Estructura de Datos y Algoritmos

Documento Reto 1

Universidad de los Andes

Simón Peña Alarcón - 202512907- s.penaa2@uniandes.edu.co

Juan Sebastian Chacón Ochoa-202513196-j.chacono@uniandes.edu.co

Manuel Santiago Figueroa-202511697- m.figueroag@uniandes.edu.co

2025

Requerimiento 1 por Juan Sebastian Chacón Ochoa-202513196

```python
def req_1(catalog, pasajeros):
    """
    Retorna el resultado del requerimiento 1
    """

    start_time = get_time()
    contador=0
    sumatoria_duracion=0
    sumatoria_costo_total=0
    sumatoria_distancia=0
    sumatoria_peajes=0
    sumatoria_propinas=0
    conteo_metodos_pago={}
    conteo_fechas_inicio={}

    for i in range(0,lt.size(catalog["taxis_info"])):
        viaje = lt.get_element(catalog["taxis_info"], i)
        pc = int(viaje["passenger_count"])
        if (pc != 0) and (pc == pasajeros):
            contador+=1
            pickup= viaje["pickup_datetime"]
            dropoff= viaje["dropoff_datetime"]
            start= pickup[11:16]
            finish= dropoff[11:16]
            h1str,m1str= start.split(":")
            h2str,m2str=finish.split(":")
```

```python
    else:
        duracion=costo_total=distancia=peajes=propina=0
metodo_frec=None
cantidad_maxima=-1
for metodo in conteo_metodos_pago:
    cantidad =conteo_metodos_pago[metodo]
    if cantidad >cantidad_maxima:
        cantidad_maxima=cantidad
        metodo_frec=str(metodo)+" - "+str(cantidad)
fecha_frec=None
max_fecha=-1
for fecha in conteo_fechas_inicio:
    cantidad=conteo_fechas_inicio[fecha]
    if cantidad>max_fecha:
        max_fecha=cantidad
        fecha_frec=fecha
end_time = get_time()
tiempo_ms = delta_time(start_time, end_time)
resultado={
    "tiempo_ejecucion_ms":tiempo_ms,
    "total_trayectos":contador,
    "duracion_promedio(min)":duracion,
    "costo_total_promedio":costo_total,
    "distancia_promedio_millas":distancia,
    "peajes_promedio":peajes,
    "metodo_pago_mas_usado":metodo_frec,
    "propina_promedio":propina,
    "fecha_inicio_mas_frecuente":fecha_frec
}
return resultado
```

Descripción

Este requerimiento recorre la lista de viajes y selecciona únicamente aquellos cuya cantidad de pasajeros coincide con el valor dado. Para cada viaje válido, acumula la duración, costo, distancia, peajes y propinas, además de registrar las frecuencias de métodos de pago y fechas de inicio. Al finalizar, calcula los promedios y determina los valores más frecuentes en métodos de pago y fechas.

Entrada	Estructura con la información de los taxis. Número de pasajeros a filtrar.
Salidas	Un diccionario con: Tiempo de ejecución. Total de trayectos encontrados. Duración promedio. Costo total promedio. Distancia promedio en millas. Promedio de peajes y propinas. Método de pago más usado. Fecha de inicio más frecuente.

Implementado (Sí/No)	Si Juan Sebastian Chacón Ochoa.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Recorrer todos los viajes en el catálogo	O(n)
Paso 2: Calcular duración, costos, distancias, peajes y propinas	O(1)
Paso 3: Actualizar conteo de métodos de pago y fechas	O(1)
Paso 4: Buscar el método de pago más usado	O(m)
Paso 5: Buscar la fecha de inicio más frecuente	O(f)
TOTAL	O(n)

Análisis

El requerimiento 1 cuenta con una complejidad temporal O(n) donde n corresponde al número de iteraciones. Esto se debe a que el programa recorre todos los datos para verificar la cantidad de pasajeros y acumular la información necesaria. Las operaciones se realizan en tiempo constante. Luego, se recorren los diccionarios de métodos de pago y fechas manteniendo la complejidad en O(n). En conclusión, el tiempo de ejecución crece de manera lineal con el tamaño de la entrada de los datos.

