UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS

Ayudantía 6: Repaso prueba 2

Profesores: Sebastián Sáez, Diego Ramos

Ayudantes: Diego Duhalde, Benjamín Wiedmaier, Fernando Zamora

Ordenamiento

1. Defina qué es un algoritmo de ordenamiento, mencione al menos 3 algoritmos de ordenamiento y explique brevemente cómo funciona cada uno. Además, mencione la complejidad en tiempo de cada uno.

2. Bajos a trabajar en base al arreglo `A = [5, 2, 9, 1, 5, 6]`, aplique 3 iteraciones de Selection Sort, 3 iteraciones de Insertion Sort y 3 iteraciones de Bubble Sort. Para cada iteración, indique el arreglo resultante, en el caso de Insertion Sort asuma que la primera iteración parte de i=1.

Búsqueda

1. Describa búsqueda lineal y búsqueda binaria, también mencione la complejidad en el peor caso, mejor caso y caso promedio.

2. Tenemos el arreglo `A = [1, 2, 3, 4, 5, 6, 7, 8, 9]`, aplique búsqueda binaria para encontrar el número 3. Describa cada paso de la búsqueda y el resultado final, mencione si estamos más cerca del peor caso, mejor caso o del caso promedio en términos de complejidad.

Listas y Hashing

1. Defina qué es una lista enlazada, mencione sus ventajas y desventajas en comparación con un arreglo. Además, mencione la complejidad en tiempo de las operaciones básicas (`insertar`, `eliminar`, `buscar`).

2. Tenemos la siguiente lista enlazada: `A = [1, 2, 3, 4, 5]`, aplique las siguientes operaciones: insertar el número 6 al final de la lista, eliminar el número 2 y buscar el número 4. Describa cada paso de la operación y el resultado final.

3. Defina qué es una tabla hash, mencione sus ventajas y desventajas en comparación con una lista enlazada. Además, mencione la complejidad en tiempo de las operaciones básicas (`insertar`, `eliminar`, `buscar`).

4. Suponga que tenemos una tabla hash de tamaño 10 y la siguiente función hash: `h(x) = x mod 10`. Inserte los siguientes números en la tabla hash: 12, 22, 32, 42, 52. Describa cada paso de la operación y el resultado final. ¿Qué tipo de colisión ocurre? ¿Cómo se podría resolver?

Colas y Pilas

1. Menciona las diferencias principales entre una cola y una pila, da un ejemplo cotidiano de cada una.

2. Explique en cada caso cómo se puede insertar y eliminar elementos de una cola y una pila. ¿Qué complejidad tienen estas operaciones? ¿Cual opciones nos conviene elegir?

RESPUESTAS

Ordenamiento

1. Un algoritmo de ordenamiento es un procedimiento o conjunto de instrucciones que permite organizar los elementos de una estructura de datos en un orden específico, ya sea ascendente o descendente. Ejemplos de algoritmos de ordenamiento incluyen:

   - **Selection Sort:** Encuentra el elemento más pequeño (o más grande) en cada iteración y lo coloca en su posición correcta. Complejidad en tiempo: $O(n^2)$.
   - **Insertion Sort:** Construye el arreglo ordenado de forma incremental, insertando cada elemento en su posición correcta. Complejidad en tiempo: $O(n^2)$ en el peor caso, $O(n)$ en el mejor caso.
   - **Merge Sort:** Divide el arreglo en mitades, las ordena recursivamente y luego las combina. Complejidad en tiempo: $O(n \log n)$.

2. A continuación, se muestran las iteraciones de los algoritmos de ordenamiento solicitados:

   - **Selection Sort:**
     (a) Iteración 1: Encuentra el menor elemento (1) y lo intercambia con el primer elemento. Resultado: `[1, 2, 9, 5, 5, 6]`.
     (b) Iteración 2: Encuentra el siguiente menor elemento (2) y lo intercambia con el segundo elemento. Resultado: `[1, 2, 9, 5, 5, 6]`.
     (c) Iteración 3: Encuentra el siguiente menor elemento (5) y lo intercambia con el tercer elemento. Resultado: `[1, 2, 5, 9, 5, 6]`.
   - **Insertion Sort:**
     (a) Iteración 1 ($i = 1$): Inserta el segundo elemento (2) en su posición correcta. Resultado: `[2, 5, 9, 1, 5, 6]`.
     (b) Iteración 2 ($i = 2$): Inserta el tercer elemento (9) en su posición correcta. Resultado: `[2, 5, 9, 1, 5, 6]`.
     (c) Iteración 3 ($i = 3$): Inserta el cuarto elemento (1) en su posición correcta. Resultado: `[1, 2, 5, 9, 5, 6]`.
   - **Bubble Sort:**
     (a) Iteración 1: Compara e intercambia elementos adyacentes si están en el orden incorrecto. Resultado: `[2, 5, 1, 5, 6, 9]`.
     (b) Iteración 2: Repite el proceso para los primeros $n - 1$ elementos. Resultado: `[2, 1, 5, 5, 6, 9]`.
     (c) Iteración 3: Repite el proceso para los primeros $n - 2$ elementos. Resultado: `[1, 2, 5, 5, 6, 9]`.

Búsqueda

1. La búsqueda lineal consiste en recorrer el arreglo elemento por elemento hasta encontrar el valor deseado o llegar al final del arreglo. Complejidad:

- Peor caso: $O(n)$ (el elemento no está o está al final).
- Mejor caso: $O(1)$ (el elemento está al inicio).
- Caso promedio: $O(n/2) \approx O(n)$.

