Estado de Avance

2D Strip Packing Problem

06 de noviembre de 2023

Florencia Ramírez Sancristoful

Representación

01 Rectángulos

Se representan con un ***struct***.

02 Región

Se representa con una ***clase***.

```
struct Position {
    int x;
    int y;
    int r;
};

struct Rectangle {
    int id;
    int width;
    int height;
    Position position;
};

class Strip {
private:
    int fixedWidth;
    vector<Rectangle> rectangles;

public:
    explicit Strip(int w) : fixedWidth(w) {}
    ~Strip() = default;
    int getHeight();
    int evaluationFunction();
    void addRectangleGreedy(Rectangle rectangle);
    bool canBePlaced(Rectangle rectangle, int posX, int posY, int orientation);
    void printOutput();
};
```

Lectura de instancias

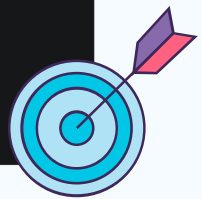
```cpp
int main(int argc, char* argv[]) {
    string instance = argv[1];
    ifstream input(instance);

    if (!input.is_open()) {
        cerr << "Ocurrió un error intentando leer la instancia." << endl;
        return 1;
    }

    int n, w;
    input >> n >> w;

    Strip strip(w);
    for (int i = 0; i < n; i++) {
        Rectangle rectangle;
        input >> rectangle.id >> rectangle.width >> rectangle.height;
        strip.addRectangleGreedy(rectangle);
    }

    input.close();
    // Sigue con el resto del código
    return 0;
}
```



Todas las instancias deben seguir el formato:

n

W

$id_1\ w_1\ h_1$

$[...]$

$id_n\ w_n\ h_n$

Restricciones

Restricción 1

Los elementos no pueden superar el ancho `W` de la región, *asumiendo que siempre se pueden colocar.*

```
if (rectangle.width < rectangle.height && rectangle.height <= fixedWidth) {
    for (int x = 0; x <= fixedWidth - rectangle.height; x++) {
        // Sigue con el resto del código
    }
} else {
    for (int x = 0; x <= fixedWidth - rectangle.width; x++) {
        //Sigue con el resto del código
    }
}
```

Restricciones

Restricción 2

Los elementos no pueden solaparse entre sí.

```
bool Strip::canBePlaced(Rectangle rectangle, int posX, int posY, int orientation) {
    for (const Rectangle &r : rectangles) {
        bool overlapX = (posX + rectangle.width + orientation * (rectangle.height - rectangle.width) <= r.position.x) ||
                        (r.position.x + r.width + r.position.r * (r.height - r.width) > posX);
        bool overlapY = (posY + rectangle.height + orientation * (rectangle.width - rectangle.height) <= r.position.y) ||
                        (r.position.y + r.height + r.position.r * (r.width - r.height) > posY);
        if (!(overlapX || overlapY)) {
            return false;
        }
    }
    return true;
}
```

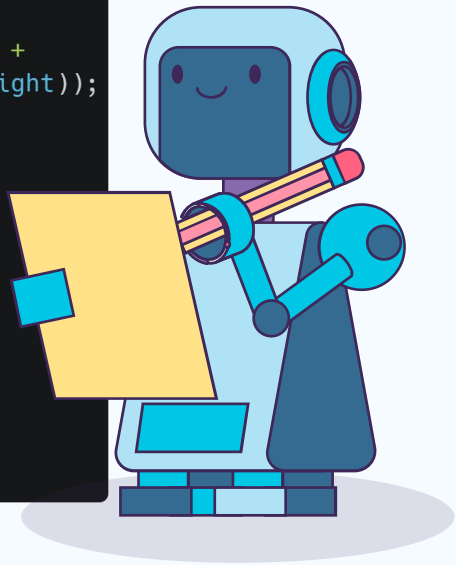
Solución inicial

```
void Strip::addRectangleGreedy(Rectangle rectangle) {
    int bestY = INT32_MAX;
    int bestX = 0;
    if (rectangle.width < rectangle.height && rectangle.height <= fixedWidth) {
        for (int x = 0; x <= fixedWidth - rectangle.height; x++) {
            int y = 0;
            for (const Rectangle &r : rectangles) {
                // Revisa si hay rectángulos posicionados en ese x
            }
            if (y < bestY && canBePlaced(rectangle, x, y, 1)) {
                // Actualiza los valores
            }
        }
        rectangle.position = {bestX, bestY, 1};
        rectangles.push_back(rectangle);
    } else {
        // Sigue la misma lógica que el anterior
    }
```

