Instituto Politécnico Nacional
Escuela Superior de Cómputo
Ingeniería en Sistemas Computacionales

Análisis y diseño de algoritmos
Práctica 3
Algoritmos ávidos: Asignación de tareas

Alumno: Sandoval Hernández Eduardo

Docente: Juárez Gambino Joel Omar

Grupo: 3CM1

Fecha de entrega: 17 de junio de 2021

Introducción

La estrategia de algoritmos ávidos es un método sencillo que se puede aplicar a muchos problemas. Dado un problema que tiene n entradas, la estrategia consiste en obtener un subconjunto de dichas entradas que satisfaga una condición determinada para el problema. Cada subconjunto que cumpla con las condiciones se toma como una posible solución. En la programación dinámica, de todas las posibles soluciones se selecciona la mejor, sin embargo, en la estrategia de algoritmos ávidos, se selecciona la mejor solución local y es con esta con la que se queda. No asegura que sea la mejor solución global.

El objetivo de esta práctica es resolver el problema de asignación de tareas o selección de actividades utilizando la estrategia de algoritmos ávidos. El problema consiste en encontrar a la mejor persona para realizar un trabajo, en particular se busca que la persona seleccionada pueda realizar el trabajo con el menor costo asociado.

Las condiciones del problema son las siguientes:

- Existe un número limitado de personas y tareas (misma cantidad de personas y tareas).
- Cualquier persona puede realizar cualquier tarea.
- Los costos dependerán de quien realiza la tarea.
- Las personas solamente pueden realizar una tarea a la vez.

El problema tiene dos opciones para implementar una estrategia voraz:

- Asignar el trabajo al trabajador que cobre menos.
- Asignar al trabajador el trabajo por el que cobre menos.

El reporte incluye el código de implementación en lenguaje java y los resultados para la prueba vista en clase.

Implementación

Para la implementación primero se creó una clase llamada Asignación con los campos “Nombre” de tipo String, “Tarea” de tipo String y “Costo” de tipo entero los cuales ayudaran a guardar que tarea se le da a que trabajador y a que costo. Su constructor asigna valores a cada uno de sus campos.

```
class Asignacion{
    public String Nombre;
    public String Tarea;
    public int Costo;

    public Asignacion(String Nombre, String Tarea, int Costo){
        this.Nombre = Nombre;
        this.Tarea = Tarea;
        this.Costo = Costo;
    }
}
```

La siguiente es la clase Tabla la cual entre sus campos se encuentran “Nombres[ ]” (array de tipo String), “Tareas[ ]” (array de tipo String), “Costos[ ][ ]” (array bidimensional de tipo entero, dos ArrayList de tipo “Asignacion” (listaAsignaciones, listaAsignaciones2) los cuales servirán para guardar los resultados de las dos opciones de implementación de la estrategia y por último dos ArrayList de tipo String los cuales servirán como auxiliares para guardar que trabajadores o tareas ya están asignados dependiendo de la opción de la estrategia. Su constructor inicializa cada uno de sus campos y les da a los arrays el tamaño asociado a la cantidad de trabajadores/tareas.

```
class Tabla{
    public String Nombres[];
    public String Tareas[];
    public int Costos[][];
    public ArrayList<Asignacion> listaAsignaciones;
    public ArrayList<Asignacion> listaAsignaciones2;
    public ArrayList<String> listaTrabajadores;
    public ArrayList<String> listaTareas;

    public Tabla(int n){
```

```
        Nombres = new String[n];
        Tareas = new String[n];
        Costos = new int[n][n];
        listaAsignaciones = new ArrayList<Asignacion>();
        listaAsignaciones2 = new ArrayList<Asignacion>();
        listaTrabajadores = new ArrayList<String>();
        listaTareas = new ArrayList<String>();
    }
```

