Problema 1

A) Utilizar el método de programación dinámica para diseñar un algoritmo que resuelva un problema entrega soluciones de peor calidad que utilizar un método goloso.

Falso, ya que el método goloso solamente nos entrega la respuesta optima al problema en ciertas ocasiones, mientras que la programación dinámica nos asegura una respuesta optima.

B) Programación dinámica permite construir algoritmos eficientes para los problemas pertenecientes a la clase NP-duro.

Falso, los problemas del tipo NP-duro son lo suficientemente complejos como para que no se pueda crear un algoritmo eficiente, además la programación dinámica utiliza la recursión y la subdivisión de problema, por lo que primero es necesario saber si es que el problema a resolver puede ser subdividido en instancias más pequeñas de él.

Problema 2

Dado un conjunto S de *n* elementos y un número natural $m < n / 2$, desarrollar un algoritmo que divida el conjunto S en $m + 1$ subconjuntos $S_1, S_2, \ldots, S_{m+1}$ no vacíos tales que los elementos de S_i sean menores que los de $S_{i+1}, \ldots, S_{m+1}$ para todo $1 \leq i \leq m$. Explique la idea básica y los supuestos que considere. Analice y comente.

Teniendo **S**, un conjunto no vacío con **n** de elementos, **m** un número entero menor a **n**/2 y los subconjuntos $\mathbf{C}_i$, con i = 1, 2, …, **m**+1, los cuales inicialmente se encuentran vacíos y j = 1, 2, …, n.

```
Ordenar S de menor a mayor
Mientras (i <= m) hacer
      Ci = Ci U {S1}      //Agregar el primer elemento de S en Ci
      i = i+1
Ci = Ci U {S}             //Agregar el resto de los elementos de S en Ci
```

Ya que es necesario que los elementos de C_i sean menores que los elementos de C_{i+1}, comenzamos ordenando los elementos dentro de S de menor a mayor; de esta manera podremos distribuir los elementos desde i = 1 hasta i = m+1 asegurándonos que el primer subconjunto tenga los elementos menores. Ya que no se especifica mucho sobre el problema, solo que los subconjuntos no deben ser vacíos, se agrega el primer elemento del conjunto S (menor elemento actual), en el subconjunto C_i, de esta manera poner un elemento entre los subconjuntos con índice entre 1 y m, cumpliendo con el hecho de que los elementos de cada subconjunto tienen que ser menor a los elementos del subconjunto que le sigue. Para finalizar se agregan los elementos restantes de S en C_{m+1}.

Si analizamos el algoritmo construido, podemos notar lo siguiente:

- Ordenar los elementos tiene una complejidad: O(n*log(n))
- El ciclo tiene una complejidad: O(n)

Por lo que se puede decir que el algoritmo construido tiene un orden de complejidad O(n*log(n)), correspondiente al ordenamiento de los elementos, por lo que se reduce la complejidad de esta tarea, el algoritmo podría ser más eficiente.

Problema 3

Enunciado:

Una empresa requiere distribuidor sus productos a *n* tiendas de venta directa a clientes. Esta empresa contrata a un repartidor que trabaja con su vehículo particular. Este repartidor retira
los productos en la mañana y los reparte a los destinos indicados. Él cobra por los paquetes entregados, por lo cual le interesa hacerlo gastando la menor cantidad de bencina, sin importar que se demore más tiempo. Además, en algunas partes debe pagar estacionamiento
para entregar el paquete. Por lo tanto, él lo contrata para planificar la ruta desde que sale de la empresa, pasa por todas las tiendas y vuelve a su casa minimizando el recorrido total. Se conoce:

- la ubicación geográfica de la empresa,
- la ubicación geográfica de cada una de las tiendas a las que debe distribuir el producto,
- la ubicación geográfica de la casa del repartidor.

Desarrolle un algoritmo para planificar el orden de entrega de los pedidos recorriendo la distancia total mínima. Analice el problema. Explique el método utilizado. Explique la idea básica y los supuestos que considere. Evalúe su método. Analice y comente. Aplique al caso
particular presentado en la Tabla 1.

