    public void xoaTrung() {
        if (header == null) {
            return;
        }
        for (Node i = header; i != null; i = i.getNext()) {
            Node NPre = i;
            for (Node j = i.getNext(); j != null; j = j.getNext()) {
                if (j.getValue() == i.getValue()) {
                    xoa1NodeSauNode(NPre);
                    j = NPre;
                } else {
                    NPre = NPre.getNext();
                }
            }
        }
    }

    // xoa k phan tu bat dau tu phan tu le dau tien
    public void xoa_K_PT_batDauTuLeDauTien(int k) {
        Node NPre = null;
        for (Node i = header; i != null; i = i.getNext()) {
            if (i.getValue() % 2 == 1) {
                for (int j = 0; j < k; j++) {
                    xoa1NodeSauNode(i);
                }
                break;
            } else {
                NPre = i;
            }
        }
    }

    // sap xep chan tang le giam
    public void chanTangLeGiam() {
        for (Node i = header; i != null; i = i.getNext()) {
            if (i.getValue() % 2 == 0) {
                for (Node j = i.getNext(); j != null; j = j.getNext())
{
                    if (j.getValue() % 2 == 0 && i.getValue() >
j.getValue()) {
                        hoanVi(i, j);
                    }
                }
            } else {
```

```
                for (Node k = i.getNext(); k != null; k = k.getNext())
{
                    if (k.getValue() % 2 != 0 && i.getValue() <
k.getValue()) {
                        hoanVi(i, k);
                    }
                }
            }
        }
    }

    // ham xoa cuoi
    public Node xoaCuoi() {
        if (header == null) {
            return null;
        }
        Node pre = null, tam = header;
        while (tam != tailer) {
            pre = tam;
            tam = tam.getNext();
        }
        return xoa1NodeSauNode(pre);
    }

    // ham xoa tat ca cac node trong danh sach
    public void deleteAll() {
        header = null;
    }
    // ham tron 2 danh thanh tang
    public simplelist tron(simplelist list) {
        simplelist lis = new simplelist();
        for (Node i = this.header; i != null; i = i.getNext()) {
            int x = i.getValue();
            lis.them1NodeVaoSauNode(lis.getTailer(), new Node(x));
        }

        Node tam = list.header;
        while (tam != null) {
            int x = tam.getValue();
            lis.them1NodeVaoSauNode(lis.getTailer(), new Node(x));
            tam = tam.getNext();
        }

        lis.sapXepTang(true);
```

```
        return lis;
    }
```

4. Bài tập liên quan đên stack

```
// bài tập đổi cơ số (ứng dung stack)
    public void doiCoSo(int so,int coSo){
        Stack<Integer> stack = new Stack<>();
        while(so!=0){
            stack.push(so%coSo);
            so = so/coSo;
        }
        while(!stack.empty()) System.out.print(stack.pop()+"\t");
        System.out.println();
    }

    // bài tập kiểm tra dấu ngoặc trong 1 biểu thức có hợp lệ hay
không
    public boolean check_Bracket(String Str){
        Stack<Integer> stack = new Stack<>();
        for (int i=0;i<Str.length();i++){
            if (Str.charAt(i)=='(') stack.push(1);
            else{
                if (Str.charAt(i)==')'){
                    if (stack.empty()) return false;
                    stack.pop();
                }
            }
        }
        if (!stack.empty()) return false;
        return true;
    }

    // bài tập tính giá trị biểu thức được nhập từ bàn phím.
    // có tính tính được cả có dấu nghoặc
    private int checkUuTien(String c) {
      if (c.equals("*")||c.equals("/")) return 2;
      else if (c.equals("+")||c.equals("-")) return 1;
      else return 0;
    }
    public float tinh(float so1,String toanTu,float so2) {
      switch (toanTu) {
            case "+":{
                 return so1+so2;
            }
            case "-":{
```

