Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 – Estructuras de Datos y Algoritmos

2024 - 1

Tarea 0

Fecha de entrega código: 27 de marzo

Link a generador de repos: CLICK AQUI

Objetivos

- Comprender las diferencias entre Arrays y Listas ligadas
- Familiarizarse con el uso de punteros y manejo de memoria
- Aplicar algoritmos de sorting y tecnicas de implementación

Introducción

¡Bienvenido a PLANET EDDXPRESS! Nuestra misión es entregar pedidos desde el planeta tierra al más allá, sin importar que tan peligroso o bajo nivel tenga que ser la entrega.

Están a punto de adentrarse en un viaje intergaláctico para entregar pedidos en diversos planetas, mientras que en el vasto cosmos se libran épicas guerras espaciales. En este universo lleno de tecnología avanzada, seres extraterrestres y robots inteligentes, su tarea será clave para el éxito de la entrega de pedidos y el mantenimiento de la paz interestelar.

Sus habilidades en programación serán fundamentales para diseñar sistemas eficientes de registro y contabilización de pedidos, así como para optimizar las rutas de entrega en este vasto y complejo espacio. Pero, ¡cuidado! En su camino se encontrarán con desafíos inesperados, desde planetas hostiles hasta piratas espaciales y amenazas intergalácticas.

Desplieguen sus habilidades, confíen en su ingenio y prepárense para vivir una experiencia que los llevará más allá de las estrellas. ¡Que comiencen la aventura en el universo EDDXPRESS!

Problema

Como programador estrella de Planet EDDxpress, tu primera tarea será programar la herramienta de planificación de entregas para naves repartidoras. Estas reciben diferentes pedidos que deben llevarlos los planetas del sistema.

Entidades

Planetas

Para esta simulación existen N planetas los cuales identificaremos con números desde 0 hasta $N-1$.

Naves

Al igual que en el caso de los planetas, existe una cantidad fija K de naves, enumeradas desde 0 hasta $K-1$. Las naves tienen una cola First In First Out (FIFO) de pedidos. Es decir, los primeros que llegan deben ser los primeros en ser repartidos, a menos de que se te indique lo contrario.

Pedidos

A lo largo de la simulación, se te irán dando pedidos para que las naves los entreguen a los diferentes planetas. Cada pedido tiene un identificador numérico único, una nave que lo posee, y un planeta a ser entregado.

Parte 1: Registrando pedidos

0. Inicializar naves y planetas

En un inicio, se dará la cantidad de planetas N y naves K que existen, para luego procesar los diferentes E eventos que vayan ocurriendo. Puedes ver un ejemplo de como serán entregados en la sección Input.

Inicializa las naves y planetas, para luego solo leer cada evento, y cerrar la simulación sin problemas.

1. `REGISTRAR-PEDIDO pedidoID naveID planetaID`

Este evento registra un nuevo pedido indicando que nave lo va a llevar y a que planeta tiene que ser entregado. Correspondientemente, este evento recibe 3 **números**: `pedidoID`, `naveID` y `planetaID`.

Por ejemplo, si la instrucción es

input

```
1 REGISTRAR-PEDIDO 0 1 2
```

esto significa que el pedido 0 debe ser llevado por la nave 1 al planeta 2.

Cada vez que ocurra este evento tu programa debe imprimir el ID del pedido agregado de la siguiente manera:

output

```
1 REGISTRADO PEDIDO pedidoID
```

2. `REPORTE-NAVE naveID`

Este evento tiene como finalidad imprimir la cantidad de pedidos pendientes de una nave en específico. Cada vez que se llame a este comando tu programa debe imprimir

```
output
1 PEDIDOS naveID PENDIENTES: X
```

siendo X la cantidad de pedidos pendientes de la nave indicada.

En caso de no haber pedidos pendientes para esa nave, se debe imprimir lo siguiente:

```
output
1 LA NAVE naveID NO TIENE PEDIDOS PENDIENTES
```

3. `REPORTE-PEDIDOS`

Este evento debe mostrar todos los pedidos pendientes de manera ordenada. Debe seguir la siguiente estructura al imprimir:

```
output
 1 REPORTE-PEDIDOS
 2     NAVE 0
 3         PEDIDO 35 CON PLANETA 0
 4         PEDIDO 23 CON PLANETA 1
 5         PEDIDO 56 CON PLANETA 0
 6         ...
 7     NAVE 1
 8         ...
 9     ...
10 TOTAL DE PEDIDOS: X
```

Donde las naves por ID y los pedidos se muestran como están ordenados en la cola de prioridad FIFO. Si una nave no tiene pedidos pendientes, esa nave no se debe imprimir.

Parte 2: Entregar pedidos

Como ya has podido registrar todos los pedidos para todas las naves y los planetas, ahora es necesario simular la entrega de los pedidos.

