INFO145 : Diseño y Análisis de Algoritmos
Tarea: Técnicas de Representación y Compresión de Arreglos Ordenados

Profesor: Héctor Ferrada
Auxiliares: Carolina Obreque & Benjamín Parra

Mayo 29, 2024

Abstract

En esta tarea se deben comprender y aplicar diferentes técnicas de representación y compresión de arreglos ordenados. Los objetivos incluyen la medición del espacio requerido por cada técnica y la estimación del tiempo de búsqueda binaria. Las técnicas a utilizar serán la representación explícita, Gap-Coding ($X[i] = A[i] - A[i-1]$), y gaps comprimidos con Shannon-Fano o Huffman.

1 Instrucciones

Se debe realizar una busqueda binaria sobre un arreglo X creciente muy grande representado de las siguientes maneras:

- Arreglo explícito sin comprimir.
- Arreglo representado con Gap-Coding.
- Arreglo representado por Gaps comprimidos con Shannon-Fano o Huffman.

Para todas estas representaciónes deberá indicar, ya sea utilizando notación asintótica o exacta, la cantidad de espacio adicional y coste computacional de la busqueda. Luego, se debe implementar cada solución y medir empíricamente el espacio que estas utilizan en la RAM; acto seguido, ejecute experimentos para medir el tiempo de consulta en cada estructura de datos utilizando gran cantidad de datos de entrada.
Las pruebas deben realizarse con una distribución inicial tanto lineal como normal para la generación aleatoria.

Informe: El informe debe ser claro, objetivo y detallado, debe contener: Resumen, Introducción , metodología, experimentación y conclusiones.

1. **La introducción** debe contener el contexto del problema, y presentar las partes del informe.
2. **La metodología** debe detallar la inicialización de los datos, la hipótesis inicial que será respondida en la conclusión, así como los costos, pseudocódigos, representaciónes y el método utilizado para medir el rendimiento. Los n de experimentación deben ser especificados considerando numeros grandes, por ejemplo, en potencias de 2 o 10.
3. **La experimentación** debe ser acompañada de gráficos o tablas que ayuden a entender los resultados.
4. **La conclusión** debe responder a la hipótesis inicial planteada y resumir los aspectos más relevantes de los resultados obtenidos.

Es importante realizar multiples pruebas antes de sacar conclusiones.
Adicionalmente, se debe incluir una sección donde se indiquen las fuentes de donde se recopiló la información utilizada en el informe.
Repositorio: Ademas del informe, debe entregar un link de github con su proyecto, donde se debe

encontrar el código fuente que debe ser consistente con su pseudocodigo.
Se puedem colocar instrucciones adicionales de compilación y explicar su código en un archivo readme.md en su repositorio.
Durante el desarrollo, no dude en indicar posibles optimizaciones o incluir casos extra que apoyen sus hipotesis.
Su codigo debe ser modular y debe estar ordenado y comentado, o se descontará puntaje.
El uso de modelos de lenguaje para la redacción del informe también es motivo de descuento.
Entrega: Debe subir el informe con el link del repositorio a siveduc con el nombre T1apellidos.pdf (Maximo 4 integrantes por grupo). Fecha de entrega: **Miércoles 19 de Junio.**
Consultas: Cualquier duda acerca de este trabajo será respondida por los ayudantes del ramo.

2 Descripción de los casos

2.1 Arreglo explícito

En el primer caso, se debe tener dos arreglos ordenados de menor a mayor, los cuales deben ser inicializados con valores aleatorios ordenados, siguiendo una distribución:

- Lineal (Puede ser definiendo A[0] = rand(), A[i] = A[i-1] + rand() % ϵ)
- Normal (Puede utilizar librerias si asi lo requiere) con distintas propuestas de desviaciones estandar.

Una vez inicializadas las estructuras, se deben realizar busquedas binarias y medir su tiempo tanto teoricamente como experimentalmente para una gran cantidad de datos.

2.2 Arreglo representado con Gap-Coding

En este caso, se debe representar ambos arreglos utilizando Gap-Coding. La idea es que los arreglos originales no seam accesibles directamente, en su lugar, se utilizara una estructura adicional llamada "sample" para ayudar a la busqueda.
Para representar un arreglo utilizando Gap-Coding se debe calcular la diferencia entre cada elemento consecutivo del arreglo **ordenado**. El primer elemento del arreglo resultante será el mismo que el primer elemento del arreglo original.
Ejemplo, para un arreglo $X = [2, 7, 10, 12, 12, 16]$ los gaps serian $GC(X) = [2, 5, 3, 2, 0, 4]$
La estructura "sample" se utiliza para guardar algunos valores especificos del arreglo original, esta debe tener un espacio m con $m < n$, y debe proponer cuales y cuantos valores guardar. Además, se puede especificar un espacio b entre los elementos del sample. La estructura adicional se utiliza para reconstruir al arreglo original sin acceder a este, por ejemplo, para el arreglo X seleccionamos un $m = 3$ y un $b = 2$ con $b = \Delta i = \frac{n}{m}$, el sample seria $[2, 10, 12]$. Tambien puede proponer b como variable, por ejemplo, $m = \log_3 n$ con $b = 3^j$.
Con esta estructura debe acortar el intervalo de busqueda usando busqueda binaria y sobre el arreglo gap-coded debe encontrar la posición del elemento final.

2.3 Representación con Shannon-Fano o Huffman

En el ultimo caso, se deben comprimir los arreglos que estaban representados con Gap-Coding, utilizando representaciónes con codigos de Huffman o Shannon-Fano.
En caso de usar el algoritmo de Huffman, describa la estructura utilizada para obtener los codigos en el informe.
La idea es utilizar una nueva estructura de arreglo para cada entrada, cuya representación en bits por elemento sea menor que la inicial, es decir, que si antes los elementos eran long de 32 bits, se comprima y se calcule un nuevo largo maximo (por ejemplo, usando shorts de 8 bits para representar la codificacion de cada), para luego que cada elemento del nuevo arreglo ocupe solo esa cantidad de bits Para la aproximación, detalle los pasos de como obtuvo los valores en el informe
Tenga en cuenta que los largos son de tamaño fijo.
Proponga tambien, una manera de tratar los outliers para que sus diferencias con los valores no causen que se empeore mucho la compresión
Si necesita, puede aproximar los bits por elemento usando los tipos de datos numericos del lenguaje,

ejemplo:

Largo de algunos tipos de **C/C++** en un terminal arbitrario:

- $\|$char$\|$: 4 bits
- $\|$short$\|$: 8 bits
- $\|$int$\|$: 16 bits
- $\|$long$\|$: 32 bits

Usando la función **sizeof()** se puede saber el largo en **bytes** de un tipo en su maquina.

Por otro lado, si quiere experimentar con la representación de bits usando librerias o quiere hacer representaciones de largo variable ($\|\mu_i\| = \log_2(\frac{1}{p(\mu_i)})$), las propuestas serán bienvenidas.
Aún se cuenta con la estructura sample para cerrar el intervalo de busqueda, pero la busqueda para el elemento final debe realizarlo sobre su arreglo comprimido, con las representaciones a nivel de bit.

3 Lenguajes de programación

En caso de no usar C/C++, asegurese de reconocer el funcionamiento del recolector de basura y de saber como operar a nivel de bit para realizar la codificación del punto **2.3**. Estos detalles deben ser explicados junto con sus efectos en las pruebas empíricas.