```
                return so1-so2;
            }
            case "*":{
                return so1*so2;
            }
            case "/":{
                return so1/so2;
            }
            default:
                return 0;
            }
    }
    public float tinhBieuThuc(String str) {
      Stack<String> stack = new Stack<>();
      ArrayList<String> arr = new ArrayList<>();
      for (int i=0;i<str.length();i++) {
            char getI = str.charAt(i);
            if (getI=='(') stack.push(getI+"");
            else {
                if (getI==')') {
                    String pop = stack.pop();
                    do {
                        arr.add(pop);
                        pop = stack.pop();
                    }while(!pop.equals("("));
                }
                else {
                    if (getI>='0'&&getI<='9') arr.add(getI+"");
                    else {
                        if (stack.empty()) stack.push(getI+"");
                        else {
                            if
(checkUuTien(getI+"")<=checkUuTien(stack.peek())) {
                                arr.add(stack.pop());
                                stack.push(getI+"");
                            }
                            else stack.push(getI+"");
                        }
                    }
                }
            }
      }
      while(!stack.empty()) {
            arr.add(stack.pop());
      }
```

```
      for (int i=0;i<arr.size();i++) {
            String getI = arr.get(i);
            if (getI.equals("+")||getI.equals("-
")||getI.equals("*")||getI.equals("/")) {
                  String so2 = stack.pop();
                  String so1 = stack.pop();
                  String kqT= tinh(Float.parseFloat(so1), getI,
Float.parseFloat(so2))+"";
                  stack.push(kqT);
            }
            else stack.push(getI);
      }
      return Float.parseFloat(stack.pop());
    }

    // bài tập ứng dụng stack để khử đệ quy trong thuật toán quick
sort
    public void khu_DQ_QuickSort(int []arr) {
      int i=0,j=arr.length-1,mid = (i+j)/2,left = 0,
      right = arr.length-1;
      Stack<Khoi_LR> stack = new Stack<>();
      stack.push(new Khoi_LR(i, j));
      while(!stack.empty()) {
            Khoi_LR temp = stack.pop();
            left = temp.getLeft();right = temp.getRight();
            i = temp.getLeft();j = temp.getRight();mid = (i+j)/2;
            do {
                  while(arr[i]<arr[mid]) i++;
                  while(arr[j]>arr[mid]) j--;
                  if (i<j) {
                        int t = arr[i];
                        arr[i] = arr[j];
                        arr[j] = t;
                  }
                  i++;j--;
            }while(i<=mid&&j>=mid);
            if (left<j) stack.push(new Khoi_LR(left, j));
            if (i<right) stack.push(new Khoi_LR(i, right));
      }
    }

// bài toán khử đệ quy thuật toán tháp hà nội (COPYRIGHT BY VIET
SACLO)
    // DATA IS CLASS . CLASS DATA HAVE THREE VALUE( A,B,C)
    public static void khuThapHN(int soDia,char A,char B,char C){
        Stack<data> stack = new Stack<>();
```

```
        while (true){
            if (soDia==1){
                System.out.println("Chuyen Dia Tu "+A+" ==> "+C);
                if (stack.empty()) return;
                data tam = stack.pop();
                soDia = tam.getSoDia();A = tam.getA();B = tam.getB();C
= tam.getC();
                System.out.println("Chuyen Dia Tu "+A+" ==> "+C);
                soDia-=1;
                char cotT = A;
                A = B;
                B = cotT;
            }
            else{
                stack.push(new data(soDia,A,B,C));
                soDia-=1;
                char cotT = B;
                B = C;
                C = cotT;
            }
        }
    }
```

- Phần 3 (3 điểm) gồm chương 4: bài toán có sử dụng ma trận Các bài toán về mảng danh sách liên kết

