PRÁCTICA 2 BÚSQUEDA CON RETROCESO

AUTORES: Manel Jordá Puig NIP: 839304
José Miguel Angós NIP: 852646

En esta práctica aplicaremos de manera práctica los fundamentos de los algoritmos de búsqueda con retroceso. Haremos uso de dicho algoritmo para implementar un optimizador del área ocupada por diferentes artículos en una página.

Se dispone de 3 tareas a realizar:

En la Tarea 1, se debe diseñar un algoritmo de búsqueda con retroceso que, dados: la dimensión de la página (anchura W y altura L); y una lista de n artículos, cada uno con su dimensión y posición, determine los artículos a colocar en la página y calcule la cantidad de espacio total ocupada por los mismos.

En la Tarea 2, se requiere el desarrollo de un programa llamado "busca" que haga uso del algoritmo a desarrollar en la Tarea 1. Este programa debe tomar un archivo de entrada que contiene los datos de los artículos y la página, procesarlas y generar un archivo de salida que contenga los resultados correspondientes, incluyendo el área total ocupada por los artículos elegidos y el tiempo de ejecución.

En la Tarea 3, se pide analizar la eficiencia (tiempo de ejecución y número de nodos del árbol de búsqueda generados) del algoritmo implementado a través de un conjunto de pruebas.

Este programa se ejecuta de la siguiente forma:

busca pruebas.txt resultados.txt

<busca> es el programa ejecutable

<pruebas.txt> es un fichero de texto que incluye los datos de artículos y la página.

<resultados.txt> es un fichero de texto que guarda los resultados correspondientes.

IMPLEMENTACIÓN

Lo primero que se ha realizado es implementar las funciones para abrir y realizar la lectura de los ficheros que contienen los datos de los artículos y de la página. Por otro lado, también se ha realizado una función para escribir los resultados en un nuevo fichero. Estas son **abrir_fichero_lectura** y **abrir_fichero_escritura**. Estas funciones utilizan flujos de entrada y salida para leer y escribir datos desde y hacia archivos respectivamente.

La exploración que realizamos de las posibles combinaciones de artículos se realiza principalmente en la función **construir_siguiente_nivel**. Esta función utiliza un enfoque recursivo para explorar el árbol de decisiones que se genera al realizar todas las combinaciones posibles de artículos. Se representa el árbol como una estructura de nodos en la que cada nodo contiene una configuración de artículos en la página. La exploración avanza desde la raíz del árbol hasta las hojas, evaluando cómo se disponen los artículos y actualizando la solución óptima encontrada hasta el momento.

Para representar las configuraciones de artículos en la página, se utilizan vectores de la estructura **Articulo**, que almacenan información sobre el ancho, alto y posición de cada artículo. Estos vectores se pasan como parámetros entre las funciones durante la búsqueda.

La función **calcular_area** se encarga del cálculo del área ocupada por los artículos en la página. Divide el área de la página en partes más pequeñas y calcula el área de intersección entre los artículos en cada parte.

La estrategia de poda se implementa en la función **aplicar_poda**, que compara el área óptima actual con el área máxima restante. Si la suma del área actual y el área restante es menor o igual que el área óptima, se aplica la poda y se descarta la rama del árbol actual.

Por último, en el programa principal, se realiza finalmente la lectura de datos desde un archivo de entrada, la ejecución del algoritmo de búsqueda con retroceso y la escritura de los resultados en un archivo de salida, utilizando las funciones de **leer_pagina** (para obtener adecuadamente los datos y guardarlas en sus respectivas variables) y **escribir_resultados.**

EXPERIMENTACIÓN

Para probar el correcto funcionamiento se han creado varios ficheros de prueba que contienen distintos ejemplos de páginas con sus respectivos artículos, variando tanto el número de estos, como sus tamaños y el tamaño de la página.

A continuación mostramos los tests a los que hemos sometido nuestro programa. En el anexo también mostramos unas imágenes informativas correspondientes a cada test realizado.