Entre los métodos que maneja esta clase se encuentran impTareas (imprime las tareas que se manejan para el problema), impFilaCostos (imprime una determinada fina de la matriz de costos), impTabla (con ayuda de los métodos anteriores imprime en pantalla la tabla completa asemejándose a la forma en que se mostró la tabla en la explicación el problema en clase).

```
//Este metodo imprime las tareas
    public String impTareas(int n){
        if(n<Tareas.length)
            return Tareas[n] + "\t" + impTareas(n+1);
        else
            return "|";
    }

    //Este metodo imprime una determinada fila de la matriz de costos
    public String impFilaCostos(int f,int n){
        if(n<Costos.length)
            return Costos[f][n] + "\t\t" + impFilaCostos(f,n+1);
        else
            return "|";
    }

    //Este metodo imprime la tabla completa, utiliza los metodos anteriores
    public void impTabla(){
        System.out.println("-----------------------------------------");
        System.out.println("Trabajador" + "    " + impTareas(0)/*Arrays.toStr
ing(Tareas) impTrabajadores(0)*/);
        for(int i=0 ; i<Nombres.length ; i++)
            System.out.println(Nombres[i] + "    \t" + impFilaCostos(i,0));
        System.out.println("-----------------------------------------");
    }
```

También tiene los métodos costoTotal y costoTotal2 los cuales regresan el costo total obtenido con la opción 1 o 2 respectivamente.

```
//Este metodo determina el costo total de la primera opcion
    public int costoTotal(){
        int a = 0;
        for (Asignacion aux : listaAsignaciones)
            a += aux.Costo;
        return a;
    }

    //Este metodo determina el costo total de la segunda opcion
    public int costoTotal2(){
        int a = 0;
        for (Asignacion aux : listaAsignaciones2)
            a += aux.Costo;
        return a;
    }
```

De igual forma están los métodos impAsignaciones e impAsignaciones2 los cuales imprimen en pantalla los resultados de las opciones para implementar la estrategia, mostrando que trabajador realiza que tarea y a que costo, utilizan las funciones anteriores para imprimir la suma total de los costos.

```
public void impAsignaciones(){
        System.out.println("-----------------Opcion 1---------------------
");
        System.out.println("Seleccion final\nTrabajador\tTarea    \tCosto");
        System.out.println("-----------------------------------------------");
        for (Asignacion aux : listaAsignaciones)
            System.out.println(aux.Nombre + "    \t" + aux.Tarea + "\t" + au
x.Costo);
        System.out.println("-----------------------------------------------");
        System.out.println("El costo total es: " + costoTotal());
        System.out.println("-----------------------------------------------");
    }

    public void impAsignaciones2(){
        System.out.println("-----------------Opcion 2---------------------
");
        System.out.println("Seleccion final\nTrabajador\tTarea    \tCosto");
        System.out.println("-----------------------------------------------");
```

```
        for (Asignacion aux : listaAsignaciones2)
            System.out.println(aux.Nombre + "    \t" + aux.Tarea + "\t" + au
x.Costo);
        System.out.println("-----------------------------------------");
        System.out.println("El costo total es: " + costoTotal2());
        System.out.println("-----------------------------------------");
    }
```

El método Opcion1 es el que se encarga de aplicar la estrategia asignando la tarea al trabajador que menos cobre por ella. Para ello recorrerá el arreglo de costos columna por columna gracias al primer ciclo for, por cada que recorra una columna utilizará dos enteros auxiliares, uno para el costo y otro para la posición donde se encuentra dicho costo en la columna, en el segundo ciclo recorrerá fila por fila en la misma columna, dentro de este ciclo comprobará si el trabajador que corresponde a dicha fila ya tiene tarea asignada y en caso de que no sea así comprobará si el costo de hacer que dicho trabajador realice la tarea es menor que el valor actual del entero auxiliar (el cual tiene un valor inicial de 10000, superior a cualquier costo de la tabla) y guardará la posición del trabajador en el arreglo de Nombres en la variable auxiliar pos, si no se cumplen las condiciones anteriores simplemente no se tomará en cuenta el subconjunto actual como posible solución, al finalizar el segundo ciclo for agregará a listaTrabajadores el nombre del trabajador al cual se le asigno la tarea, de igual forma se creará la asignación con el nombre del trabajador, el costo y la tarea que realiza y se agregará a listaAsignaciones.

```
public void Opcion1(){
        for(int i=0 ; i<Nombres.length ; i++){
            int aux = 10000, pos = 0 ;
            for(int j=0 ; j<Nombres.length ; j++){
                if(!listaTrabajadores.contains(Nombres[j])){
                    if(Costos[j][i]<aux){
                        aux = Costos[j][i];
                        pos = j;
                    }
                }
            }
            listaTrabajadores.add(Nombres[pos]);
            Asignacion a = new Asignacion(Nombres[pos],Tareas[i],Costos[pos]
[i]);
```

```
            listaAsignaciones.add(a);
        }
    }
```