1. PEDIDO-CONTAMINADO naveID pedidoID

Se identificó un pedido como contaminado en una nave en específica, por lo cual deberás eliminarlo de los pedidos de la nave, y luego imprimir:

```
output
PEDIDO pedidoID HA SIDO ELIMINADO
```

En caso de que el pedido no esté en la nave estipulada o simplemente no exista, se debe imprimir:

```
output
PEDIDO pedidoID NO ENCONTRADO EN NAVE naveID
```

Notar que el pedido contaminado no será de carácter entregado, para evitar casos borde.

2. ENTREGAR-PEDIDOS

Este evento debe simular la entrega de pedidos para todas las naves, porque, ¿cuál es el punto de tener muchas naves si no puedes hacer entregas simultaneas?

Cada nave debe viajar al planeta de destino de su pedido más prioritario y entregarlo. Para aprovechar el tiempo la nave, debe además entregar todos los paquetes que tenga como destino ese el planeta del pedido más prioritario. Una vez entregado un pedido, se debe imprimir una línea que indique su correcta entrega:

```
output
PEDIDO pedidoID ENTREGADO EN PLANETA planetaID
```

El orden de entrega de las naves debe ser desde la de indice menor hasta la de indice mayor. Además considerar que si no existen pedidos por entregar, es decir, ninguna nave entrega un pedido, se debe imprimir el siguiente mensaje:

```
output
NO HAY PEDIDOS POR ENTREGAR
```

3. REPORTE-PLANETAS

Esta acción se dedica a imprimir los planetas por orden de mayor a menor, según la cantidad de pedidos correctamente entregados. Al terminar de ordenar, debe imprimir los planetas en ese orden con su respectiva cantidad de pedidos. Ejemplo:

```
output
PLANETAS-ORDENADOS
    PLANETA 3: 300 pedidos
    PLANETA 1: 10 pedidos
    PLANETA 0: 5 pedidos
    PLANETA 2: 0 pedidos
    ...
TOTAL-PEDIDOS-ENTREGADOS: 315
```

En caso de empate, queda primero el planeta con menor id (estable).

4. `TOMAR-DESPUES-MAX pedidoID naveID planetaID`

Similar a `REGISTRAR-PEDIDO`, pero añade el pedido atrás del pedido con mayor Id que posee actualmente la nave.

Por ejemplo, si la nave al que se añade el pedido, tiene la cola de pedidos como $3, 8, 5, 6$, y se busca añadir un nuevo pedido N, la nueva cola de pedidos quedará como $3, 8, \text{N}, 5, 6$.

Si la nave no tiene pedidos, se añade como único pedido a la cola.

Este evento genera el mismo output que `REGISTRAR-PEDIDO`.

Parte 3: Trucos estelares

1. `INVERSO naveID`

Frecuentemente las naves pasan por agujeros de gusano sin darse cuenta. Estos agujeros tienen el efecto de invertir todo tipo de orden. Desgraciadamente, como los tripulantes de la nave no nos creen cuando les decimos que tienen que darle la vuelta a sus pedidos, nosotros solo nos resignamos a registrar este cambio.

Por lo tanto, cuando recibas este comando, debes invertir la lista de pedidos de la nave indicada. Es decir, el orden de la cola se deberá invertir, y los primeros registrados deberán considerarse como últimos. El comportamiento de la cola deberá seguir siendo FIFO para nuevos pedidos.

Por ejemplo, si está la cola $1, 2, 3$, al invertirla, quedará como $3, 2, 1$, y al tomar un pedido 4, será $3, 2, 1, 4$.

Este comando no requiere de una impresión de output.

2. `COORDINAR-PEDIDOS nave1ID nave2ID planetaID`

Para optimizar la entrega de pedidos, dos naves pueden coordinar pedidos para que una nave entregue los pedidos de un planeta en específico. Cuando ocurre este evento, deberás traspasar los pedidos del planeta indicado, entre las naves indicadas, de manera que desde la nave con menor cantidad de pedidos pendientes de ese planeta se transfiera hacia la nave con mayor cantidad. Si las dos naves tienen la misma cantidad de paquetes, la nave 2 debe pasarle los pedidos a la nave 1.

Finalmente debes retornar la cantidad de pedidos pendientes totales (considerando todos los planetas) con los que quedan las naves. Es decir, si fueron traspasados de la nave 1 a la nave 2 un total de 4 paquetes que dejan a la nave 1 con 2 paquetes restantes y a la nave 2 con 19 paquetes restantes, entonces el output esperado es el siguiente:

```
output
1 PAQUETES TRANSFERIDOS: 4
2     NAVE nave1ID: 2 PEDIDOS
3     NAVE nave2ID: 19 PEDIDOS
```

Ejecución

Tu programa se debe compilar con el comando `make` y debe generar un ejecutable de nombre `ship` que se ejecuta con el siguiente comando:

```
./ship input output
```

donde `input` será un archivo con los eventos a simular y `output` el archivo donde se guardarán los resultados.

