**Diseño y Análisis de Algoritmos**

**TP Obligatorio - 2C-2023**

Profesor:

Orosco, Ricardo Fabian

Leonetti, Cesar Osvaldo

Grupo:

Angel, Leonardo – LU: 1076550

Cardozo, Ivana Elizabeth– LU: 1071244

Greco, Tiziano– LU: 1150710

Zanon, Francisco– LU: 1135581

Buenos Aires, 21 de Noviembre de 2023.-

**Tabla de Contenidos**

[Introducción 3](#_heading=h.gjdgxs)

[Descripción del Problema 3](#_heading=h.30j0zll)

[Tipos de Datos Abstractos 3](#_heading=h.3dy6vkm)

[Estrategia de Resolución 3](#_heading=h.1fob9te)

[Análisis de Complejidad Temporal 3](#_heading=h.3znysh7)

[Conclusiones 3](#_heading=h.2et92p0)

[Bibliografía 4](#_heading=h.tyjcwt)

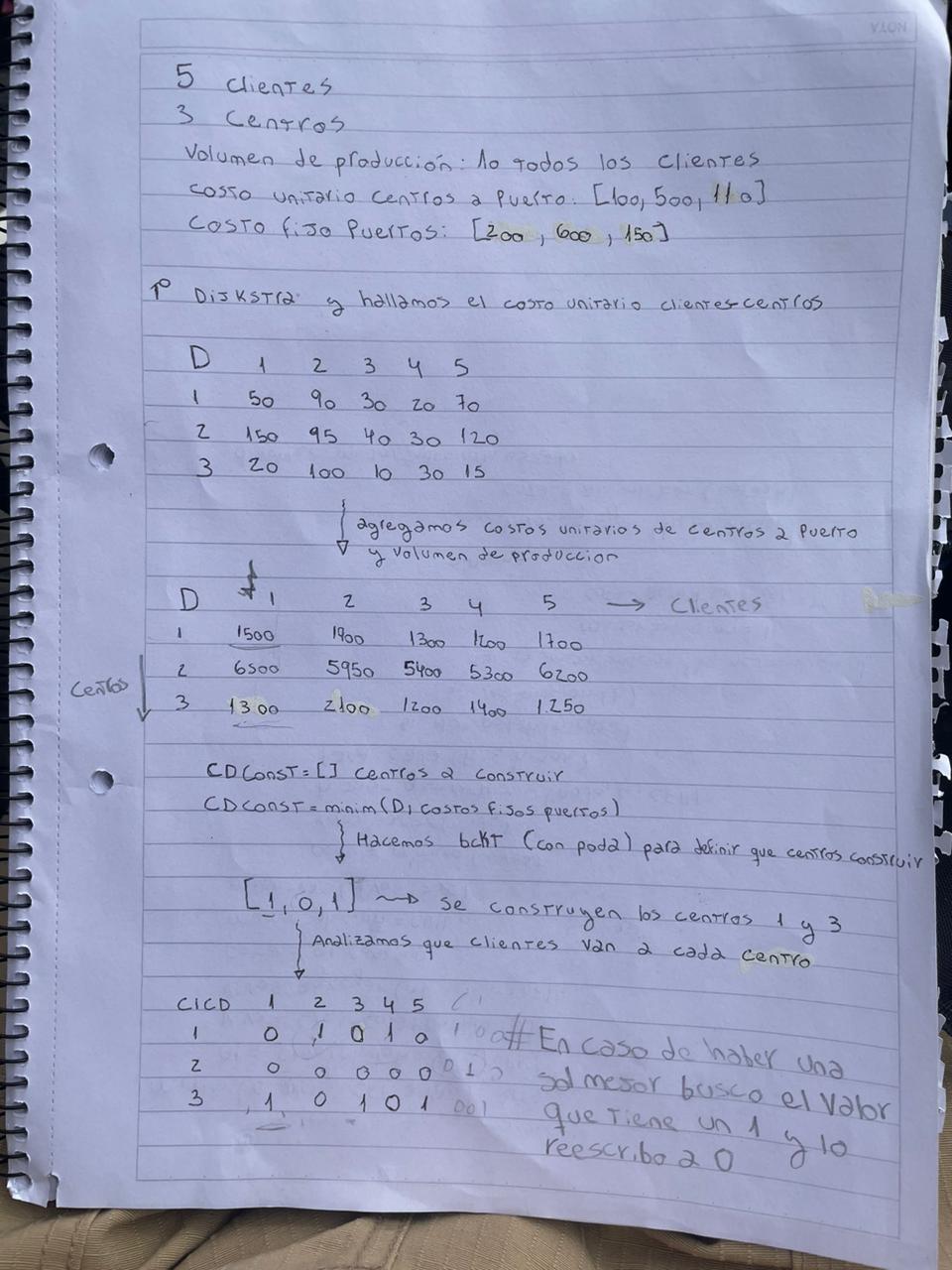
# Introducción

*Este trabajo consiste en tener que hallar, mediante la utilización de los algoritmos de Dijkstra y backtracking, qué centros de distribución deben construirse de manera de poder hallar el mínimo costo total anual de transportar la materia prima en un proceso de logística mediante un análisis estratégico de que centros de distribución deben construirse, y determinar a qué centro de distribución enviará la materia prima cada cliente.*