Teniendo un grafo G(V,A) donde V son los vértices (lugares a visitar), A la matriz de adyacencia que representa las conexiones entre cada vértice, n la cantidad de vértices e i = 1, 2, …, n-1. Se asume que inicialmente los vértices están ordénanos como se muestra en la tabla del enunciado, primero la empresa, luego todos los locales y al final la casa del repartidor, además se asume que al pasar un vértice del conjunto V al conjunto C, este deja de estar en V, junto con el hecho de que el tiempo que se demora desde A-> B, es el mismo desde B->A.

```
C = {}                  //Conjunto de vértices visitados
V = {Vi, …, Vn}         //Conjunto de vértices por visitar
VF = Vn                 //Se almacena el vértice que representa la casa
                        en otro lugar temporalmente.
C = C U {V1}            //Se agrega la empresa al conjunto de elementos
                        visitados
Mientras (V no sea vacío) hacer
     Ordenar (V) de menor a mayor según el peso con respecto a Ci

     C = C U {V1}       //Se agrega el primer elemento (más cercano)
                        al conjunto de visitados
     i = i + 1
C = C U VF          //Se agregar el vértice de la casa al final del
                    conjunto
```

El algoritmo construido utiliza el método goloso y consiste en tener los datos del problema modelado en forma de un grafo, este grafo contiene un conjunto V que contiene a todos los vértices (locales) y una matriz de adyacencia A, que representa las aristas entre cada uno de los vértices y el valor que contiene representa el peso (tiempo) que tiene cada arista. Inicialmente se tiene un conjunto vacío C el cual contendrá los elementos visitados, estos se irán agregando al final, de manera que este conjunto estará ordenado en la manera en que se visitó cada local.

Antes de comenzar, almacenamos el vértice que representa la casa del repartidor, ya que este es el ultimo lugar a visitar y de esta manera nos aseguramos de que no sea seleccionado durante el proceso de entrega, luego se añade el primer elemento del conjunto V al conjunto C de elementos visitado, este elemento corresponde a la empresa. Ahora dentro

del ciclo comenzamos a visitar los distintos locales, primer se ordena de menor a mayor los elementos no visitados (conjunto V), con respecto al peso que tiene la arista que conecta cada elemento de V con el ultimo elemento agregado en C (C_i). De esta manera podremos lograr seleccionar siempre el local más cercano o con menor peso, al final del ciclo se aumenta el valor de i para que en el siguiente ciclo se comparen los pesos con el último elemento agregado.

Cuando ya se han agregado todos los elementos de V, solamente resta agregar el vértice que representa la casa (VF) al final del conjunto C, de esta manera el conjunto C contendrá todos los elementos (locales) ordenados según el orden en que se visitaron, representado esta la ruta que debe seguir el repartidor para minimizar el tiempo.

Si analizamos el algoritmo construido podemos obtener el siguiente orden de complejidad:

- Ordenar V según su peso con Ci depende del algoritmo de ordenamiento utilizado, pensado en que se quiere disminuir el orden de complejidad, se considera merge sort para el ordenamiento que posee un $O(n * \log(n))$ en el peor de los casos.
- Ciclo principal del algoritmo encargado de ordenar y seleccionar elementos, posee un $O(n^2)$.

Con lo anteriormente indicado, podemos concluir en que el algoritmo construido posee un orden de complejidad $O(n^2)$.

En cuanto al rendimiento del algoritmo, es posible decir que, debido a la utilización del método goloso, este nos asegura un buen rendimiento, pero no siempre la solución óptima, por lo que se podría mejorar utilizando un método que nos asegure tanto eficiencia como calidad de solución, como la programación dinámica.

Traza del ejemplo

Matriz de adyacencia

Local	Empresa	X	Y	T	Z	Casa
Empresa	0	20	30	15	25	35
X	20	0	10	25	15	20
Y	30	10	0	10	20	25
T	15	25	10	0	25	35
Z	25	15	25	25	0	30
Casa	35	20	25	35	30	0

Inicialmente de tiene los siguientes valores

C = {}

V = {Empresa, X, Y, T, Z, Casa}

i = 1

VF = {}

Ejecución

C = {Empresa}

VF = {Casa}

V = {X, Y, T, Z}

i = 1

- **Primer ciclo**

  C_1 = Empresa
  $V_{ordenado}$ = {T, X, Z, Y}
  $C_{resultante}$ = {Empresa, T}
  $V_{resultante}$ = {X, Z, Y}

- **Segundo ciclo**

  C_2 = T

$V_{ordenado}$ = {Y, X, Z}
$C_{resultante}$ = {Empresa, T, Y}
$V_{resultante}$ = {X, Z}

- **Tercer ciclo**

  C_3 = Y
  $V_{ordenado}$ = {X, Z}
  $C_{resultante}$ = {Empresa, T, Y, X}
  $V_{resultante}$ = {Z}

- **Cuarto ciclo**

  C_4 = Y
  $V_{ordenado}$ = { Z}
  $C_{resultante}$ = {Empresa, T, Y, X, Z}
  $V_{resultante}$ = { }

Se sale del ciclo y se agrega VF en C

$C_{resultante}$ = {Empresa, T, Y, X, Z, Casa}