El método Opcion2 realiza algo similar al método Opcion1, con la diferencia de que en este caso se recorrerá el arreglo fila por fila, en el segundo ciclo for se comprobará en este caso si la tarea que corresponde a la posición donde se encuentra actualmente el ciclo no se encuentra ya asignada a un trabajador en caso de que no sea así comprobará si el costo de hacer que dicha tarea la realice el trabajador actual es menor que el valor actual del entero auxiliar (el cual también tiene un valor inicial de 10000, superior a cualquier costo de la tabla) y guardará la posición de la tarea en el arreglo de Tareas en la variable auxiliar pos, de igual forma si no se cumplen las condiciones anteriores simplemente no se tomará en cuenta el subconjunto actual como posible solución, al finalizar el segundo ciclo for agregará a listaTareas la tarea que ya tiene trabajador asignado, de igual forma se creará la asignación con el nombre del trabajador, el costo y la tarea que realiza y se agregará a listaAsignaciones2.

```
public void Opcion2(){
        for(int i=0 ; i<Nombres.length ; i++){
            int aux = 10000, pos = 0 ;
            for(int j=0 ; j<Nombres.length ; j++){
                if(!listaTareas.contains(Tareas[j])){
                    if(Costos[i][j]<aux){
                        aux = Costos[i][j];
                        pos = j;
                    }
                }
            }
            listaTareas.add(Tareas[pos]);
            Asignacion a = new Asignacion(Nombres[i],Tareas[pos],Costos[i][p
os]);
            listaAsignaciones2.add(a);
        }
    }
```

Finalmente, se tiene la clase AsignacionTareas la cual contiene el código principal del programa, en el main se encuentra un array de Strings el cual guarda los nombres de los trabajadores, un array de Strings el cual guarda las tareas y un array bidimensional de enteros el cual guarda el costo asociado al trabajador por tarea. A cada uno de estos arrays se le dan los valores del ejemplo mostrado en clase, se crea una Tabla con la cantidad de tareas/trabajadores como argumento, a los campos de dicha Tabla se les asigna los arrays con los Nombres, Tareas y Costos según lo requiera, se aplican los métodos Opcion1() y Opcion2() para resolver el problema y se muestra la tabla con los resultados con los métodos impTabla(), impAsignaciones() e impAsignaciones2().

```
public class AsignacionTareas {
    public static void main(String[] args){
        String trabajadores[] = {"Jaime","Felipe","Cesar","Bernardo","Alan"}
;
        String tareas[] = {"Carpinteria","Plomeria","Electricidad","Jardiner
ia","Albanileria"};
        int costos[][] = new int[5][5];
        costos[0][0] = 1100;
        costos[0][1] = 800;
        costos[0][2] = 830;
        costos[0][3] = 400;
        costos[0][4] = 3500;
        costos[1][0] = 1900;
        costos[1][1] = 350;
        costos[1][2] = 450;
        costos[1][3] = 400;
        costos[1][4] = 3000;
        costos[2][0] = 1000;
        costos[2][1] = 500;
        costos[2][2] = 510;
        costos[2][3] = 550;
        costos[2][4] = 1900;
        costos[3][0] = 1700;
        costos[3][1] = 320;
        costos[3][2] = 700;
        costos[3][3] = 500;
        costos[3][4] = 4900;
        costos[4][0] = 2000;
        costos[4][1] = 300;
        costos[4][2] = 600;
```

```
        costos[4][3] = 280;
        costos[4][4] = 2000;

        Tabla T1 = new Tabla(5);
        T1.Nombres = trabajadores;
        T1.Tareas = tareas;
        T1.Costos = costos;
        T1.Opcion1();
        T1.Opcion2();
        T1.impTabla();
        T1.impAsignaciones();
        T1.impAsignaciones2();
    }
}
```