Requerimiento 2 por por Simón Peña Alarcón – 202512907

```python
def req_2(catalog, filtro):

    contador = 0
    duration = 0
    total_costs = 0
    distancia = 0
    peajes = 0
    pasajeros = {}
    propina = 0
    fechas = {}

    for each in range(0,lt.size(catalog["taxis_info"])):
        metodo = lt.get_element(catalog["taxis_info"],each)
        pm = str(metodo["payment_type"])
        if pm == filtro:
            contador += 1

            pickup = metodo["pickup_datetime"]
            dropoff = metodo["dropoff_datetime"]
            start = pickup[11:16]
            finish = dropoff[11:16]
            h1str , m1str = start.split(":")
            h2str, m2str = finish.split(":")
            h1 , m1 = int(h1str), int(m1str)
            h2, m2 = int(h2str), int(m2str)
            duration_sum = (h2 *60 + m2) - (h1 *60 + m1)
            duration += duration_sum

            total_costs += float(metodo["total_amount"])
```

```python
cantidad_max = -1
for num_pasajeros in pasajeros:
    cantidad = pasajeros[num_pasajeros]
    if cantidad > cantidad_max:
        cantidad_max = cantidad
        pasajero_frec = str(num_pasajeros)+ " - "+ str(cantidad)

fecha_frec = None
max_fecha = -1
for fecha in fechas:
    cantidad = fechas[fecha]
    if cantidad > max_fecha:
        max_fecha = cantidad
        fecha_frec = fecha

resultado = {
    "total_trayectos": contador,
    "duración_promedio(min)": duration,
    "costo_total_promedio": total_costs,
    "distancia_promedio_millas": distancia,
    "peajes_promedio": peajes,
    "pasajeros_mas_frecuente": pasajero_frec,
    "propina_promedio": propina,
    "fecha_finalizacion": fecha_frec
}
return resultado
```

Descripción

El requerimiento 2 consiste en filtrar los viajes de acuerdo con el método de pago especificado y, sobre esa base, calcular estadísticas como el número total de trayectos, la duración promedio, el costo promedio, la distancia promedio, los peajes, la propina, el número de pasajeros más frecuente y la fecha de finalización más común. Para ello, el algoritmo recorre cada registro, selecciona únicamente los que cumplen con el filtro y acumula la información necesaria en estructuras auxiliares.

Entrada	Catálogo de taxis con la información de los viajes. Parámetro filtro que corresponde al método de pago a evaluar.
Salidas	Un diccionario con las estadísticas calculadas: Total de trayectos. Duración promedio. Costo total promedio. Distancia promedio en millas. Peajes promedio. Pasajeros más frecuentes. Propina promedio.

	Fecha de finalización más frecuente.
Implementado (Sí/No)	Si por Simón Peña Alarcón

Análisis de complejidad

Pasos	**Complejidad**
Paso 1: Recorrer todos los viajes en el catálogo	O(n)
Paso 2: Calcular duración, costos, distancias, peajes y propinas	O(1)
Paso 3: Actualizar conteo de pasajeros y fechas	O(1)
Paso 4: Calcular promedios de las métricas acumuladas	O(1)
Paso 5: Buscar el número de pasajeros más frecuente	O(p)
Paso 6: Buscar la fecha de finalización más frecuente	O(f)
TOTAL	O(n)

Análisis

El requerimiento 2 tiene una complejidad temporal O(n), donde n corresponde al número de iteraciones. Esto se debe a que el programa recorre todos los datos para filtrar los trayectos según el tipo de pago y acumular la información correspondiente. Las operaciones dentro del ciclo se realizan en tiempo constante. Al finalizar, se recorren los diccionarios de pasajeros y fechas, los cuales en el peor caso pueden tener hasta n elementos, manteniendo la complejidad en O(n). En conclusión, el tiempo de ejecución crece de manera lineal con el tamaño de la entrada.