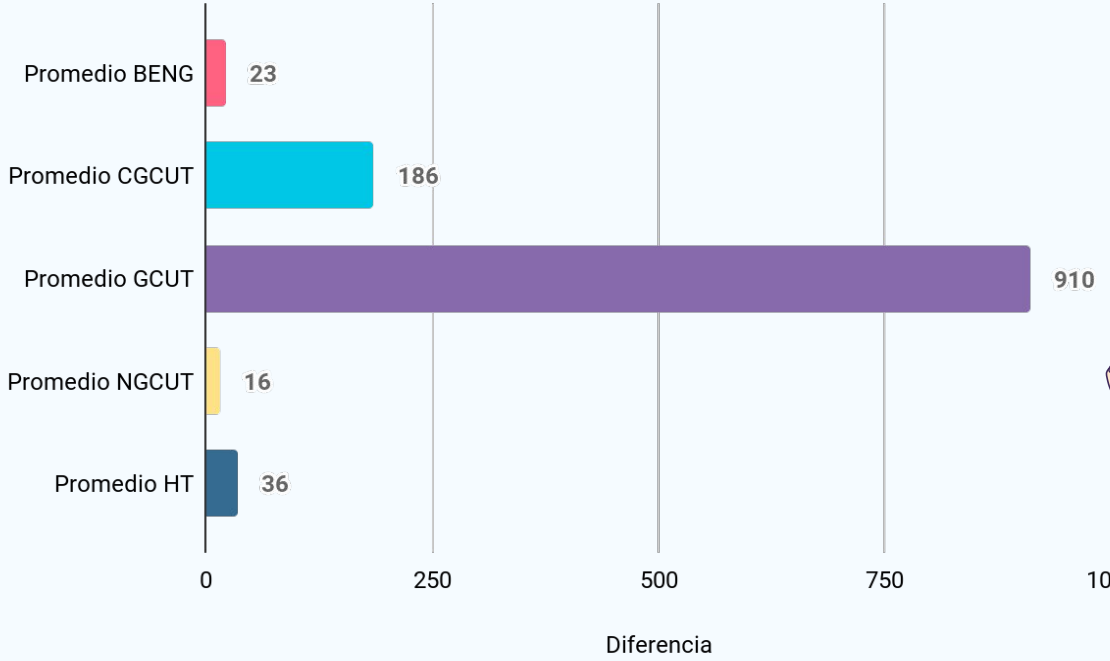
Función de evaluación

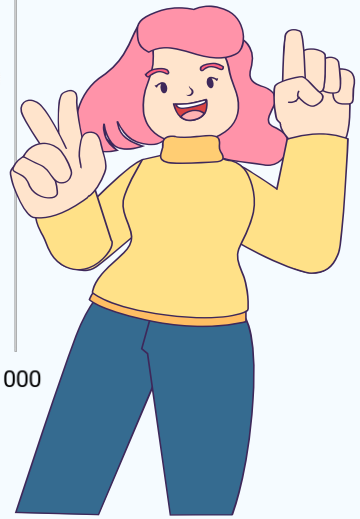
```cpp
int Strip::getHeight() {
    int maxHeight = 0;
    for (const Rectangle &rectangle : rectangles) {
        maxHeight = max(maxHeight, rectangle.position.y + rectangle.height +
                    rectangle.position.r * (rectangle.width - rectangle.height));
    }
    return maxHeight;
}

int Strip::evaluationFunction() {
    int totalArea = getHeight() * fixedWidth;
    int totalRectangleArea = 0;
    for (const Rectangle &rectangle : rectangles) {
        totalRectangleArea += rectangle.width * rectangle.height;
    }
    return totalArea - totalRectangleArea;
}
```



Comparaciones de instancias conocidas





Martello, S., Monaci, M., & Vigo, D. (2003). An exact approach to the strip-packing problem. INFORMS journal on Computing, 15(3), 310-319.

Resultados instancias de prueba

2 - 5		6 - 10		11 - 15		16 - 20	
1.	11 - 86 - 0,005	1.	30 - 101 - 0,020	1.	35 - 344 - 0,024	1.	37 - 232 - 0,019
5.	4 - 26 - 0,022	5.	23 - 116 - 0,022	5.	38 - 127 - 0,021	5.	35 - 324 - 0,021
10.	24 - 83- 0,018	10.	36 - 65 - 0,019	10.	35 - 127 - 0,023	10.	57 - 660 - 0,022
15.	5 - 25 - 0,021	15.	18 - 107 - 0,019	15.	50 - 249 - 0,019	15.	48 - 287 - 0,022
20.	13 - 112 - 0,022	20.	22 - 170 - 0,024	20.	53 - 83 - 0,019	20.	52 - 101 - 0,019
25.	2 - 16 - 0,022	25.	9 - 73 - 0,023	25.	36 - 376 - 0,021	25.	36 - 379 - 0,019
30.	10 - 56 - 0,018	30.	17 - 35 - 0,019	30.	30 - 167 - 0,019	30.	48 - 288 - 0,018

Siguiendo la forma **`altura - área inutilizada - tiempo en segundos`**.

Trabajo restante

- Implementar `Hill Climbing Alguna Mejora`.
- Definir movimiento y generar el vecindario.
- Revisar casos de prueba con problemas.
- Mejorar eficiencia.
- Futuras entregas.