Resultados

De la tabla vista en clase:

Trabajador	Carpintería	Plomería	Electricidad	Jardinería	Albañilería
Jaime	$1,100	$800	$830	$400	$3,500
Felipe	$1,900	$350	$450	$400	$3,000
Cesar	$1,000	$500	$510	$550	$1900
Bernardo	$1,700	$320	$700	$500	$4,900
Alan	$2,000	$300	$600	$280	$2,000

Se obtuvieron los siguientes resultados:

```
Scanning for projects...

-------< ADA_SandovalHernandez:Practica3_AlgoritmosVoraces_SHE >--------
Building Practica3_AlgoritmosVoraces_SHE 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Practica3_AlgoritmosVoraces_SHE ---
-----------------------------------------------
Trabajador   Carpinteria        Plomeria        Electricidad    Jardineria      Albanileria     |
Jaime           1100            800             830             400             3500            |
Felipe          1900            350             450             400             3000            |
Cesar           1000            500             510             550             1900            |
Bernardo        1700            320             700             500             4900            |
Alan            2000            300             600             280             2000            |
-----------------------------------------------
-----------------Opcion 1----------------------
Seleccion final
Trabajador      Tarea           Costo
-----------------------------------------------
Cesar           Carpinteria     1000
Alan            Plomeria        300
Felipe          Electricidad    450
Jaime           Jardineria      400
Bernardo        Albanileria     4900
-----------------------------------------------
El costo total es: 7050
-----------------------------------------------
-----------------Opcion 2----------------------
Seleccion final
Trabajador      Tarea           Costo
-----------------------------------------------
Jaime           Jardineria      400
Felipe          Plomeria        350
Cesar           Electricidad    510
Bernardo        Carpinteria     1700
Alan            Albanileria     2000
-----------------------------------------------
El costo total es: 4960
-----------------------------------------------
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.083 s
Finished at: 2021-06-16T19:28:52-05:00
------------------------------------------------------------------------
```