Requerimiento 3 por Manuel Santiago Figueroa- 202511697

```python
def req_3(catalog,valor_menor, valor_mayor):
    """
    Retorna el resultado del requerimiento 3
    """
    time_start = get_time()

    contador = {
        "tiempo_ejecucion": 0,
        "numero_viajes": 0,
        "tiempo_promedio":0,
        "precio_promedio_usd":0,
        "disatancia_total_promedio":0,
        "precio_peaje_promedio": 0,
        "cantidad_pasajeros_frecuente" : {},
        "cantidad_propinas_promedio": 0,
        "fecha_promedio": {}
        }

    for taxi in catalog["taxis_info"]["elements"]:
        lista = []
        if (float(taxi["total_amount"])) >= float(valor_menor) and (float(taxi["total_amount"])) <= float(valor_mayor):
            contador["numero_viajes"] +=1
            lista = [taxi["pickup_datetime"], taxi["dropoff_datetime"]]
            hora, minuto , segundo = lista[0][11:].split(":")
            hora_inicial = int(hora) *60 + int(minuto) + (int (segundo) / 60)
            hora_final, minuto_final, segundo_final = lista[1][11:].split(":")
            hora_termino = int(hora_final) *60 + int(minuto_final) + (int (segundo_final)/ 60)
```

Descripción

Esta función analiza los viajes de taxis cuyo costo total se encuentra dentro de un rango específico (valor_menor y valor_mayor). Calcula estadísticas como el tiempo promedio del viaje, precio promedio, distancia promedio, peajes promedio, propinas promedio, la fecha con más viajes y la cantidad de pasajeros más frecuente.

Entrada	catalog: diccionario con la información de los taxis y barrios. valor_menor: valor mínimo del costo del viaje a filtrar. valor_mayor: valor máximo del costo del viaje a filtrar.
Salidas	Diccionario con: tiempo_ejecucion: tiempo de ejecución de la función. numero_viajes: total de viajes que cumplen el filtro. tiempo_promedio: tiempo promedio por viaje (minutos). precio_promedio_usd: costo promedio por viaje. disatancia_total_promedio: distancia promedio de los viajes. precio_peaje_promedio: promedio de peajes pagados. cantidad_pasajeros_frecuente: número de pasajeros más frecuente y su frecuencia. cantidad_propinas_promedio: promedio de propinas. fecha_promedio: fecha con más viajes registrados.
Implementado (Sí/No)	Si por Manuel Santiago Figueroa

Análisis de complejidad

Pasos	Complejidad
Paso 1: Inicialización de variables y contador	O(1)
Paso 2: Iterar sobre todos los taxis en catalog["taxis_info"]["elements"] y filtrar por costo	O(n), siendo n = número de taxis
Paso 3: Calcular tiempo de viaje, actualizar contadores y sumas	O(1) por viaje, total O(n)
Paso 4: Calcular promedios dividiendo sumas por numero_viajes	O(1)
Paso 5: Encontrar la fecha con más viajes	O(m), m = número de fechas distintas
Paso 6: Encontrar la cantidad de pasajeros más frecuente	O(p), p = número de cantidades de pasajeros distintas
Paso 7: Calcular tiempo de ejecución	O(1)
TOTAL	O(n)

Análisis

El requerimiento 3 presenta una complejidad temporal O(n), donde n corresponde al número de iteraciones. El algoritmo recorre todos los registros para verificar si el costo total del viaje se encuentra dentro del rango dado y, en caso afirmativo, acumula la información necesaria. Todas las operaciones se realizan en tiempo constante. Al final, se procesan los diccionarios de fechas y pasajeros para identificar los valores más frecuentes y en el peor caso también puede requerir hasta n operaciones. En conclusión, el tiempo de ejecución crece de manera lineal con respecto al tamaño de la entrada.