En caso de querer correr el programa para ver los leaks de memoria utilizando valgrind deberás utilizar el siguiente comando:

```
valgrind ./ship input output
```

Tu tarea será ejecutada con archivos de creciente dificultad, asignando puntaje a cada una de estas ejecuciones que tenga un output igual al esperado. A continuación detallaremos los archivos de input y output.

Input

El archivo de input comenzará con el número de `Planetas N`, la siguiente línea sería un número `K` que representa la cantidad de `naves`. Por último, recibirás un número `E` que corresponde al número de eventos a recibir. (Esto incluye todo lo relacionado a los eventos, ya sea registros de pedidos, entregas, consultas, operaciones, etc). Se verían con el siguiente formato:

input

```
N
K
E
REGISTRAR-PEDIDO 1 1 1
REPORTE-NAVES
REPORTE-PENDIENTES 2
REGISTRAR-PEDIDO 2 3 2
ENTREGAR-PEDIDO 1
...
```

En este input describe `N` planetas, y `K` naves, y `E` eventos en total.

Output

El output deberá consistir de un archivo con la información solicitada por cada evento en el archivo de input. Ojo, ten en consideración los saltos de línea (`\n`), donde todas las lineas terminan con un salto. Así también, considera los espacios de cada impresión para que tu output coincida con el de los casos de prueba.

Informe

Además del código de la tarea, se debe redactar un **breve** informe que explique como se pueden implementar algunas estructuras y algoritmos del enunciado. Les recomendamos **comenzar la tarea escribiendo este informe**, ya que pensar y armar un esquema antes de pasar al código ayuda a estructurarse al momento de programar y adelantar posibles errores que su solución pueda tener.

Este informe se debe entregar en un archivo PDF escrito usando LaTeX junto a la tarea (por lo cual tienen la misma fecha de entrega), con nombre `informe.pdf` en la raíz del repositorio. En este se debe explicar al menos los siguientes puntos:

1. Principales estructuras en el enunciado, sus características y como se pueden representar.
2. Como implementar los comandos `PEDIDO-CONTAMINADO` y `COORDINAR-PEDIDOS`.
3. ¿Cómo se puede invertir una lista ligada?
4. La acción `REPORTE-PLANETAS` requiere que se imprima de mayor a menor según la cantidad de pedidos. ¿Cómo se puede implementar este ordenamiento?

Se encuentra un template en este link para el uso en la tarea.

En la documentación de Overleaft se indica como clonar un template.

No se aceptarán informes de más de 2 planas, informes ilegibles, o generados con inteligencia artificial.

Evaluación

La nota de tu tarea es calculada a partir de testcases de Input/Output y además de un breve informe escrito en LaTeX que explique un modelamiento sobre como abarcar el problema, las estructuras de datos a utilizar, etc. La ponderación se descompone de la siguiente forma

- 20 % Por el informe de modelamiento, este debe incluir:
  - 5 % Por incluir máximo un parrafo con la motivación del problema, es decir, que lo hace relevante y las posibles aplicaciones.
  - 10 % Por el informe como tal, explicando el modelamiento y las estructuras de datos a utilizar.
  - 5 % Por una conclusión de máximo un párrafo donde cierren su idea y resuman como es que su modelamiento va de la mano con el problema a investigar.
- 70 % a la nota entre las partes 1, 2 y 3
  - 25 % a la nota de los tests de la parte 1
  - 30 % a la nota de los tests de la parte 2
  - 15 % a la nota de los tests de la parte 3
- 10 % Por manejo de memoria
  - 5 % Por no poseer leaks de memoria
  - 5 % Por no poseer errores de memoria

Para cada test de evaluación, tu programa deberá entregar el output correcto en **menos de 10 segundos** y utilizar menos de 1 GB de RAM[1]. De lo contrario, recibirás 0 puntos en ese test.

Recomendación de tus ayudantes

Las tareas generalmente requieren de mucha dedicación, por lo que desde ya te recomendamos distribuir tu tiempo considerando los plazos definidos. Así mismo, te recomendamos fuertemente que antes de empezar a programar tu tarea, leas el enunciado detenidamente y te dediques a entender de manera profunda lo que te pedimos. Una vez que hayas comprendido el enunciado, dedica el tiempo que sea necesario para la planificación, modelación y elaboración de de tu informe, para posteriormente poder programar de manera más eficiente. Estos son consejos de tus ayudantes que te pueden ayudar a pasar el ramo.

Entrega

Código: GIT - Rama `master` del repositorio asignado. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

Atraso: Esta tarea no considera cupones ni política de atrasos, cualquier atraso debe ser justificado con el equipo docente del curso.

Integridad académica

Este curso se adscribe al Código de Honor establecido por la Escuela de Ingeniería. Todo trabajo evaluado en este curso debe ser hecho **individualmente** por el alumno y **sin apoyo de terceros**. Se espera que los alumnos mantengan altos estándares de honestidad académica, acorde al Código de Honor de la Universidad. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Pregrado de la Escuela de Ingeniería.

[1] Puedes revisarlo con el comando `htop` u ocupando `valgrind --tool=massif`