```
// hàm nhập mảng dánh sách liên kết
    public void scanALK(){
        input = new Scanner(System.in);
        System.out.print("Scan line of ALK: ");
        size = input.nextInt();
        alk = new LinkedList[size];
        for (int i=0;i<size;i++){
            alk[i] = new LinkedList<>();
            System.out.printf("Scan elemnent of ALK[%d]: ",i);
            int column = input.nextInt();
            for (int j=0;j<column;j++){
                System.out.printf("alk[%d][%d]= ",i,j);
                alk[i].add(input.nextInt());
            }
        }
    }

    // hàm in tất cả các phần tử trong mảng danh sách liên kết
    public void printALK(){
```

```
        for (int i=0;i<size;i++){
            for (Integer j:alk[i])
                System.out.printf("%d\t",j);
            System.out.println("\n");
        }
    }

    // hàm tìm giá trị lớn nhất trong mảng danh sách liên kết
    private String returnFirtIter(){
        String inter = "";
        for (int i=0;i<size;i++)
            for (Integer j:alk[i])
                return j+"";
        return inter;
    }

    public void returnMaxALK(){
        String firtInter = returnFirtIter();
        if (firtInter.isEmpty()){System.out.println("Can't find Max of
ALK");}
        else{
            int Max = Integer.parseInt(firtInter);
            for (int i=0;i<alk.length;i++)
                for (int j=0;j<alk[i].size();j++)
                    if (alk[i].get(j)>Max) Max = alk[i].get(j);
            System.out.println("*Max of alk is: "+Max);
        }
    }

    // hàm tính tổng các phần tử trên dòng i
    public int sumElementLine(int line){
        int result = 0;
        if (line>-1&&line<alk.length){
            for (Integer i:alk[line]) result+=i;
        }
        return result;
    }

    // hàm tính tổng tất cả các phần tử trong mảng danh sách liên kết
    public int sumElementALK(){
        int result = 0;
        for (int i=0;i<alk.length;i++)
            for (Integer j:alk[i])
                result+=j;
        return result;
```

```
}

// hàm đếm số lần xuất hiện của phần tử X bất kỳ
public int countElementX(int x){
    int count = 0;
    for (int i=0;i<alk.length;i++)
        for (Integer j:alk[i])
            if (j==x) count++;
    return count;
}

// hàm sắp xếp tăng theo tổng từng dòng
public void increseBySumLine(){
    int max,lineM;
    for (int i=0;i<alk.length-1;i++){
        max = this.sumElementLine(i);
        lineM = i;
        for (int j=i+1;j<alk.length;j++){
            if (this.sumElementLine(j)<max){
                max = this.sumElementLine(j);
                lineM = j;
            }
        }
        LinkedList<Integer> temp = alk[i];
        alk[i] = alk[lineM];
        alk[lineM] = temp;
    }
}

// hàm sắp xếp các phần tử tăng theo hình zic zac
public void sortALK_zicZac(){
    for (int i=0;i<alk.length;i++){
        for (int j=0;j<alk[i].size();j++){
            for (int l=0;l<alk.length;l++){
                for (int m=0;m<alk[l].size();m++){
                    if (alk[i].get(j)<alk[l].get(m)){
                        int temp = alk[i].get(j);
                        int temp1 = alk[l].get(m);
                        alk[i].set(j,temp1);
                        alk[l].set(m,temp);

                    }
                }
            }
        }
```

```
        }
    }

    // hàm kiểm tra các phần tử đối xúng qua dòng thứ n/2
    public boolean checkOposideALK(){
        int mid = size/2;
        int i=0,last = size-1;
        if (size%2==0||size==1) return false;
        while(i<mid&&last>mid){
            if (alk[i].size()!=alk[last].size()) return false;
            for (int column=0;column<alk[i].size();column++){
                if (alk[i].get(column)!=alk[last].get(column)) return
false;
            }
            i++;last--;
        }
        return true;
    }
```

5. Các bài toán trên ma trận liên kết

