UNIVERSIDAD METROPOLITANA

Departamento de Gestión de Proyectos y Sistemas.
Estructuras de datos. BPTSP06
Prof. Antonio Guerra, Luís Stanislao y Christian Guillén Drija

Hotel Booking: Tablas de Dispersión (Hash Tables) y Árboles Binarios de Búsqueda.

Contexto preliminar:

En el siguiente proyecto, usted tendrá que realizar una investigación documental que le permita obtener información sobre el contexto del problema. En tal sentido, se sugiere que comience por realizar la siguiente lectura relativa a Hash Tables, pero tome en cuenta que es solo un recurso inicial que debe ser complementado con la búsqueda autónoma de información por parte de los integrantes del equipo de trabajo:

https://www.hackerearth.com/practice/data-structures/hash-tables/basics-of-hash-tables/tutorial/

El problema:

Un reconocido Hotel de la Ciudad ha contratado a su equipo de trabajo para construir un sistema en Java para la administración de las reservas. El sistema deberá implementar una solución con uso eficiente de los recursos del computador.

Requerimientos funcionales:

1. **Registro de clientes.** El usuario podrá conocer los clientes que actualmente se encuentran hospedados en el hotel; para lo cual se deberá introducir el apellido del cliente y el nombre, para luego obtener el número de la habitación en la que se encuentra alojado. La solución para tal requerimiento deberá tener una complejidad en tiempo lo más cercano posible a O(1).

2. Búsqueda de reservación: Dada la cédula de identidad del cliente, el recepcionista podrá conocer los datos de la reservación, en caso de existir. La solución para tal requerimiento deberá tener una complejidad en tiempo de O(log N).

3. Historial de habitación: dado el número de una habitación, el sistema emitirá una lista de los clientes que alguna vez se hospedaron en ella. La solución para tal requerimiento deberá tener una complejidad en tiempo lo más cercado a O(log N). Deberá utilizar un ABB.

4. Check-in: Un cliente con reservación llega para comenzar su estadía. En ese momento el recepcionista le asignará una habitación de manera que la reservación para al registro de los huéspedes del hotel.

5. Check-out: Al terminar su estadía, el recepcionista cierra el registro del cliente, quedando libre la habitación y actualizándose el historial de la habitación utilizada.

Requerimientos Técnicos:

1. La solución debe ser implementada con base en la utilización de tablas de dispersión y ABB. Puede, sin embargo, utilizar otros TDA auxiliares.
2. La aplicación debe ofrecer una interfaz gráfica al usuario.
3. **Al comenzar el sistema, los datos de las reservaciones, de los huéspedes y del histórico de las habitaciones debe cargarse automáticamente. En una hoja de cálculo podrá encontrar la información requerida para cumplir con este requerimiento: datos**

Consideraciones:

- Los proyectos podrán ser sometidos a defensa, es decir, el facilitador convocará al equipo para una revisión presencial.
- Los equipos de trabajo deberán utilizar GitHub para el control de versiones y facilitar el trabajo en equipo de manera remota. En el registro se deberá reflejar la participación activa y significativa de los integrantes. Los proyectos que no tengan interfaz gráfica, serán calificados con **0 (cero)**.
- Los proyectos que sean iguales o parecidos, serán calificados con 0 (cero).
- Los programas que “no corran”, serán calificados con 0 (cero).
- Los equipos pueden tener como máximo 3 personas.
- El nombre del archivo del proyecto deberá contener los apellidos de los integrantes del equipo.

Criterios de evaluación

- *Funcionalidad:* Capacidad para proporcionar las funcionalidades que satisfacen las necesidades explícitas e implícitas bajo unas ciertas condiciones. **(40%)**
    - *Adecuación:* El programa ofrece todas funcionalidades que respondan a las necesidades, tanto explícitas (contenidas en el documento descriptivo del proyecto) como implícitas; entendiendo como necesidades implícitas, aquellas que, no estando descritas en el documento, surgen como resultado de un concienzudo análisis del problema planteado y que aseguran el correcto funcionamiento del programa.
    - *Exactitud*: El programa genera los resultados o efectos correctos o acordados, con el grado necesario de precisión.
- *Fiabilidad*: Capacidad para mantener un nivel especificado de prestaciones cuando se usa bajo ciertas condiciones.
    - *Madurez*: El programa no presenta fallas originadas por errores de programación, análisis o diseño. **(10%)**
    - *Tolerancia a fallos*: El programa responde adecuadamente al manejo inadecuado del usuario; es decir, mantiene su correcto funcionamiento aun cuando el usuario introduzca datos erróneos o manipule inadecuadamente las interfaces de usuario. **(10%)**

- *Usabilidad*: Capacidad del proyecto para ser entendido, aprendido, usado y al mismo tiempo, ser atractivo para el usuario, cuando se usa bajo condiciones específicas.

    - *Comprensibilidad*: El programa ofrece una interfaz de fácil comprensión, facilitando su aprendizaje y correcta utilización. El programa emite mensajes de alerta cuando se introducen valores erróneos. Existen elementos informativos que indican al usuario como operar el programa. **(5%)**

    - *Capacidad de ser atractivo*: El diseño de la interfaz de usuario, esto es: disposición de controles, esquema de colores, utilización de cajas de diálogo y demás elementos gráficos; es atractivo para el usuario. **(5%)**

- *Eficiencia*: Capacidad para proporcionar prestaciones apropiadas, relativas a la cantidad de recursos usados, bajo condiciones determinadas.

    - *Estructuras de datos*: Utiliza eficientemente las estructuras de datos para la solución del problema. **(30%)**