Requerimiento 4

Descripción

La función analiza los viajes de taxis dentro de un rango de fechas (fecha_ini a fecha_fin) y determina la ruta entre barrios con el mayor o menor costo promedio, según el filtro (MAYOR o MENOR). Calcula estadísticas promedio de distancia, tiempo y costo para cada ruta entre barrios y retorna la ruta seleccionada con sus métricas.

Entrada	catalog: diccionario con información de taxis y barrios. fecha_ini: fecha inicial del filtro (formato %Y-%m-%d). fecha_fin: fecha final del filtro (formato %Y-%m-%d). filtro: "MAYOR" o "MENOR" para seleccionar la ruta con mayor o menor costo promedio.
Salidas	Diccionario con: tiempo_ejecucion: tiempo de ejecución de la función. filtro_seleccionado: filtro aplicado. numero_viajes_totales: número de viajes dentro del rango de fechas. barrio_inicio: barrio de inicio de la ruta seleccionada. barrio_final: barrio final de la ruta seleccionada. distancia_promedio: promedio de distancia de la ruta seleccionada. tiempo_promedio: tiempo promedio de viaje de la ruta seleccionada (minutos). costo_promedio: costo promedio de la ruta seleccionada.
Implementado (Sí/No)	Si se implementó.

Análisis de complejidad

Pasos	**Complejidad**
Paso 1: Inicialización de variables y diccionario	O(1)
Paso 2: Llamada a barrios(catalog) para obtener barrios filtrados	O(b), b = número de barrios
Paso 3: Iterar sobre todos los taxis en catalog["taxis_info"]["elements"]	O(n), n = número de viajes
Paso 4: Verificar si cada viaje está dentro del rango de fechas	O(1) por viaje → O(n)
Paso 5: Determinar barrio de inicio y final usando coordenadas y Haversine	O(k) por viaje, k = número de sub-barrios → O(n*k)

Pasos	Complejidad
Paso 6: Actualizar estadísticas por ruta	O(1) por viaje → O(n)
Paso 7: Calcular promedio de distancia, tiempo y costo por ruta	O(r), r = número de rutas distintas
Paso 8: Seleccionar ruta según filtro MAYOR o MENOR	O(r)
Paso 9: Calcular tiempo total de ejecución	O(1)
TOTAL	*O(n*m)*

Análisis

El requerimiento 4 presenta una complejidad temporal O(n*m), donde n corresponde al número de viajes y m al número de barrios. El algoritmo recorre todos los registros dentro del rango de fechas indicado y para cada viaje determina los barrios de origen y destino comparando las coordenadas con los polígonos de cada zona, lo que implica varios cálculos de distancia. Después acumula información de costo, tiempo y distancia, y al finalizar realiza una búsqueda del barrio de mayor o menor costo según el filtro seleccionado. Las operaciones finales sobre los diccionarios no son tan grandes, por lo que no afectan la complejidad general. En conclusión, el tiempo que tarda en ejecutarse aumenta a medida que crece el número de viajes revisados como la cantidad de barrios en los que se hacen las comparaciones, ya que ambos se combinan en el proceso.

Requerimiento 5

Descripción

El requerimiento analiza los viajes en un rango de fechas y agrupa la información por franjas horarias de una hora (según la hora de inicio del viaje). Posteriormente, identifica la franja con mayor o menor costo promedio, según el filtro elegido por el usuario. Finalmente, retorna estadísticas como número de trayectos, promedios de duración, costo, pasajeros, y los costos máximo y mínimo de esa franja.

Entrada	catalog: estructura que contiene los viajes (taxis_info). filter: criterio de selección, puede ser "MAYOR" o "MENOR". fecha_ini: fecha inicial del rango de consulta (formato YYYY-MM-DD). fecha_fin: fecha final del rango de consulta (formato YYYY-MM-DD).

Salidas	tiempo_ms: tiempo de ejecución del requerimiento. total_trayectos_filtrados: número de trayectos dentro del rango de fechas. franja_horaria: intervalo de horas con mayor/menor costo promedio. costo_promedio: costo promedio de los trayectos en la franja seleccionada. numero_trayectos: cantidad de trayectos en esa franja. duracion_promedio: duración promedio (minutos) de los trayectos en esa franja. pasajeros_promedio: número promedio de pasajeros. costo_mayor: costo máximo en esa franja. costo_menor: costo mínimo en esa franja.
Implementado (Sí/No)	Si se implementó.