```
    // hàm tìm cột lớn nhất
    public int searchMaxColumn() {
        int maxC = -1;
        for (LinkedList<Node_LR> i : maTrix) {
            for (Node_LR j : i) {
                if (j.getEl() > maxC)
                    maxC = j.getEl();
            }
        }
        return maxC;
    }

    // hàm tính tổng hai ma trận thưa
    public MaTrix_Linked sumWith(MaTrix_Linked other) {
        if (size != other.size || this.searchMaxColumn() !=
other.searchMaxColumn())
            return null;
        MaTrix_Linked sum = new MaTrix_Linked(size);
        int i, j, sizeThis, sizeOther;
        for (int l = 0; l < size; l++) {
            sum.maTrix[l] = new LinkedList<>();
            i = 0;
            j = 0;
            sizeThis = maTrix[l].size();
```

```
                sizeOther = other.maTrix[l].size();
                while (i < sizeThis && j < sizeOther) {
                      if (maTrix[l].get(i).getEl() <
other.maTrix[l].get(j).getEl())
                            sum.maTrix[l].add(maTrix[l].get(i++));
                      else if (other.maTrix[l].get(j).getEl() <
maTrix[l].get(i).getEl())

      sum.maTrix[l].add(other.maTrix[l].get(j++));
                      else
                            sum.maTrix[l].add(new
Node_LR(maTrix[l].get(i).getEl(),
                                        maTrix[l].get(i++).getEr() +
other.maTrix[l].get(j++).getEr()));
                }
                while (i < sizeThis)
                      sum.maTrix[l].add(maTrix[l].get(i++));
                while (j < sizeOther)
                      sum.maTrix[l].add(other.maTrix[l].get(j++));
          }
          return sum;
      }

      // hàm trừ hai ma trận thưa
      public MaTrix_Linked tru(MaTrix_Linked other) {
          if (size != other.size || this.searchMaxColumn() !=
other.searchMaxColumn())
                return null;
          MaTrix_Linked result = new MaTrix_Linked(size);
          int i, j, sizeThis, sizeOther;
          for (int l = 0; l < size; l++) {
                result.maTrix[l] = new LinkedList<>();
                i = 0;
                j = 0;
                sizeThis = maTrix[l].size();
                sizeOther = other.maTrix[l].size();
                while (i < sizeThis && j < sizeOther) {
                      if (maTrix[l].get(i).getEl() <
other.maTrix[l].get(j).getEl())
                            result.maTrix[l].add(maTrix[l].get(i++));
                      else if (other.maTrix[l].get(j).getEl() <
maTrix[l].get(i).getEl())
                            result.maTrix[l]
```

```
                                        .add(new
Node_LR(other.maTrix[l].get(j).getEl(), -
other.maTrix[l].get(j++).getEr()));
                        else
                              result.maTrix[l].add(new
Node_LR(maTrix[l].get(i).getEl(),
                                        maTrix[l].get(i++).getEr() -
other.maTrix[l].get(j++).getEr()));
                  }
                  while (i < sizeThis)
                        result.maTrix[l].add(maTrix[l].get(i++));
                  while (j < sizeOther)
                        result.maTrix[l].add(new
Node_LR(other.maTrix[l].get(j).getEl(), -
other.maTrix[l].get(j++).getEr()));
            }
            return result;
      }

      // hàm nhân hai ma trận thưa
      public MaTrix_Linked nhanMaTrix(MaTrix_Linked other) {
            if (this.searchMaxColumn() != other.size - 1)
                  return null;
            MaTrix_Linked result = new MaTrix_Linked(size);
            for (int i = 0; i < size; i++) {
                  result.maTrix[i] = new LinkedList<Node_LR>();
                  for (int j = 0; j <= other.searchMaxColumn(); j++) {
                        int value = 0;
                        boolean yesAdd = false;
                        for (int l = 0; l < maTrix[i].size(); l++) {
                              int col = maTrix[i].get(l).getEl();
                              for (int m = 0; m <
other.maTrix[col].size(); m++) {
                                    if (other.maTrix[col].get(m).getEl()
== j) {
                                          value +=
maTrix[i].get(l).getEr() * other.maTrix[col].get(m).getEr();
                                          yesAdd = true;
                                          break;
                                    }
                              }
                        }
                        if (yesAdd)
                              result.maTrix[i].add(new Node_LR(j,
value));
```

