Tarea 1
INFO088 - Taller de Estructuras de Datos y Algoritmos

Instituto de Informática, Universidad Austral de Chile.

Agosto 2025

Resumen

El objetivo de este trabajo es profundizar en el uso de estructuras de datos y la STL mediante la simulación de un sistema simplificado de resolución de nombres, similar al funcionamiento del *Domain Name System* (DNS). Para ello se utilizarán dos vectores (uno con nombres de dominio y otro con direcciones IP asociadas), sobre los cuales se implementarán operaciones fundamentales de búsqueda y ordenamiento. Se estudiará el rendimiento de algoritmos de búsqueda secuencial y búsqueda binaria, junto con técnicas de ordenamiento, analizando sus costos asintóticos y empíricos. La actividad busca destacar la importancia de elegir la estructura y el algoritmo adecuado al resolver problemas inspirados en sistemas reales de la informática.

Descripción del problema

Se requiere simular un sistema básico de resolución de nombres de dominio (**DNS**) utilizando dos vectores:

- Un vector **Nombres**, que contiene cadenas de texto (`strings`) representando los dominios (ej: `www.uach.cl`).
- Un vector **IPs**, que contiene las direcciones IP asociadas a cada dominio (ej: `146.83.181.2`).

El índice de cada vector representa un identificador único (**ID**), de manera que $Nombres[i]$ corresponde a $IPs[i]$.

Los requisitos específicos son:

- Construir los dos vectores paralelos a partir de un conjunto de datos.

- Implementar un algoritmo de **búsqueda secuencial** sobre el vector de nombres, que retorne la ip correspondiente al dominio buscado.
- Implementar un algoritmo de **ordenamiento lexicográfico** en base a std::sort sobre el vector de nombres, reordenando el vector de IPs de forma coherente.
- Implementar **búsqueda binaria** sobre el vector de nombres ya ordenado.
- Realizar comparaciones de tiempos entre la estructura desordenada y ordenada.

Descripción de la solución

La solución se divide en tres etapas principales:

1. **Construcción de la estructura:**
   - Generar un conjunto de n nombres de dominio y sus respectivas direcciones IP.
   - Almacenar los datos en dos vectores paralelos: `vector<string> Nombres` y `vector<string> IPs`.
2. **Búsqueda y ordenamiento:**
   a) **Búsqueda secuencial:** recorrer el vector `Nombres` elemento por elemento hasta encontrar el dominio solicitado, retornando el ip asociado.
   b) **Ordenamiento:** ordenar el vector `Nombres` en orden lexicográfico (por ejemplo, con `std::sort`) y reorganizar el vector `IPs` para mantener la correspondencia.
   c) **Búsqueda binaria:** implementar búsqueda binaria sobre el vector de nombres ordenado, retornando el ip asociado.
3. **Comparación de rendimiento:**
   - Medir tiempos de búsqueda secuencial y búsqueda binaria para un mismo conjunto de dominios.
   - Analizar el comportamiento al aumentar el tamaño de los vectores.

Experimentación

Se deberán realizar pruebas con distintos tamaños de vectores:

- Conjuntos pequeños ($n = 200,000$ pares dominio–IP).
- Conjuntos medianos ($n = 1,000,000$ pares).
- Conjuntos grandes ($n = 10,000,000$ pares).

En caso de que no sea posible realizar la experimentación con los valores propuestos, el estudiante podrá modificarlos. No obstante, será obligatorio especificar en el documento cuáles fueron los nuevos valores utilizados.

Para cada caso:

1. Medir el tiempo de construcción de los vectores.
2. Realizar $REP = 100\,000$ búsquedas aleatorias secuenciales, registrando el tiempo promedio.
3. Ordenar los vectores con std::sort y medir el tiempo empleado.
4. Realizar $REP = 100\,000$ búsquedas binarias, registrando tiempos promedio.
5. Generar:
   - **Gráfico de linea**, cantidad de dominios (eje x) vs tiempo de contrucción (eje y).
   - **Gráfico de dos lineas**:
     - Cantidad de dominios vs tiempo promedio de búsqueda secuencial
     - Cantidad de dominios vs tiempo promedio de búsqueda binaria.

Requisitos de Entrega

Documento. [60 %] (Máximo 14 páginas, todo incluido) El informe técnico deberá contener:

- **Abstract** Resumiendo en breves párrafos el trabajo presentado en el documento.

- **Introducción** destacando el contexto del DNS, funcionamiento de std::vector y std::sort, iteradores, breve explicación de la solución implementada.
- **Metodología** con descripción de la construcción, búsqueda secuencial, ordenamiento y búsqueda binaria, se incluyen pseudocódigos en el formato especificado y declaración de structs o clases creadas junto con una breve explicación.
- **Resultados experimentales** con gráficos comparativos junto con una breve descripción.
- **Análisis asintótico:** Analisis de las operaciones implementadas, entre ellas, ordenamiento léxicografico, búsqueda secuencial y binaria (utilizar guía de notación asintótica).
- **Conclusiones**: Realizar una comparación entre el análisis teórico (análisis asintótico) y los resultados obtenidos experimentalmente. Se espera contrastar el comportamiento de la estructura de datos previamente utilizada con la nueva implementación, explicando las posibles diferencias observadas. Enfatizar el papel que desempeñan las estructuras de datos en el rendimiento del algoritmo y cómo la forma en que se organiza la información influye en los resultados.

Código Fuente. [40 %] Basándose en el repositorio guía, la implementación en C++ deberá incluir:

- Archivo `main.cpp` que reciba argumentos (cantidad de dominios, cantidad de búsquedas), realiza llamadas a las funciones creadas en dns.cpp y experimentacion.cpp.
- `dns.cpp`, donde se implementen funciones de búsqueda secuencial, búsqueda binaria y ordenamiento.
- `experimentacion.cpp`, con las pruebas de rendimiento.
- Estructura modular (usar repositorio guía)
- Código organizado y bien documentado.
- Buen uso de la STL.

Entrega y revisión

- **Fecha de entrega:** Viernes 24 de Octubre 2025
- **Revisión:** Mediante reuniones presenciales la semana del 27 de octubre.
- **Inscripción:** Selección de horario en hoja de cálculo. compartida mediante correo electronico el día Viernes 24 de Octubre.

Formato

A continuación el pseudocódigo de una solución, este es el formato de pseudocódigo que se espera para las rutinas de los algoritmos de búsqueda, inserción y eliminación que se piden para cada estructura:

Input: un arreglo de n enteros $A[0 \ldots n-1]$
Output: u, el elemento más repetido en A

```
masRepetido(A, n) {
    u = A[0]
    occ = 0
    for i = 0 to n-1 do {
        count = 0
        x = A[i]
        for j = 0 to n-1 do
            if (A[j] == x) then
                count = count+1
        if (count > occ) then {
            occ = count
            u = x
        }
    }
    return u
}
```