

**PONTIFICIA UNIVERSIDAD JAVERIANA**

**ANÁLISIS DE ALGORITMOS**

**SEGUNDA ENTREGA PROYECTO:**

**DISEÑO DETALLADO DEL JUGADOR**

**NORMAN DANILO CASTRO**

**JUAN SEBASTIÁN PRADO VALERO**

**SANTOS DAVID NÚÑEZ VILLAMIL**

**27/10/2019**

1. **nombreJugador:**

String nombreJugador()

{

String name = this.nombre;

return name;

}

**Análisis Temporoespacial:**

Tamaño de la entrada: ninguna

Operación Básica: asignación ( vale 1)

T(n) = O(1)

S(n) = O(1)

1. **inicializar:**

void inicializar( int numeroAtributos, int numeroValores )

{

this.n = numeroAtributos;

this.m = numeroValores;

}

**Análisis Temporoespacial:**

Tamaño de la entrada: valor de los parámetros: n y m

Operación Básica: asignación ( vale 1)

T(n) = O(1)

S(n) = O(1)

1. **condicionConjunto:**

Recibe una lista de cartas y una carta a adicionar y valida si la nueva carta hace conjunto con la lista de cartas. Retorna True si forma conjunto o de lo contrario, False.

boolean condicionConjunto ( Seq[Carta] lista, Carta nueva)

{

boolean iguales = true, distintos = true, resultado = true;

for ( int i = 0; i < n; i++)

{

for( Carta card : lista )

{

if ( card.valorDeArtributo( i ) = = nueva.valorDeArtributo( i ) )

distintos = false;

else

iguales = false;

}

if ( distintos = = true || iguales = = true)

resultado = true;

else

return false;

iguales = true;

distintos = true;

}

return true;

}

**Análisis Temporoespacial:**

Tamaño de la entrada: tamaño de una lista de cartas: v, cantidad de atributos: n

Operación Básica: comparación (vale 1)

**-Temporal:**

Al realizar un ciclo la cantidad de atributos que hay ( n veces ) y dentro de este se recorre una lista de cartas recibida de tamaño “v” y en este ciclo se realizan comparaciones, por lo que T(n,v) = O(n\*v).

**-Espacial:**

Al utilizar solo tres variables booleanas, S(n,v) = O(1).

1. **jugar:**

List<Carta> jugar( List<Carta> cartasEnMesa )

{

ArrayList<ArrayList<Carta>> c = new ArrayList<>();

for(int i = 0; i < m; i++)

{

ArrayList<Carta> aux = new ArrayList<>();

c.add(aux);

}

c.get(0).add(cartasEnMesa.get(0));

int cont = 1, cont2 = 1, p = cartasEnMesa.size();

for(int i = 1; i<m; i++)

{

if(condicionConjunto(c.get(i-1), cartasEnMesa.get(cont)))

{

c.get(i).addAll(c.get(i-1));

c.get(i).add(cartasEnMesa.get(cont));

}

else

{

for(int k = cont+1; k< p; k++)

{

if(condicionConjunto(c.get(i-1), cartasEnMesa.get(k)))

{

c.get(i).addAll(c.get(i-1));

c.get(i).add(cartasEnMesa.get(k));

}

}

if(c.get(i).isEmpty())

{

c.get(i-1).addAll(c.get(i-2));

c.get(i-1).add(cartasEnMesa.get(cont));

i--;

}

}

if(!c.get(m-1).isEmpty())

break;

if(cont = = p-1)

{

c.clear();

for(int j = 0; j < m; j++)

{

ArrayList<Carta> aux = new ArrayList<>();

c.add(aux);

}

c.get(0).add(cartasEnMesa.get(cont2));

cont2++;

cont = cont2;

i = 1;

}

if(cont2 = = p-m)

break;

cont++;

}

return c.get(m-1);

}

**Análisis Temporoespacial:**

Tamaño de la entrada: tamaño de la lista de cartasEnMesa: p, cantidad de valores: m, cantidad de atributos: n.

Operación Básica: llamada a la función condicionConjunto ( vale O(n\*v) )

**-Temporal:**

En el peor caso, las m cartas que generan un conjunto son las ultimas m cartas de la lista cartasEnMesa.

**-Espacial:**

Al utilizar solo tres variables booleanas, S(n,v) = O(1).