```
                }
            }
            return result;
        }

// Node_LR là một class , trong class này chứa 2 phần tử là cột và giá trị
    // EL là Element left đại diện cho cột,
    // ER là Element right đại diện cho giá trị
    // hàm xuất ma trận thưa thành ma trận đầy đủ
    public void xuatFull() {
        for (LinkedList<Node_LR> i : maTrix) {
            int id = 0;
            for (Node_LR j : i) {
                for (; id < j.getEl(); id++) {
                    System.out.print("0\t");
                }
                System.out.print(j.getEr() + "\t");
                id++;
            }
            for (;id<=searchMaxColumn();id++){
                System.out.print("0\t");
            }
            System.out.println("");
        }
    }

    // hàm in dòng thứ i
    private void in_DongI(int i){
        for (Node_LR j:maTrix[i])
            j.printNode();
        System.out.println("\n");
    }

    // hàm in dòng nào có cột bằng 10
    public void in_dong_Cot10() {
        for (int i=0;i<size;i++){
            for (Node_LR j:maTrix[i])
                if (j.getEl()==10){
                    in_DongI(i);
                    break;
                }
        }
    }

    // hàm xóa dòng thứ i
```

```
public void xoaDongThuI(int i){
    for (int j=0;j<size;j++){
        if (j==i){
            for (int k=j;k<size-1;k++){
                maTrix[k] = maTrix[k+1];
            }
            size--;
            break;
        }
    }
}
```

❖ - Phần 4 (3 điểm) gồm chương 5: bài toán có sử dụng cây AVL

6. Download tree: https://drive.google.com/open?id=1NNuIJqHLrbVq4s7sVti1LtkQC37rHmqC

TÀI LIỆU THỰC HÀNH Ở TRƯỜNG

I. THỰC HÀNH BUỔI 1: https://drive.google.com/open?id=1a_ciaqd5mIk9QQxOZwTUBCdSvSG8p8jJ

II. THỰC HÀNH BUỔI 2: https://drive.google.com/open?id=1fbA-hzfcdmUrXptmFrXTyb_gB1-itSBO

III. THỰC HÀNH BUỔI 3: https://drive.google.com/open?id=1BodbZOwZpHaSedVihA6whUDJ1WgG3Srs

IV. THỰC HÀNH BUỔI 4: https://drive.google.com/open?id=15FHbQuCzXzFbfoTZSSVQitJBQjS6DWuv

V. THỰC HÀNH BUỔI 5: https://drive.google.com/open?id=12qbrjTqHv0ZkHwTug8v8jW-9qxyjIRZW

VI. THỰC HÀNH BUỔI 6: KIỂM TRA

VII. THỰC HÀNH BUỔI 7: https://drive.google.com/open?id=1pxQyn3VUYzFfoGFw5XBcx_2BB5ap_1fJ

VIII. THỰC HÀNH BUỔI 8: https://drive.google.com/open?id=1BoDyBPKJMS-jjlhFWJWgQ1hyy8VhHiwm

IX. THỰC HÀNH BUỔI 9: https://drive.google.com/open?id=1BoDyBPKJMS-jjlhFWJWgQ1hyy8VhHiwm

X. THỰC HÀNH BUỔI 10: https://drive.google.com/open?id=1NNuIJqHLrbVq4s7sVti1LtkQC37rHmqC

Phần kết

Tài liệu với danh nghĩa cá nhân không chắt đúng hoàn toàn. Có thể sẽ sai ở một nơi nào đó :D