Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2133 - Estructura de datos y algoritmos

Pauta Tarea 0

10 de abril de 2022

1º semestre 2022 - Profesores Yadran Eterovic, Mario drogget

Estructuras 4pts

- Restaurant (1 pt)
  Los resturants son **struct**, que contiene un **id**(int), **cantidad de mesas**(int) y un arreglo de punteros hacia **mesas**(structs).
- Mesas (1 pt)
  Las Mesas son **struct**, estos contienen **id**(int), **capacidad**(int) , **cantidad clientes**(int), **clientes**(array de punteros)
- Cliente (1 pt)
  Los Clientes son **struct**, estos tiene **id**(int) , **cuenta**(int) , **cantidad platos**(int), **Platos**(Puntero a Struct Plato)
- Plato (1 pt)
  El Plato es un **Struct** que tiene de atributos **id**(int), **precio**(int), **next**(Puntero a Struct Plato), **prev**(Puntero a struct Plato).
  Como vemos por la modelacion este struct tiene comportamiento de Lista Ligada.
- Estructura General.
  A modo general tenemos N_k0 restaurants, que se inicializan como array, cada restaurant posee N_k1 Mesas, inicializadas de la misma forma, estas mesas a su vez poseen capacidad para N_k_3 clientes, finalmente estos a su vez guardan sus platos, a modo de lista ligada.

Consideraciones :

Asignar un punto al mencionar por que deben utilizar Lista Ligada y arrays cuando corresponde. (limitaciones de C y/o temas de eficiencia al indexar).
Un Alumno pudo haber realizado la carta como un struct, no existe puntaje por este item.

En caso de que el alumno haya utilizado LL para otra estructura que no sea **Platos** asignar puntaje parcial (0.5 PTS), de manera análoga para los arrays.
Si el Alumno posee estructuras adicionales y/o le faltan estructuras, queda a criterio del corrector. (ver si hace sentido la modelacion).

Complejidades teóricas :

Eventos : 9 pts

- Para los eventos {OPEN-TABLE, MENU-ITEM}
  - Se calcula la complejidad del evento en base a las operaciones, se nota que son operaciones constantes $\mathcal{O}(1)$ para todos los casos. (0.4 pts por evento)
  - Se justifica correctamente que son operaciones constantes. (0.6 pts por evento)
    - Desgloce eventos.
    - OPEN-TABLE consta en inicializar memoria en el heap $\mathcal{O}(1)$, accediendo a directamente a índices de array $\mathcal{O}(1)$, en este caso particular es acceder al array de punteros del restaurant.
    - MENU-ITEM consta en inicializar memoria en el heap $\mathcal{O}(1)$, accediendo a directamente a índices de array $\mathcal{O}(1)$, en este caso se agrega de forma directa en el array de la carta.
- Para el evento Customer.
  - Sea n la cantidad de asientos que posee dicha mesa, se debe recorrer el array hasta encontrar un asiento vacio, por lo que la complejidad es $\mathcal{O}(n)$
  - 0.4pts por el calculo correcto de la complejidad, 0.6pts por la explicacion dada. No se otorga puntaje parcial.
- Para el evento Table Status.
  - sea n la cantidad de asientos en la mesa, en caso, acceder a la mesa en particular es $\mathcal{O}(1)$ ya que tenemos los indices del array, recorrer toda la mesa para chequear los clientes es $\mathcal{O}(n)$, por lo que la complejidad del evento es $\mathcal{O}(n)$.
  - 0.4pts por el calculo correcto de la complejidad, 0.6pts por la explicación dada. 0.3 pts en caso de tener bien la complejidad de recorrer la silla, teniendo mala la complejidad final (ya que no considera $\mathcal{O}(1)$ acceder a esta.
- Para los eventos Order-CREATE Order-CANCEL
  - acceder hasta el cliente para agregar un plato es $\mathcal{O}(1)$, ya que tenemos todos los índices de los array para acceder mediante los punteros, acá depende de la

implementación de la LL podemos tener $\mathcal{O}(n)$, con n el tamaño de la LL o $\mathcal{O}(1)$, si guardan referencia a tail en su struct.

- 0.4pts por el calculo correcto de la complejidad, 0.6pts por la explicación dada para cada evento.
  Ambas implementaciones, con y sin tail, se consideran correctas y se otorga puntaje completo.
  0.3 pts en caso de tener bien la complejidad de recorrer agregar el plato, teniendo mala la complejidad final (ya que no considera $\mathcal{O}(1)$ acceder al cliente.

- BILL-CREATE

  - llegar hasta la mesa tiene complejidad $\mathcal{O}(1)$, debido a que tenemos las ref de los punteros directos, acá debemos recorrer los n asientos de la mesa, debemos recorrer cada LL de platos de los clientes para sumarlos, siendo p la cota asintótica de estas LL la complejidad de esta implementación es $\mathcal{O}(n*p)$, lo cual no podemos acotar.
  - 0.4pts por el calculo correcto de la complejidad, 0.6pts por la explicación dada. Ambas implementaciones se consideran buenas. No se otorga puntaje parcial.

- CHANGE-STEATS

  - No sabemos donde están sentados los clientes, por lo si tenemos una mesa con n asientos, debemos recorrer la mesa buscando los clientes, lo que es $\mathcal{O}(n)$, luego se realiza un cambio en los punteros de los respectivos asientos, esta asignación es $\mathcal{O}(1)$

- PERROU-MUERTO

  - en una LL simple, es solo una resignación de punteros simple $\mathcal{O}(1)$, en caso de guardar tail y/o head se debe actualizar toda la LL del cliente que se hizo **el larry**, siendo n la cantidad de platos que tenia **el larry**, la complejidad del evento seria $\mathcal{O}(n)$.
  - 0.4pts por el calculo correcto de la complejidad, 0.6pts por la explicacion dada. Ambas implementaciones se consideran buenas. No se otorga puntaje parcial.

LL VS Arrays 3 pts

- menciona correctamente 3 ventajas y desventajas de cada estructura 3pts.
  se otorga puntaje parcial por estructura, 1.5 pts cada estructura, puntaje parcial minimo 0.5 pts

C gatta go fast 2pts

- logra mencionar de forma clara y precisa las diferencias entre C y python, menciona que es un lenguaje de bajo nivel y como esto nos otorga una ventaja de velocidad. Se otorga puntaje parcial a criterio del corrector