**Exame época normal 2020**

1.

public static<K,E extends Comparable<E>> List<Pair<K,E>> mergeLists(Lists(Lists<Pair<K,E>> a, List<Pair<K,E>> b){

list<Pair<k,e>>result=new ArrayList<Pair<k,E>>();

int i=0,j=0;

Pair<k,e>pairA;

Pair<k,e>pairB;

while(i<a.size()&& j<b.size()){

pairA= a.get(i);

pairB=b.get(j);

if(pairA.getValue().compareTo(pairB.getValue())==0){

result.add(pairA);

result.add(pairB);

i++;

j++;

}

if(pairA.getValue().compareTo(pairB.getValue())<0){

result.add(pairA);

i++;

}

if(pairA.getValue().compareTo(pairB.getValue())>0){

result.add(pairB);

j++;

}

}

while(i<a.size()){

pairA=a.get(i++);

result.add(pairA);

}

while(j<b.size()){

pairB=b.get(j++);

result.add(pairB);

}

}

2

O que o método faz é colocar na lista alternadamente os caracteres das varias strings

é determinístico O(n2)

3.

fazer como o find

private E onOrderSource(Node<e> node, E elem, Node<E> prev){

if(node==null)

return null;

if(node.getElement().compareTo(elem)==0{

if(node.getRigth()!=null){

node=node.getRigth();

while(node.getLeft()!=null)

node=node.getLeft();

return node.getElement();

}

else return (prev==null ? null :prev.getElement());

if(node.getElement().compareTo(elem)>0)

inOrdersource(node.getLeft(),elem,node);

return inOrderSource(node.getRigth(),elem,prev);

}

4.

public class RedePetri{

private Graph<NoPetri, Integer> g = new Graph<> (true);

private boolean disparavel(NoPetri npt){

if(!npt.isTransicao()) return false;

for(Edge<NoPetri.Integer> e : g.incomingEdges(npt))

if(!e.getVOrig().temToken()) return false;

return true;

}

private List<NoPetri>nosDisparaveis(){

List <NoPetri> l = new ArrayList<>();

for(NoPetri np: g.vertices())=

if(disparavel(npl:np)) l.add(np);

return l;

}

}

5.

public HeapPriorityQueue<Integer,String> mergeHeaps(HeapPriorityQueue<Integer,String> hp1, HeapPriorityQueue<Integer, String> hp2){

int heapSize= hp1.size() + hp2.size();

Integer[] keys = new Integer[heapSize];

String[] values = new String [heapSize];

Entry<Integer, String> aux;

int i=0;

while (!hp1.isEmpty()){

aux= hp1.removeMin();

keys[i]=aux.getKey();

values[i++] = aux.getValue();

}

while (!hp2.isEmpty()){

aux= hp2.removeMin();

keys[i]=aux.getKey();

values[i++] = aux.getValue();

}

return new HepPriorityQueue<>(keys,values);

}