Análisis de complejidad

Paso	Descripción	Complejidad
1	Recorrido de todos los viajes en taxis_info	**O(n)**
2	Comparación de fechas para filtrar cada viaje	**O(1)**
3	Cálculo de duración, costos, pasajeros y actualización de diccionario por hora	**O(1)**
4	Selección de la franja horaria con mayor/menor costo promedio (máx. 24 horas)	**O(1)**
Total	La complejidad está dominada por el recorrido de los viajes	**O(n)**

Análisis

El requerimiento 5 presenta una complejidad temporal O(N), donde n corresponde al número de iteraciones en el catálogo. El algoritmo recorre todos los viajes para verificar si la fecha se encuentra dentro del rango dado y en caso de que esto pase, acumula la información necesaria en un diccionario organizado por franjas horarias. Todas las operaciones realizadas dentro del ciclo son de tiempo constante. Después se evalúan como máximo 24 franjas horarias, lo cual es constante y no afecta la complejidad general. En conclusión, el tiempo de ejecución crece de manera lineal con respecto al tamaño de la entrada.

Requerimiento 6

Descripción

El requerimiento 6 calcula estadísticas de trayectos que inician en un barrio específico dentro de un rango de fechas. Determina el número total de viajes, distancia y tiempo promedio, barrio de destino más frecuente, así como estadísticas detalladas de medios de pago (frecuencia, recaudación, promedios de costo y tiempo).

Entrada	catalog: catálogo con información de taxis y barrios. fecha_ini: fecha de inicio (YYYY-MM-DD). fecha_fin: fecha de fin (YYYY-MM-DD). barrio: nombre del barrio de origen a evaluar.
Salidas	tiempo_ejecucion (ms). numero_viajes_totales. distancia_promedio. tiempo_promedio. nombre_barrio_final (destino más frecuente). medios_de_pago (con detalle de cada medio, si fue el más usado y/o el que más recaudó).
Implementado (Sí/No)	Si se implementó.

Análisis de complejidad

Paso	Descripción	Complejidad
1	Recorrer todos los viajes en taxis_info	**O(n)**
2	Filtrar viajes por rango de fechas	**O(1)**
3	Cálculo de tiempos (inicio y fin en minutos) y acumulación de promedios	**O(1)**
4	Determinar barrio de inicio y fin con distancia mínima (comparación con lista de barrios)	**O(m)**, donde *m* es el número de barrios por zona
5	Actualización de diccionario de medios de pago (acumulación de trayectos, costos y tiempos)	**O(1)**
6	Normalización de promedios en medios de pago	**O(k)**, donde *k* es la cantidad de medios de pago distintos

Paso	Descripción	Complejidad
7	Determinar barrio destino más frecuente	**O(d)**, donde *d* es la cantidad de barrios destino registrados
8	Determinar medio de pago más usado y con mayor recaudación	**O(k)**
TOTAL	Dominado por el recorrido de viajes y cálculos por barrio	**O(n·m)**

Análisis

El requerimiento 6 presenta una complejidad temporal O(n*m), donde n corresponde al número de viajes y m al número de barrios. El algoritmo recorre todos los registros para filtrar los que se encuentran dentro del rango de fechas y para cada viaje evalúa las coordenadas de inicio y final relacionándola con los polígonos de los distintos barrios, lo que implica cálculos adicionales de distancia. Estas operaciones de comparación y distancia se repiten para cada barrio y cada viaje, por lo que la complejidad depende de ambas cosas. Al finalizar, el algoritmo realiza algunas operaciones para sumar valores y encontrar los máximos valores en estructuras pequeñas, lo cual no impacta en el tiempo total. En conclusión, el tiempo de ejecución depende de la combinación entre la cantidad de viajes recorridos y la cantidad de barrios evaluados.