Ing. de Sistemas e Informática
Estructuras de Datos y Algoritmos
Jorge Mario Londoño P.

Ejercicios de Repaso

Tablas de dispersión

Tablas asociativas

1. En clase consideramos la implementación de la tabla de dispersión enlazada haciendo uso de un arreglo de Bags. En ese momento solo se consideraron las operaciones `get` y `put`.

   (a) Proponer una implementación "perezosa" de la operación `delete`. (Marca elementos de la lista como borrados, pero no hace el borrado físico). Se debe así mismo modificar la operación `put` para que pueda reutilizar los nodos "borrados" de las listas.

   (b) Proponer una implementación activa de la operación `delete`. Estimar su eficiencia en función del número de comparaciones realizadas.

2. Operaciones de conjuntos tales como la intersección y la diferencia normalmente requieren ciclos doblemente anidados que llevan a implementaciones cuadráticas.

   (a) Explicar como podrían implementarse estas operaciones en tiempo lineal haciendo uso de tablas asociativas.

   (b) Dar una implementación (pseudo-código) de la intersección y la diferencia haciendo uso de la idea propuesta en el numeral (a).

3. En clase se discutió la necesidad de hacer una "redispesión" de la tabla cuando el factor de carga se hace superior a la unidad. Proponer una implementación del algoritmo redispersion como un método de la clase SeparateChainingHashST que tome como argumento el nuevo valor M del tamaño del arreglo.

```
private void redispersion(int m)
```

Algoritmos de grafos

Grafos

1. Se tiene un grafo no dirigido g (instancia de `Graph`).
   a) Dar un algoritmo para determinar si el grafo es conexo o no.
   b) Dar un algoritmo para determinar cuántas componentes conexas tiene el grafo.
   c) Dar un algoritmo para determinar si el grafo contiene ciclos o no.
      a) En caso afirmativo indicar el ciclo encontrado.
      b) Mejor aún, obtener todos los ciclos que contenga el grafo.*
   d) Estimar la eficiencia de las soluciones propuestas.

2. Resolver los mismos problemas del punto anterior para un grafo dirigido (instancia de `Digraph`).

3. Hacer un procedimiento que convierta un `Graph` en un `Digraph` equivalente.

4. Hacer un procedimiento que convierta un `Graph` / `Digraph` a su representación por medio de una matriz de adyacencia.

5. En clase se considero el problema de encontrar la salida de un laberinto. Analizar la solución propuesta en caso que el grafo sea no conexo o que contenga ciclos. Se garantiza que encuentre la salida?

6. En el problema del laberinto se consideró la solución por medio de un recorrido DFS. Plantear la solución utilizando un recorrido BFS. Cuál de las dos es mejor en el peor caso?

7. Cuándo se analizó el ordenamiento topológico en clase solo se consideraron las dependencias entre tareas. El problema de la ruta critica debe tener en cuenta además la duración de las tareas (representadas como una arista con peso igual a la duración y con vértices que corresponden al inicio y fin de la tarea). Encontrar la ruta critica en el grafo de tareas representado de esta forma (Información adicional).

8. Dijkstra encuentra las rutas más cortas de un vértice a todos los demás

vértices. Serán las rutas opuestas las rutas más cortas desde todos los vértices al origen? Analizar para el caso del grafo no dirigido y del grafo dirigido.

9. Ilustrar con un contra-ejemplo que el árbol de caminos mínimos encontrado por Dijkstra no es igual al árbol de cubrimiento mínimo.

10. Se quiere enviar un mensaje “Broadcast” a todos los nodos de una red. Proponer un algoritmo que envie el mensaje a cada nodo una vez (ningún nodo debe recibir dos veces el mismo mensaje).