Departamento de Engenharia de Electrónica
e Telecomunicações e de Computadores
Lic. Engenharia Informática e Computadores

Algoritmos e Estruturas de Dados

Relatório Série Exercicios
Primeira Série

Semestre Verão
2009/2010

Elaborado por:
Grupo 3
30896 – Ricardo Canto
31453 – Ricardo Barreto
31654 – Ana Oliveira

A/C: Eng. José Simão
ISEL, 12 de Abril de 2010

1 – Introdução

1.

```
package Serie1;

public class Exercicio1 {
     /*
     * Dado o array ordenado v e o inteiro s, retorna dois elementos
desse array
     * tal que a sua soma seja igual a s. Esses dois elementos sao
retornados
     * atraves dum array com duas posi簗s. Caso nao existam dois
elementos cuja
     * soma seja s, deve ser retornado null. Serao penalizadas as
solu簗s com
     * custo superior a O(n), onde n 頡 dimensao de v.
     */
    static int read = 0;
    static int write = 0;

    public static int retirarMaiores(int[] a, int value) {
        int meio = -1, l = 0, r = a.length - 1;

        while (l <= r) {
            meio = l + (r - l) / 2;
            // se o valor do meio foi igual ao value retorna a ultima
posi硯
            // valida
            ++read;
            if (a[meio] == value)
                return meio - 1;
            // caso o valor do meio seja maior que o value
            ++read;
            if (a[meio] > value) {
                // verificar se o anterior 頭enor
                ++read;
                if (meio != 0 && a[meio - 1] < value)
                    return meio - 1; // caso seja retorna meio-1
                else
                    r = meio - 1; // se nao o right passa a ser meio-1
            } else {
                // se o valor do meio for menor que o meio e o valor
seguinte
                // for maior
                ++read;
                if ((meio != a.length - 1) && a[meio + 1] > value)
                    return meio; // retorna-se o meio
                else
                    l = meio + 1; // se nao actualiza-se o left para
meio+1;
            }
        }
        return -1;
    }

    public static int[] searchPositions(int[] v, int value, int l, int
r) {
        int sub = -1;
        int[] retorno = new int[2];
```

```java
        while (l <= r) {
            read +=2;
            sub = v[r] + v[l];
            if (sub == value) {
                read +=2;
                write +=2;
                retorno[0] = v[l];
                retorno[1] = v[r];
                return retorno;
            } else {
                if (sub > value)
                    --r;
                else
                    ++l;
            }
        }
        return null;
    }

    public static int[] findElementsThatSumTo(int[] v, int s) {
        ++read;
        if (v[0] >= 0) {
            int maiores = retirarMaiores(v, s);
            if (maiores == -1)
                return null;
            return searchPositions(v, s, 0, maiores);
        }
        return searchPositions(v, s, 0, v.length - 1);
    }

    public static void main(String[] args) {
        int[] v1 = { 1, 2, 3, 4, 7, 9, 12, 14, 16, 17 };
        int[] v2 = { 12, 14, 16, 17 };
        int[] v3 = {};
        int[] v4 = { 14, 16, 17 };
        int[] v5 = { 1, 2, 13 };
        int[] v6 = { 1, 2, 3 };
        int[] v7 = { -2, -1, 0, 14 };
        int value = 13;
        int[] ret = findElementsThatSumTo(v7, value);

        if (ret != null)
            System.out.println(ret[0] + " " + ret[1]);
        else
            System.out.println("Nao existem valores.");
        System.out.println("leituras: " + read+ ", escritas: "+write);
        /*
         * System.out.println(findElementsThatSumTo(v1, value));
         * System.out.println(findElementsThatSumTo(v2, value));
         * System.out.println(findElementsThatSumTo(v3, value));
         * System.out.println(findElementsThatSumTo(v4, value));
         * System.out.println(findElementsThatSumTo(v5, value));
         * System.out.println(findElementsThatSumTo(v6, value));`
         */
    }
}
```