Código completo

```java
package ada_sandovalhernandez.practica3_algoritmosvoraces_she;

import java.util.ArrayList;

class Asignacion{
    public String Nombre;
    public String Tarea;
    public int Costo;

    public Asignacion(String Nombre, String Tarea, int Costo){
```

```
        this.Nombre = Nombre;
        this.Tarea = Tarea;
        this.Costo = Costo;
    }
}

class Tabla{
    public String Nombres[];
    public String Tareas[];
    public int Costos[][];
    public ArrayList<Asignacion> listaAsignaciones;
    public ArrayList<Asignacion> listaAsignaciones2;
    public ArrayList<String> listaTrabajadores;
    public ArrayList<String> listaTareas;

    public Tabla(int n){
        Nombres = new String[n];
        Tareas = new String[n];
        Costos = new int[n][n];
        listaAsignaciones = new ArrayList<Asignacion>();
        listaAsignaciones2 = new ArrayList<Asignacion>();
        listaTrabajadores = new ArrayList<String>();
        listaTareas = new ArrayList<String>();
    }

    //Este método resuelve el problema por la primera opcion
    public void Opcion1(){
        for(int i=0 ; i<Nombres.length ; i++){
            int aux = 10000, pos = 0 ;
            for(int j=0 ; j<Nombres.length ; j++){
                if(!listaTrabajadores.contains(Nombres[j])){
                    if(Costos[j][i]<aux){
                        aux = Costos[j][i];
                        pos = j;
                    }
                }
            }
            listaTrabajadores.add(Nombres[pos]);
            Asignacion a = new Asignacion(Nombres[pos],Tareas[i],Costos[pos]
[i]);
            listaAsignaciones.add(a);
        }
    }

    //Este método resuelve el problema por la segunda opcion
```

```
    public void Opcion2(){
        for(int i=0 ; i<Nombres.length ; i++){
            int aux = 10000, pos = 0 ;
            for(int j=0 ; j<Nombres.length ; j++){
                if(!listaTareas.contains(Tareas[j])){
                    if(Costos[i][j]<aux){
                        aux = Costos[i][j];
                        pos = j;
                    }
                }
            }
            listaTareas.add(Tareas[pos]);
            Asignacion a = new Asignacion(Nombres[i],Tareas[pos],Costos[i][p
os]);
            listaAsignaciones2.add(a);
        }
    }

    //Este metodo imprime las tareas
    public String impTareas(int n){
        if(n<Tareas.length)
            return Tareas[n] + "\t" + impTareas(n+1);
        else
            return "|";
    }

    //Este metodo imprime una determinada fila de la matriz de costos
    public String impFilaCostos(int f,int n){
        if(n<Costos.length)
            return Costos[f][n] + "\t\t" + impFilaCostos(f,n+1);
        else
            return "|";
    }

    //Este metodo imprime la tabla completa, utiliza los metodos anteriores
    public void impTabla(){
        System.out.println("-----------------------------------------");
        System.out.println("Trabajador" + "   " + impTareas(0)/*Arrays.toStr
ing(Tareas) impTrabajadores(0)*/);
        for(int i=0 ; i<Nombres.length ; i++)
            System.out.println(Nombres[i] + "    \t" + impFilaCostos(i,0));
        System.out.println("-----------------------------------------");
    }

    //Este metodo determina el costo total de la primera opcion
```

```
    public int costoTotal(){
        int a = 0;
        for (Asignacion aux : listaAsignaciones)
            a += aux.Costo;
        return a;
    }

    //Este metodo determina el costo total de la segunda  opcion
    public int costoTotal2(){
        int a = 0;
        for (Asignacion aux : listaAsignaciones2)
            a += aux.Costo;
        return a;
    }

    public void impAsignaciones(){
        System.out.println("-----------------Opcion 1----------------------
");
        System.out.println("Seleccion final\nTrabajador\tTarea    \tCosto");
        System.out.println("-----------------------------------------------");
        for (Asignacion aux : listaAsignaciones)
            System.out.println(aux.Nombre + "    \t" + aux.Tarea + "\t" + au
x.Costo);
        System.out.println("-----------------------------------------------");
        System.out.println("El costo total es: " + costoTotal());
        System.out.println("-----------------------------------------------");
    }

    public void impAsignaciones2(){
        System.out.println("-----------------Opcion 2----------------------
");
        System.out.println("Seleccion final\nTrabajador\tTarea    \tCosto");
        System.out.println("-----------------------------------------------");
        for (Asignacion aux : listaAsignaciones2)
            System.out.println(aux.Nombre + "    \t" + aux.Tarea + "\t" + au
x.Costo);
        System.out.println("-----------------------------------------------");
        System.out.println("El costo total es: " + costoTotal2());
        System.out.println("-----------------------------------------------");
    }

}

public class AsignacionTareas {
    public static void main(String[] args){
```

```
        String trabajadores[] = {"Jaime","Felipe","Cesar","Bernardo","Alan"}
;
        String tareas[] = {"Carpinteria","Plomeria","Electricidad","Jardiner
ia","Albanileria"};
        int costos[][] = new int[5][5];
        costos[0][0] = 1100;
        costos[0][1] = 800;
        costos[0][2] = 830;
        costos[0][3] = 400;
        costos[0][4] = 3500;
        costos[1][0] = 1900;
        costos[1][1] = 350;
        costos[1][2] = 450;
        costos[1][3] = 400;
        costos[1][4] = 3000;
        costos[2][0] = 1000;
        costos[2][1] = 500;
        costos[2][2] = 510;
        costos[2][3] = 550;
        costos[2][4] = 1900;
        costos[3][0] = 1700;
        costos[3][1] = 320;
        costos[3][2] = 700;
        costos[3][3] = 500;
        costos[3][4] = 4900;
        costos[4][0] = 2000;
        costos[4][1] = 300;
        costos[4][2] = 600;
        costos[4][3] = 280;
        costos[4][4] = 2000;

        Tabla T1 = new Tabla(5);
        T1.Nombres = trabajadores;
        T1.Tareas = tareas;
        T1.Costos = costos;
        T1.Opcion1();
        T1.Opcion2();
        T1.impTabla();
        T1.impAsignaciones();
        T1.impAsignaciones2();
    }
}
```