La búsqueda binaria requiere que el arreglo esté ordenado. Divide el arreglo en mitades y compara el elemento central con el valor buscado, descartando la mitad donde no puede estar el valor. Complejidad:

- Peor caso: $O(\log n)$ (el elemento está al final del proceso).
- Mejor caso: $O(1)$ (el elemento está en el centro inicial).
- Caso promedio: $O(\log n)$.

2. Para encontrar el número 3 en el arreglo `A = [1, 2, 3, 4, 5, 6, 7, 8, 9]` usando búsqueda binaria:

   (a) Paso 1: El rango inicial es $[1, 9]$. El elemento central es $A[5] = 5$. Como $3 < 5$, descartamos la mitad derecha.

   (b) Paso 2: El rango ahora es $[1, 4]$. El elemento central es $A[2] = 2$. Como $3 > 2$, descartamos la mitad izquierda.

   (c) Paso 3: El rango ahora es $[3, 3]$. El elemento central es $A[3] = 3$. Encontramos el número buscado.

   El caso se encuentra más cerca del mejor caso, ya que el número fue encontrado en pocos pasos.

Listas y Hashing

1. Una lista enlazada es una estructura de datos compuesta por nodos, donde cada nodo contiene un valor y un puntero al siguiente nodo en la lista. Ventajas y desventajas:

   - Ventajas:
     - Inserciones y eliminaciones son más eficientes que en un arreglo, especialmente en posiciones intermedias.
     - No requiere un tamaño fijo, puede crecer dinámicamente.
   - Desventajas:
     - Acceso secuencial: no se puede acceder directamente a un elemento por su índice.
     - Mayor uso de memoria debido a los punteros.

   Complejidad en tiempo:

   - `insertar`: $O(1)$ si se realiza al inicio o al final, $O(n)$ en el peor caso.
   - `eliminar`: $O(1)$ si se tiene acceso directo al nodo, $O(n)$ en el peor caso.
   - `buscar`: $O(n)$.

2. Operaciones en la lista enlazada `A = [1, 2, 3, 4, 5]`:

(a) Insertar el número 6 al final:

- Crear un nuevo nodo con valor 6.
- Ajustar el puntero del último nodo (5) para que apunte al nuevo nodo.
- Resultado: `[1, 2, 3, 4, 5, 6]`.

(b) Eliminar el número 2:

- Encontrar el nodo con valor 2.
- Ajustar el puntero del nodo anterior (1) para que apunte al nodo siguiente (3).
- Resultado: `[1, 3, 4, 5, 6]`.

(c) Buscar el número 4:

- Recorrer la lista nodo por nodo hasta encontrar el valor 4.
- Resultado: Nodo con valor 4 encontrado.

3. Una tabla hash es una estructura de datos que utiliza una función hash para mapear claves a índices en un arreglo. Ventajas y desventajas:

   - Ventajas:
     - Operaciones de búsqueda, inserción y eliminación son muy rápidas en promedio ($O(1)$).
     - Ideal para grandes volúmenes de datos.
   - Desventajas:
     - Puede haber colisiones, lo que requiere técnicas de resolución.
     - No mantiene el orden de los elementos.

   Complejidad en tiempo:

   - `insertar`: $O(1)$ en promedio, $O(n)$ en el peor caso.
   - `eliminar`: $O(1)$ en promedio, $O(n)$ en el peor caso.
   - `buscar`: $O(1)$ en promedio, $O(n)$ en el peor caso.

4. Inserción en la tabla hash de tamaño 10 con función `h(x) = x mod 10`:

   - Insertar 12: $h(12) = 2$. Colocar 12 en la posición 2.
   - Insertar 22: $h(22) = 2$. Colisión. Resolver con encadenamiento o direccionamiento abierto.
   - Insertar 32: $h(32) = 2$. Colisión. Resolver con la misma técnica.
   - Insertar 42: $h(42) = 2$. Colisión. Resolver con la misma técnica.
   - Insertar 52: $h(52) = 2$. Colisión. Resolver con la misma técnica.

   Tipo de colisión: Todas las claves tienen el mismo índice. Resolución: Usar encadenamiento (listas enlazadas) o direccionamiento abierto (sondeo lineal, cuadrático, etc.).

Colas y Pilas

1. Las principales diferencias entre una cola y una pila son:

   - **Cola:** Sigue el principio FIFO (First In, First Out), es decir, el primer elemento en entrar es el primero en salir. Ejemplo cotidiano: una fila en un banco.
   - **Pila:** Sigue el principio LIFO (Last In, First Out), es decir, el último elemento en entrar es el primero en salir. Ejemplo cotidiano: una pila de platos.

2. Para insertar y eliminar elementos:

   - **Cola:**
     - Inserción: Se realiza al final de la cola. Complejidad: $O(1)$.
     - Eliminación: Se realiza al inicio de la cola. Complejidad: $O(1)$.
   - **Pila:**
     - Inserción: Se realiza en la parte superior de la pila. Complejidad: $O(1)$.
     - Eliminación: Se realiza en la parte superior de la pila. Complejidad: $O(1)$.

   La elección depende del caso de uso. Si necesitamos procesar elementos en el orden en que llegan, usamos una cola. Si necesitamos procesar el elemento más reciente primero, usamos una pila.