2.

```
package Serie1;

public class Exercicio2 {

    static int read=0;
    static int write=0;

    public static int FindCommonElements(int[] a1, int[] a2){
        int count = 0;

        if(a1.length == 0 || a2.length == 0)
            return -1;

        for(int i=0; i< a1.length-1; ++i){
            read +=2;
            if(a1[i] != a1[i+1]){
                ++read;
                count += ArrayUtils.CountBinarySearch(a2, a1[i]);
            }
        }
        read+=2;
        if(a1[a1.length-1]==a1[a1.length-2]){
            ++read;
            count += ArrayUtils.CountBinarySearch(a2, a1[a1.length-
1]);
        }
        read+=ArrayUtils.read;
        System.out.println("leituras: " + read+ ", escritas: "+write);
        return count;
    }

    public static int
numberOfDistinctElementsOccuringOnBothSortedArrays(int[] v1, int[]
v2){
        //verificar o menor array
        if(v1.length <= v2.length)
            return FindCommonElements(v1, v2);
        return FindCommonElements(v2, v1);
    }

    public static void main(String[] args) {
        int [] a1 = {1,2,3,3,4,4,5,5,6};
        int [] a2 = {0,2,2,2,5,6,6};
        int [] a3 = {0,1,2,3,4,5,6,7,8,9,10};
        int [] a4 = {1,3,3,3,4,7,7};
        int [] a5 = {};
        int [] a6 = {1,1,1,1,1,1,1,1,1};

        int retorno =
numberOfDistinctElementsOccuringOnBothSortedArrays(a1, a2);
        if( retorno == 0)
            System.out.println("Nao tem elementos em comum.");
        else if(retorno == -1)
            System.out.println("Por favor introduza arrays com
valores.");
        else
            System.out.println(retorno);
    }
}
```

3.

```
package Serie1;

public class Exercicio3 {
    static int read=0;
    static int write=0;

    public static void nearestValues(int[] v, int x, int[] rt){
        int i=1;
        int ret =0;
        int [] aux = new int[v.length];

        for(int j=0; j<v.length; ++j){
            ++read;
            ++write;
            aux[j]=v[j];
        }
        ArrayUtils.mergeSort(aux, 0, aux.length-1);
        if((ret=ArrayUtils.indexBinarySearch(aux, x))== x){
            ++write;
            rt[0]=x;
        }
        else{
            ++read;
            ++write;
            rt[0]= aux[ret];
        }

        int l = ret-1;
        int r = ret+1;

        while( i<rt.length && l>=0 && r<aux.length){
            read+=2;
            int difLef = Math.abs(x-aux[l]);
            int difRig = Math.abs(x-aux[r]);

            if(difLef<=difRig){
                ++read;
                ++write;
                rt[i]=aux[l];
                --l;
            }
            else{
                read+=2;
                if(rt[i-1]!= aux[r]){
                    ++read;
                    ++write;
                    rt[i]=aux[r];
                }
            }
            ++i;
            ++r;
        }

        if(l<0 && r<rt.length){
            while(i<rt.length && r<rt.length)
                ++read;
                ++write;
                rt[i++]=v[r++];
        }
        else if(r>=rt.length && l>0){
            while(i<rt.length && l>=0){
                ++read;
                ++write;
                rt[i++]=v[l--];
            }
        }
    }


    public static int nearest (int[] v,int x, int[] r){
        if(r.length > v.length){
            for(int i=0; i<v.length; ++i){
                ++read;
                ++write;
                r[i]=v[i];
            }
            read+=ArrayUtils.read;
            System.out.println("leituras: " + read+ ", escritas:
"+write);
            return v.length;
        }
        else{
            nearestValues(v, x, r);
            read+=ArrayUtils.read;
            System.out.println("leituras: " + read+ ", escritas:
"+write);
            return r.length;
        }
    }

    public static void main(String[] args) {
        int[] a1 = {-2,3,4,5,2};
        int[] r = new int[2];
        System.out.println(nearest(a1, 4, r));
        System.out.println("");
        for(int i=0; i<r.length; ++i)
            System.out.println(r[i]);
    }
}
```