# Descripción del Problema

## Estrategia de Resolución

*Este es un ejemplo con 5 clientes y 3 centros para que se vea de manera más clara el funcionamiento de nuestro algoritmo. No es el pseudocódigo, es un ejemplo gráfico que muestra la estrategía que utilizamos en cada paso*

**

## 

## Pseudocódigo del Algoritmo de Resolución del Problema

**Algoritmo Centros**

Funcion leerTXT

CI=cant clientes(int)

CD=cant centros(int)

Vpa=volumen de prod anual []

CDP= costo unitario centro-puerto []

CFP = costo fijo centro-puerto []

D=[CD][CI]

G= Grafo

para i en CD

R= Dijkstra(G,i)

para j en CI

D[i][j]= R.pesoArista(i+50,j)

para i en CL

para j en CD

D[i][j]= (D[j][i]\*Vpa[i])+(CDP[j]\*Vpa[i])

CDconst= centros a construir []

CDcons= minim(D,CFP)

CICD=[CD][CI]

para cad i en CI

mejorSol=+

Pos=-1

para cada j en CD

si CDconst[j]==1

si D[j][i]<mejorSol

mejorSol=D[j][i]

CICD[j][i]=1

si pos != -1

CICD[pos][i]=0

pos=j

sino

CICD[j][i]=0

sino CICD[j][i]=0

Print(CDconst) # Centros a construir

Print(ClCD) # Qué clientes envían a qué centro

minim(C:array, CFP:array):array

n1= crearNodoRaiz(C,CFP)

Vivos=crearColaPrioridad()

Vivos.agregar(n1, n1.cotainf)

cota=n1.cotaSup

mejorSolucion=null

mientras(Vivos!=0)

nodo=primero(Vivos)

Vivos.sacar(nodo)

hijos= generarHijos(nodo, C, CFP)

para cada (h en hijos)

si NO podar (h ,cota)

si esSolucion(h)

si es mejorSolucion(h, mejorSolucion)

mejorSolucion=h

cota=actualizar(cota, h)

sino

Vivo.agregar(h, h.cotainf)

cota=actualizar(cota, h)

devolver mejorSolucion.estado

## 

## Análisis de Complejidad Temporal

*La complejidad del algoritmo diseñado es de O(2n) ya que la función cuyo costo temporal es más alto en nuestro algoritmo es “minim” y su complejidad es O(2n)*

**Algoritmo Centros**

Funcion leerTXT => O(n)

CI=cant clientes(int)

CD=cant centros(int)

Vpa=volumen de prod anual []

CDP= costo unitario centro-puerto []

CFP = costo fijo centro-puerto []

D=[CD][CI]

G= Grafo

para i en CD

R= Dijkstra(G,i) => O(n^2) =>O(n^2\*n) =>O(n^3)

para j en CI

D[i][j]= R.pesoArista(i+50,j) => O(n)

para i en CL

para j en CD

D[i][j]= (D[j][i]\*Vpa[i])+(CDP[j]\*Vpa[i]) =>O(n\*n)

CDconst= centros a construir []

CDcons= minim(D,CFP)

CICD=[CD][CI]

para cad i en CI => O(n)

mejorSol=+

Pos=-1

para cada j en CD =>O(n)

si CDconst[j]==1

si D[j][i]<mejorSol

mejorSol=D[j][i]

CICD[j][i]=1

si pos != -1

CICD[pos][i]=0

pos=j

sino

CICD[j][i]=0

sino CICD[j][i]=0

**Complejidad Temporal:** O(n\*n) =>de los for anidados

Print(CDconst) # Centros a construir

Print(ClCD) # Qué clientes envían a qué centro

minim(C:array, CFP:array):array

n1= crearNodoRaiz(C,CFP)

Vivos=crearColaPrioridad()

Vivos.agregar(n1, n1.cotainf)

cota=n1.cotaSup

mejorSolucion=null

mientras(Vivos!=0)

nodo=primero(Vivos)

Vivos.sacar(nodo)

hijos= generarHijos(nodo, C, CFP)

para cada (h en hijos)

si NO podar (h ,cota)

si esSolucion(h)

si es mejorSolucion(h, mejorSolucion)

mejorSolucion=h

cota=actualizar(cota, h)

sino

Vivo.agregar(h, h.cotainf)

cota=actualizar(cota, h)

devolver mejorSolucion.estado

**Complejidad Temporal: minim()** =>(2^n) => en cada nivel del árbol el número de nodos se multiplica por 2

O(n+(n\*m)+(n^3\*n)+(n\*n)+(2^n))=O(2^n)

# Conclusiones

*En este problema se busca analizar qué centros de distribución se van a abrir y qué clientes se van a asignar a cada uno de estos centros.*

*Primero, para calcular el costo unitario de transporte del cliente a cada centro, utilizamos dijkstra ya que nos proporciona el costo mínimo de un nodo al resto de los nodos.*

*Luego, para decidir cuál centro construir y cual no, utilizamos backtracking (con poda) para que analice todas las opciones sin analizar aquellas que es seguro que el resultado no va a ser el mejor.*

*Por último creamos una matriz y, utilizando la información de las distancias y los centros que se van a construir, determinamos qué cliente iba a distribuir su materia prima a qué centro.*

# Bibliografía

-Apuntes Cátedra - Programación III