INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS PUEBLA

Análisis y diseño de algoritmos avanzados (Gpo 602)

TC2038.602

Manual de usuario E1. Actividad Integradora 1

Profesor:

Juan Manuel Ahuactzin Larios

Roberto Castro Gómez | A01425602

Santiago Rodríguez Gutiérrez | A01738097

Luis Antonio Salinas Gonzalez | A01735375

19 de Octubre de 2025

Descripción

Este proyecto incluye dos algoritmos de construcción de suffix arrays:

1. SA-IS (Suffix Array Induced Sorting) – eficiente para textos grandes.
2. Manber-Myers – clásico, útil para textos medianos y búsqueda de patrones.

Ambos permiten medir tiempo de ejecución y consumo de memoria (SA-IS tiene medición real, Manber-Myers tiene estimación).

Requisitos

- Sistema operativo Linux (para medición de memoria en SA-IS).
- Compilador C++11 o superior.
- Carpeta de proyecto organizada así:

```Shell
algoritmos-busqueda/
 ├─ libros/
 │   ├─ dracula.txt
 │   ├─ homer.txt
 │   └─ ... otros libros .txt
 ├─ sais/
 │   └─ sais.cpp
 └─ manber_myers/
     └─ manbermyers.cpp
```

Algoritmo SA-IS

Uso

- Modificar `nombreArchivo` en `main()` para cambiar el archivo:

```
C/C++
string nombreArchivo = "dracula.txt";
```

- Ejecutar:

```
Shell
g++ -std=c++11 sais.cpp -o sais
./sais
```

Funciones principales

- `leerArchivo(string &nombre)`: devuelve el contenido del archivo como `string`.
- `getBuckets(vector<int>& T)`: genera rangos (buckets) por símbolo.
- `sais(vector<int>& T)`: construye el Suffix Array usando SA-IS.
- `getMemoryUsage()`: devuelve memoria en KB leída de `/proc/self/status`.

Salida

- Suffix Array.
- Nombre del archivo.
- Tamaño del texto.
- Tiempo transcurrido (ms).
- Memoria utilizada (MB).

Algoritmo Manber-Myers

Uso

- Modificar `nombre` en `main()` para cambiar el archivo:

```C/C++
string nombre = "../libros/homer.txt";
string patron = "of";
```

- Ejecutar:

```Shell
g++ -std=c++11 manbermyers.cpp -o manbermyers
./manbermyers
```

Funciones principales

1. `readFile(string &nombre)`: abre un archivo y devuelve el contenido como `string`.
2. `SubstrRank`: estructura que almacena los rankings izquierdo y derecho y el índice del sufijo.
3. `makeRanks()`: asigna un ranking único a cada subcadena.
4. `suffixArray(const string &T)`: construye el suffix array mediante el algoritmo Manber-Myers.
5. `searchPattern(const string &T, const string &pattern, const vector<int> &SA)`: busca todas las ocurrencias de un patrón en el texto usando binary search sobre el suffix array.

Salida

- Estimación de memoria usada.
- Tiempo de construcción del suffix array (ms).

- Número de ocurrencias encontradas del patrón.
- Primeras 10 posiciones (si existen).

Comparativa

Algoritmo	Uso recomendado	Tamaño de texto	Característica adicional
SA-IS	Alta eficiencia	Muy grande	Memoria real medida
Manber-Myers	Pruebas o educativo	Pequeño o mediano	Búsqueda de patrones

- Para archivos grandes, SA-IS es más rápido con un mayor uso de memoria.
- Para pruebas rápidas o búsqueda de patrones, Manber-Myers es suficiente.

Notas

- Ambas implementaciones soportan caracteres ASCII.
- Para cambiar archivos, modificar la variable correspondiente en `main()`.
- SA-IS mide memoria real, Manber-Myers ofrece estimación aproximada.
- Ambos algoritmos pueden integrarse para comparar eficiencia y consumo de memoria.