4.

```java
package Serie1;

import java.util.*;

public class Exercicio4 {
    static double timeInit;
    static double timeFin;

    public static int[] geraArray(int len) {
        int[] a = new int[len];
        Random r = new Random();

        for (int i = 0; i < len; ++i) {
            a[i] = r.nextInt();
        }

        Arrays.sort(a);
        return a;
    }

    public static int[] criaArray(int len) {
        return new int[len];
    }

    public static void main(String[] args) {

        ex41();
        ex42();
        ex43();
    }

    public static void ex41() {
        int[] a;
        int[] array1 = { 16, 32, 48, 64, 128, 256, 512, 1024, 1536,
2048, 4096,
                8192, 16384, 32768, 49152, 65536 };
        double[] times = new double[array1.length];

        for (int i = 0; i < array1.length; ++i) {
            a = geraArray(array1[i]);
            Arrays.sort(a);
            timeInit = System.nanoTime();
            Exercicio1.findElementsThatSumTo(a, 13);
            timeFin = System.nanoTime();
            times[i] = timeFin - timeInit;
            System.out.println("Read: "+Exercicio1.read+ ", Write:
"+Exercicio1.write);
        }

        for (int i = 0; i < times.length; ++i) {
            System.out.println(times[i]);
        }
    }

    public static void ex42() {
        int[] v1 = { 128, 256, 384, 512, 768, 1024 };
        int[] v2 = { 64, 128, 192, 256, 512, 768 };
        int[] a1;
        int[] a2;
        double[] times2 = new double[v1.length * v2.length];
        int nTimes2 = 0;

        for (int i = 0; i < v1.length; ++i) {
            a1 = geraArray(v1[i]);
            for (int j = 0; j < v2.length; ++j) {
                a2 = geraArray(v2[j]);
                Arrays.sort(a1);
                Arrays.sort(a2);
                timeInit = System.nanoTime();

Exercicio2.numberOfDistinctElementsOccuringOnBothSortedArrays(
                       a1, a2);
                timeFin = System.nanoTime();
                times2[nTimes2++] = timeFin - timeInit;
                System.out.println("Read: "+Exercicio2.read+ ", Write:
"+Exercicio2.write);
            }
        }

        for (int i = 0; i < times2.length; ++i) {
            System.out.println(times2[i]);
        }
    }

    public static void ex43() {
        int[] v = { 32, 64, 96, 128, 256, 512, 1024, 2048, 3072, 4096
};
        int[] r = { 4, 8, 12, 16, 20, 24 };
        int[] a3;
        int[] a4;

        double[] times3 = new double[v.length * r.length];
        int nTimes3 = 0;
        for (int i = 0; i < v.length; ++i) {
            a3 = geraArray(v[i]);
            for (int j = 0; j < r.length; ++j) {
                a4 = criaArray(r[j]);
                timeInit = System.nanoTime();
                Exercicio3.nearestValues(a3, 4, a4);
                timeFin = System.nanoTime();
                times3[nTimes3++] = timeFin - timeInit;
                System.out.println("Read: "+Exercicio3.read+ ", Write:
"+Exercicio3.write);
            }
        }

        for (int i = 0; i < times3.length; ++i) {
            System.out.println(times3[i]);
        }
    }
}
```

5.

```
package Serie1;
import java.io.File;
import java.io.FileFilter;
import java.util.ArrayList;

public class Exercicio5 {

    private static void addArray(ArrayList<File> ret, File[] aux) {
            for(int i=0; i<aux.length; ++i)
                ret.add(aux[i]);
    }

    public static File[] listAllFiles (File baseDir, FileFilter
filter){
        ArrayList <File> ret = new ArrayList <File> ();
        File[] files;
        File[] retArray;
        //usar iterador para percorrer a directoria e subdirectoria
        //por cada directoria verificar se o file pertence ao filter
        //se pertencer adiciona ao array

        //lista de todos os ficheiros da directoria
        files = baseDir.listFiles();

        for(int i=0; i<files.length; ++i){
            if(files[i].isDirectory()){
                retArray= listAllFiles(files[i], filter);
                //adicionar a lista
                if(retArray != null)
                    addArray (ret, retArray);
            }
                if(filter.accept(files[i]))
                    ret.add(files[i]);
        }

        if(ret.size()!=0){
            retArray = new File[ret.size()];
            return  ret.toArray(retArray);
        }
        return null;
    }

    public static void main(String[] args) {
        File directoria = new File("C:\\testes");
        File[] ret;

        txtFilter filter = new txtFilter();

        ret = listAllFiles(directoria, filter);

        if(ret != null)
            for(int i=0; i<ret.length; ++i){
                System.out.println(ret[i]);
            }
        else
            System.out.println("Nao foram encontrados ficheiros.");
    }

}
```

2 – Análise de desempenho

1.

```
package Serie1;

public class Recursivo1 {
    static int c8, c1,c2,c3,c4,c5,c6,c7;
    static double res;
    public static int method (int a,int  n){
        c1++;
        if(n==0){
            c2++;
            return 1;
        }
        c3++;
        if(n==1){
            c4++;
            return a;
        }
        c5++;
        if((n%2)==0){
            c6++;
            res=method(a,n/2);
            return (int) (res*res);
        }
        c7++;
            res =method(a,(n-1)/2);
            return (int) (res*res*a);
    }
    public static void main(String[] args) {
        System.out.println(method (799,2*2*2*2*2*2));
        System.out.println("chamou "+ c6);
        System.out.println("chamou "+ c7);
        System.out.println(res);
    }
}
```