Test 1

```
5 20 20
3 3 3 3
3 3 1 1
3 3 5 1
3 3 1 5
3 3 5 5
```

Test 2

```
4 12 7
1 1 1 1
2 3 1 3
8 4 1 3
9 5 3 1
```

Test 3

```
4 12 7
1 12 1 0
12 3 0 1
12 1 0 5
6 3 5 1
```

Test 4

```
4 12 7
3 7 0 0
6 7 3 0
3 7 9 0
12 7 0 0
```

Resultados para Test 1
Área: 36 | Tiempo de ejecución: 736 ms

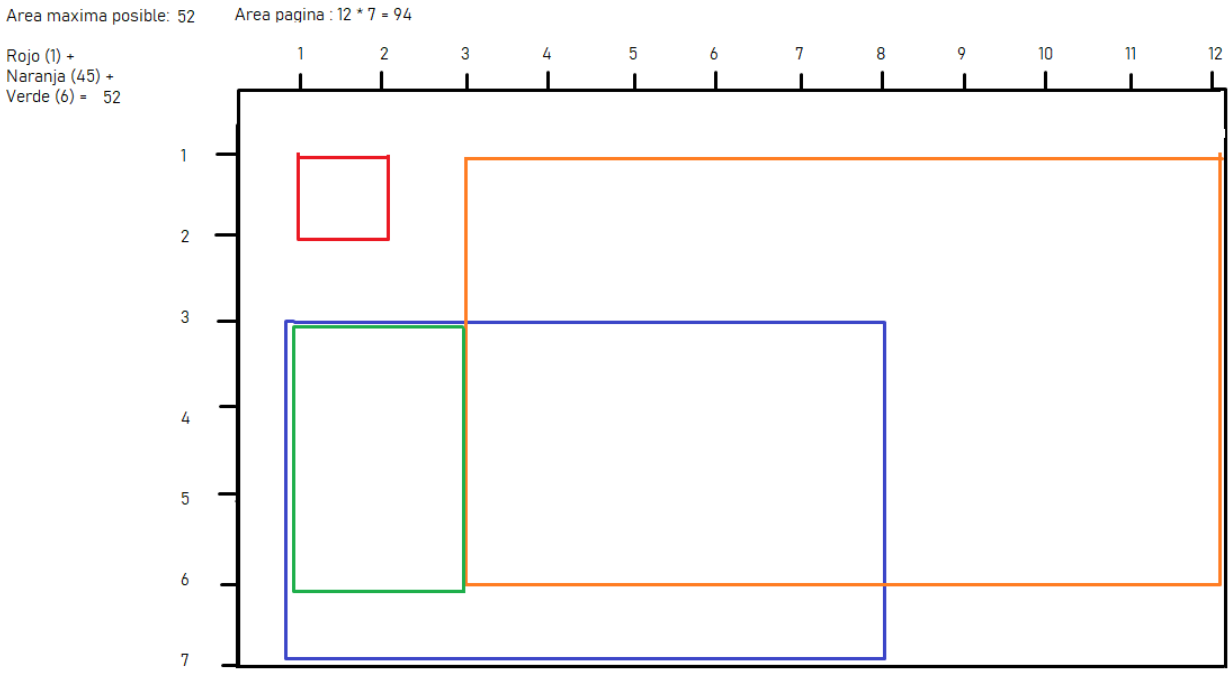
Resultados para Test 2
Área: 52 | Tiempo de ejecución: 739 ms

Resultados para Test 3
Área: 48 | Tiempo de ejecución: 601 ms

Resultados para Test 4
Área: 84 | Tiempo de ejecución: 392 ms

ANEXO

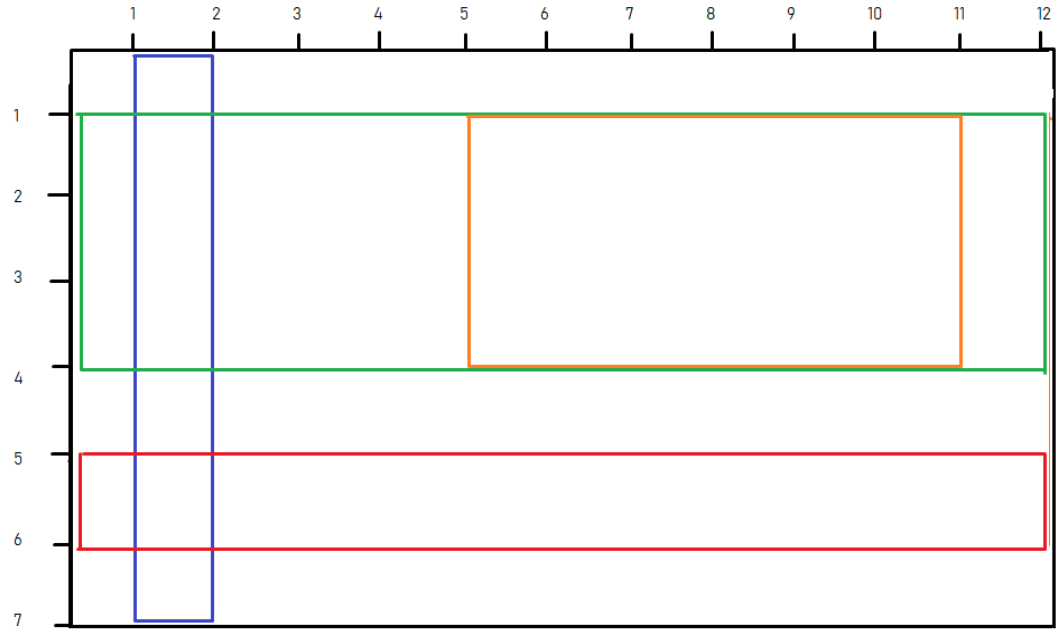
Test 2:



Test 3:

Area óptima máxima: 48

Verde(36) +
Rojo (12) = 48



Test 4:

Area óptima máxima: 7 * 12 = 84

2 opciones:

-Rojo (12*7)

- Naranja(21) +
  Verde(42) +
  Azul(21)