PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA Y ESTRUCTURA DE DATOS

2da. práctica (tipo B)
(Segundo Semestre 2024)

Duración: 1h 50 min.

- **No puede utilizar apuntes, solo hojas sueltas en blanco.**
- En cada función el alumno deberá incluir, a modo de comentario, la forma de solución que utiliza para resolver el problema. De no incluirse dicho comentario, el alumno perderá el derecho a reclamo en esa pregunta.
- No puede emplear plantillas o funciones no vistas en los cursos de programación de la especialidad.
- Los programas deben ser desarrollados en el lenguaje C++. Si la implementación es diferente a la estrategia indicada o no la incluye, la pregunta no será corregida.
- Un programa que no muestre resultados coherentes y/o útiles será corregido sobre el 50% del puntaje asignado a dicha pregunta.
- Debe utilizar comentarios para explicar la lógica seguida en el programa elaborado. El orden será parte de la evaluación.
- **Solo está permitido acceder a la plataforma de PAIDEIA, cualquier tipo de navegación, búsqueda o uso de herramientas de comunicación se considera plagio por tal motivo se anulará la evaluación y se procederá con las medidas disciplinarias dispuestas por la FCI.**
- Para esta evaluación solo se permite el uso de las librerías **iostream, iomanip, climits cstring, cmath o fstream**
- Su trabajo deberá ser subido a PAIDEIA.
- **Es obligatorio usar como compilador NetBeans.**
- Los archivos deben llevar como nombre su código de la siguiente forma codigo_LAB2_P# (donde # representa el número de la pregunta a resolver)

Pregunta 1 (10 puntos)

Debido al incremento de la delincuencia en la ciudad, las instituciones encargadas del control de la seguridad han decidido formar grupos a cargo de las investigaciones, por tal motivo necesitan un algoritmo que les ayude a conformar los equipos. Estos equipos pueden contar con un solo efectivo o más. Cuando el equipo tiene mas de un efectivo el grupo debe tener como primer efectivo el de mayor grado que el resto, si en caso los grados de los efectivos es el mismo se debe considerarse el de mayor tiempo. Para conformar los grupos debe considerar el orden en que han sido registrados los efectivos por tal motivo deben ser consecutivos, por ejemplo, si cuenta con 3 efectivos no puede formar un equipo con el efectivo 1 y 3 solamente, para que sea válido debe considerar el efectivo 2, desde luego si cumple con ser menor grado que el efectivo 1. A continuación, algunos ejemplos:

Datos de entrada:

Efectivo	Grado	Tiempo
1	2	4
2	2	1
3	2	4

Solución:

Se pueden formar 4 agrupaciones.

Las agrupaciones serían: {1}, {2}, {3}, {1,2}

Datos de entrada:

Efectivo	Grado	Tiempo
1	1	1
2	2	3
3	2	5
4	1	2

Solución:

Se pueden formar 5 agrupaciones

Las agrupaciones serían: {1}, {2}, {3}, {4}, {3,4}

Datos de entrada:

Efectivo	Grado	Tiempo
1	3	1
2	2	1
3	2	4

Solución:

Se pueden formar 5 agrupaciones.

Las agrupaciones serían: {1}, {2}, {3}, {1,2}, {1,2,3}

Se le pide:

a) Defina las estructuras necesarias para la solución del problema. (2 puntos)
b) Elabore un programa que empleando una función iterativa muestre cuantos grupos se pueden formar. No es necesario que se muestren los grupos a formar (8 puntos).

En esta pregunta debe usar una pila auxiliar, los datos deben ingresarse en un arreglo de forma predefinida. No puede usar iteraciones que anidadas donde ambas recorran los n datos.

Pregunta 2 (10 puntos)

En un servicio de entrega por delivery, se desea organizar una lista de pedidos en función del tiempo estimado total de cada pedido, que incluye el tiempo de preparación y el tiempo estimado de viaje. El tiempo de preparación varía según la complejidad del pedido y si hay disponibilidad del personal. Además, el tiempo estimado de viaje depende de la distancia al destino y si es hora punta. La velocidad promedio de los repartidores es de 45 km/h.