4.1.

$$C(n) = C\left(\frac{n}{2}\right) + \frac{4}{2} \times n$$

$$C\left(\frac{n}{2}\right) = C\left(\frac{1}{2} \times \frac{n}{2}\right) + \frac{4}{3} \times \frac{n}{2}$$

$$C(n) = C\left(\frac{n}{4}\right) + \frac{4}{3} \times \frac{n}{2} + \frac{4}{3} \times n$$

$$C\left(\frac{n}{4}\right) = C\left(\frac{1}{4} \times \frac{n}{2}\right) + \frac{4}{3} \times \frac{n}{4}$$

$$C(n) = C\left(\frac{n}{8}\right) + \frac{4}{3} \times \frac{n}{4} + \frac{4}{3} \times \frac{n}{2} + \frac{4}{3} \times n$$

$$C\left(\frac{n}{8}\right) = C\left(\frac{1}{8} \times \frac{n}{2}\right) + \frac{4}{3} \times \frac{n}{8}$$

$$C(n) = C\left(\frac{n}{16}\right) + \frac{4}{3} \times \frac{n}{8} + \frac{4}{3} \times \frac{n}{4} + \frac{4}{3} \times \frac{n}{2} + \frac{4}{3} \times n$$

$$\boldsymbol{C(n) = C\left(\frac{n}{2^k}\right) + \sum_{k=1}^{n} \left(\frac{4}{3} \times \frac{n}{2^{k-1}}\right)}$$

4.2

$$C(n) = 3C\left(\frac{n}{3}\right) + n$$

$$C\left(\frac{n}{3}\right) = 3C\left(\frac{1}{3} \times \frac{n}{3}\right) + \frac{n}{3}$$

$$C(n) = 3^2 C\left(\frac{1}{3} \times \frac{n}{3}\right) + \frac{n}{3} + n$$

$$C\left(\frac{n}{3^2}\right) = 3C\left(\frac{1}{3^2} \times \frac{n}{3}\right) + \frac{n}{3^2}$$

$$C(n) = 3^3 C\left(\frac{1}{3^2} \times \frac{n}{3}\right) + \frac{n}{3^2} + \frac{n}{3} + n$$

$$\boldsymbol{C(n) = 3^k \times C\left(\frac{1}{3^{k-1}} \times \frac{n}{3}\right) + \sum_{k=1}^{n} \left(\frac{n}{3^{k-1}}\right)}$$

