GRADO EN INTELIGENCIA ARTIFICIAL (MOSTOLES)

2361 - ESTRUCTURAS DE DATOS II - TARDE A - 20

> Evaluación > Parcial 3: Tablas hash y grafos

Comenzado el	miércoles, 8 de mayo de 2024, 16:10
Estado	Finalizado
Finalizado en	miércoles, 8 de mayo de 2024, 16:30
Tiempo	20 minutos
empleado	
Calificación	6.50 de 10.00 (65 %)

Pregunta **1**

Correcta

Se puntúa 1,00 sobre 1,00

[1 puntos] El factor de carga indica la relación entre el número de elementos de la tabla hash y el tamaño de esta. ¿Cuál de las siguientes respuestas es correcta?

- a. Para encadenamiento separado no tiene sentido.
- b. El valor ideal es 1 ya que indica que no se desperdicia espacio.
- o. Un valor superior a 1 indica eficiencia en el almacenamiento.

Respuesta correcta

La respuesta correcta es:

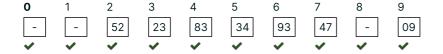
Para direccionamientos abierto es importante que sea menor que 0.5

Pregunta 2
Correcta
Se puntúa 1,00 sobre 1,00

[1 punto] Dada una tabla hash en direccionamiento abierto con prueba lineal, donde el hash es el último dígito, muestra el estado final de la tabla al insertar los siguientes elementos:

52, 23, 47, 83, 34, 93, 09

Tabla Hash



NOTA: La tabla debe estar completa, en las casillas que queden vacías poner "-

<u>"-</u>



Respuesta correcta

La respuesta correcta es:

[1 punto] Dada una tabla hash en direccionamiento abierto con prueba lineal, donde el hash es el último dígito, muestra el estado final de la tabla al insertar los siguientes elementos:

52, 23, 47, 83, 34, 93, 09

Tabla Hash

0 1 2 3 4 5 6 7 8 9 [-] [-] [52] [23] [83] [34] [93] [47] [-] [09]

NOTA: La tabla debe estar completa, en las casillas que queden vacías poner "-

".

```
Pregunta 3
Finalizado
Se puntúa 0,00 sobre 3,00
```

[3 puntos] Implementa en pseudocódigo el recorrido en profundidad.

```
def profundidad(arbol, nodoActual):
   si nodoActual.hijoIzq != NULL:
        nodoActual = nodoActual.hijoIzq
        profundidad(arbol, nodoActual)
    else
         si nodoActual.hijoDer != NULL:
              nodoActual = nodoActual.hijoDer
             profundidad(arbol, nodoActual)
              nodoActual = nodoActual.padre
              profundidad(arbol, nodoActual)
```

Comentario:

SOLUCIÓN:

```
Pila pila
Lista recorrido
Conjunto visitados

pila.apilar(nodo_inicial)

Mientras No esVacia(pila)
   aux = cima(pila)
   Si No visitados.contiene(aux)
   visitados.insertar(aux)
   recorrido.insertar(aux)
   Para n in aux.adyacentes
        pila.apilar(n)

devolver recorrido
```

```
Pregunta 4
Finalizado
Se puntúa 3,50 sobre 4,00
```

[4 puntos] Implementa en pseudocódigo una función que dado un grafo dirigido devuelva los nodos sumidero.

```
def nodosSumidero(grafo):
   sumidero = []
   para nodos en grafo:
       si nodos.listaAdy == NULL:
           nodos.añadir(sumidero)
return sumidero
```

Comentario:

La lista podría no ser nula, simplemente estar vacía.

SOLUCIÓN:

```
for nodo: grafo
   Si adyacencias(nodo).size() == 0
    sumideros.insertar(nodo)

devolver sumideros
```

Pregunta 5	
Correcta	
Se puntúa 1,00 sobre 1,00	

[1 punto] Empareja los distintos contenedores con su descripción.

Guardo elementos de consulta eficiente pero no sé cual es el primero.

Guardo pares clave valor donde una clave permite varios valores y me importa el orden.

Guardo pares clave valor sin importar el orden y sin repetidos.

Guardo elementos con consulta eficiente pero es importante el tamaño.

Respuesta correcta

La respuesta correcta es:

Guardo elementos de consulta eficiente pero no sé cual es el primero. \rightarrow std::unordered_set,

Guardo pares clave valor donde una clave permite varios valores y me importa el orden. → std::multimap,

Guardo pares clave valor sin importar el orden y sin repetidos. \rightarrow std::unordered_map,

Guardo elementos con consulta eficiente pero es importante el tamaño. \rightarrow std::set

Actividad previa

Parcial 2: Árboles

Ir a...

Siguiente actividad

InnovacionIA EDII-Preguntas