Se pide implementar un algoritmo que ordene los pedidos de menor a mayor tiempo estimado total, utilizando una lista de estructuras **Pedido** con los siguientes campos:

- id (cadena de caracteres de 4 dígitos): el identificador del pedido.
- tiempoPreparacion (entero): tiempo en minutos, es la complejidad de la Preparación más el tiempo que se adiciona si no se cuenta con la disponibilidad del personal.
- tiempoEstimadoViaje (entero): tiempo que tarda en llegar a la dirección de entrega más el tiempo que se adiciona si es hora punta.

Para cargar los datos a la estructura se cuenta con la siguiente información de los pedidos, la lectura la puede realizar desde un archivo o de alguna otra forma conveniente:

```
P001, m, 1, 12.5, 0
P002, b, 0, 6.3, 0
P003, a, 0, 8.0, 0
P004, b, 1, 10.5, 1
P005, m, 0, 15.5, 0
```

En cada línea se muestra la siguiente información:

- El identificador del pedido (cadena de caracteres de 4 dígitos).
- La complejidad de la preparación: puede tomar los valores 'b' (baja), 'm' (media) o 'a' (alta), que definen el tiempo de preparación en 10, 20, o 30 minutos respectivamente.
- Hay disponibilidad del personal: si es 0, se añaden 5 minutos adicionales al tiempo de preparación.
- Distancia a la dirección: la distancia en kilómetros hasta la dirección de entrega.
- Es hora punta: si es 1, se añaden 10 minutos al tiempo estimado de viaje.

Se deben realizar por lo menos las siguientes tareas:

1. **cargarDatosLista:** Esta función recibe la información sobre los pedidos y carga una lista de Pedidos.
2. **ordenarLista:** Ordena los pedidos usando el método de la raíz (Radix Sort), que ordena los elementos en función de los dígitos de su valor total. La clave de ordenación para cada pedido es el tiempo de espera (la suma del tiempo de preparación y del tiempo estimado de viaje).

Método de la Raíz (Radix Sort):

El método de la raíz es un algoritmo de ordenación que clasifica los elementos basándose en los dígitos individuales de sus valores, procesando los de menor a mayor (es decir, de las unidades a las decenas, centenas, etc.).

Los pedidos al inicio son los siguientes:

```
[P001 36, P002 33, P003 45, P004 34, P005 55]
```

En este caso, el tiempo total de cada pedido es la **clave** que vamos a usar para ordenar. Cada clave tiene dos dígitos, por lo que la ordenación raíz se hará en dos pasadas:

1. **Primera pasada:** Se ordenan los pedidos según el dígito de las unidades.
2. **Segunda pasada:** Se ordenan los pedidos según el dígito de las decenas.

Primera pasada (dígito de las unidades):

En la primera pasada, colocamos los pedidos en diferentes (listas) según el dígito de las unidades del tiempo total:

- **Dígito 3:** P002 (33)
- **Dígito 4:** P004 (34)
- **Dígito 5:** P003 (45), P005 (55)
- **Dígito 6:** P001 (36)

Después de la primera pasada, los pedidos se reorganizan en este orden:

```
[P002 33, P004 34, P003 45, P005 55, P001 36]
```

Segunda pasada (dígito de las decenas):

En la segunda pasada, los pedidos se organizan nuevamente, esta vez según el dígito de las decenas:

- **Dígito 3:** P002(33), P004 (34), P001 (36)
- **Dígito 4:** P003 (45)
- **Dígito 5:** P005 (55)

Después de la segunda pasada, los pedidos están completamente ordenados:

```
[P002 33, P004 34, P001 36, P003 45, P005 55]
```

Consideraciones:

La lista de los pedidos debe quedar ordenada al final del proceso y es donde se van colocando los pedidos después de cada pasada. Los pedidos no se pueden duplicar en ningún momento.

No olvide destruir las listas que deja de utilizar al final de todo el proceso.

Al finalizar el laboratorio, comprima la carpeta de su proyecto empleando el programa Zip que viene por defecto en el Windows, **no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares**. Luego súbalo a la tarea programa en Paideia para este laboratorio.

Profesores del curso:

Ana Roncal
Fernando Huamán
David Allasi
Heider Sánchez
Rony Cueva

San Miguel, 12 de octubre del 2024