Anexos

ArrayUtils.java

```
package Serie1;


public class ArrayUtils {
    static int read=0;
    static int write =0;
/*  public static void mergeSort1 (int[] v, int l, int r){
        if(l<r){
            int meio = l+(r-l)/2;
            mergeSort1(v, l, meio);
            mergeSort1(v, meio+1, r);
            merge (v,l, meio, r);
        }
    }
    */
    private static void merge(int[] v, int l, int meio, int r) {
        int [] aux = new int[v.length];
        int j = l, k=meio+1;

        for(int i=0; i<v.length-1; ++i){
            if(v[i] < v[k] && j<= meio){
                aux[i]=v[j];
                ++i;
            }
            else{
                if(k < v.length){
                    aux[i]=v[k];
                    ++k;
                }
            }
        }

        for(int i=0; i<v.length; ++i){
            v[i]=aux[i];
        }
    }

    public static void swap (int[] v, int l, int r){
        int temp = v[l];
        v[l] = v[r];
        v[r]=temp;
    }

    public static int CountBinarySearch (int[]v, int value){
        int l =0, r = v.length-1, mid = 0;

        while(l<=r){
            mid = l+(r-l)/2;
            ++read;
            if(v[mid] == value)
                return 1;
            ++read;
            if(v[mid] > value)
                r = mid-1;
            else
                l = mid+1;
        }
    return 0;
}

public static int BinarySearch (int[]v, int value){
    int l =0, r = v.length-1, mid = 0;

    while(l<=r){
        mid = l+(r-l)/2;

        if(v[mid] == value)
            return mid;

        if(v[mid] > value)
            r = mid-1;
        else
            l = mid+1;
    }
    return -1;
}

public static int indexBinarySearch(int[]v, int value){
    int l =0, r = v.length-1, mid = 0;

    while(l<=r){
        mid = l+(r-l)/2;
        ++read;
        if(v[mid] == value)
            return mid;
        ++read;
        if(v[mid] > value)
            r = mid-1;
        else
            l = mid+1;
    }
    return l;
}

public static int lowerBound (int [] v, int value){
    int l=0, r=v.length-1,mid=0;
    while (l<=r){
        mid =l+(r-l)/2;
        if(v[mid]>=value) r=mid-1;
        else l=mid+1;
    }
return l;
}

public static void selectionSort(int[] v) {
    for (int i = 0; i < v.length-1; ++i) {
        int menor = SelectMenor(v, i);
        int aux;
        if(v[i]>v[menor]){
            aux = v[i];
            v[i]=v[menor];
            v[menor] = aux;
        }
    }
}

private static int SelectMenor(int[] v, int l) {
    int menor = -1;
```

```
        for(int i=1; i<v.length-1; ++i){
            if(v[i]<v[i+1])
                menor = i;
            else
                menor = i+1;
        }
        return menor;
    }

    public static void mergeSort(int[] list, int start, int end) {
        if (start >= end)
            return; // too many splits (return unsortable array)

        int mid = (start + end) / 2; // middle point

        mergeSort(list, start, mid); // split up left half
        mergeSort(list, mid + 1, end); // split up right half

        // start sorting
        int left_end = mid;
        int right_start = mid + 1;

        while ((start <= left_end) && (right_start <= end)) {
            read+=2;
            if (list[start] >= list[right_start]) // left most element
of Left
                                                        // side > left
most element
                                                        // of Right side
            {
                ++read;
                int temp = list[right_start];

                for (int i = right_start - 1; i >= start; i--) {
                    ++read;
                    ++write;
                    list[i + 1] = list[i]; // shift elements
                }
                ++write;
                list[start] = temp; // insert element where it belongs
                mid++;
                right_start++; // advance counters
            }
            start++; // advance in either case
        }
        return;
    }

}
```

testes.java

```
package Serie1;

public class testes {

	public static void mergeSort(int[] v, int l, int r) {
		if (l < r) {
			int meio = l + (r - l) / 2;
			mergeSort(v, l, meio);
			mergeSort(v, meio + 1, r);
			merge(v, l, meio, r);
		}
	}

	private static void merge(int[] v, int l, int meio, int r) {
		int[] left = new int[meio - l + 1];
		int[] rigth = new int[r - meio];

		for (int i = 0; i < left.length; ++i)
			left[i] = v[l + 1];

		for (int i = 0; i < rigth.length; ++i)
			rigth[i] = v[meio + 1 + i];

		int i = 0, j = 0, w = l;

		while (i < left.length && j < rigth.length) {
			if (left[i] < rigth[i]) {
				v[w] = left[i];
				++i;
			} else {
				v[w] = rigth[i];
				++j;
			}
			++w;
		}

		while (i < left.length)
			v[w++] = left[i++];

		while (i < rigth.length)
			v[w++] = rigth[i++];
	}

	public static void selectionSort(int[] v) {
		for (int i = 0; i < v.length-1; ++i) {
			int menor = SelectMenor(v, i);
			int aux;
			if(v[i]>v[menor]){
				aux = v[i];
				v[i]=v[menor];
				v[menor] = aux;
			}
		}
	}

	private static int SelectMenor(int[] v, int l) {
		int menor = -1;
		for(int i=l; i<v.length-1; ++i){
			if(v[i]<v[i+1])
				menor = i;
```

```
                else
                        menor = i+1;
                }
                return menor;
        }

        public static void main(String[] args) {
                int[] a = { -2, -4, 1, -5 };
                selectionSort(a);

                for (int i = 0; i < a.length; ++i)
                        System.out.println(a[i]);
        }

}
```

txtFilter.java

```
package Serie1;

import java.io.File;
import java.io.FileFilter;

public class txtFilter implements FileFilter{

      @Override
      public boolean accept(File file) {
            String name = file.getName();
            String ext=null;
            int i=name.length();

            //ler o nome do ficheiro da direita para a esquerda ate
encontrar ponto.
            //assim tenho a extensao do ficheiro
            //depois comparar com o a minha extensao (.txt)

            //while(name.charAt(i--)!='.');

            //while(i<name.length())
                        //ext = ext+(name.charAt(i++)+"");

            //if(ext.equals(".txt"))
            if(name.endsWith(".txt"))
                  return true;
            return false;

      }

}
```