

# Advanced Data Structure

## Project VI

### Strip Packing

Group3: 冯恺睿, 洪奕迅, 朱易凡

2024- 12- 11

#### **Abstract**

The strip packing problem is a classic 2-dimensional geometric minimization problem. And in this project, we would implement some classic strip packing algorithms to solve the problem. We would test each algorithm with different dataset and examine its approximation is whether right to the theoretic analysis.

# Index

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Algorithms</b>	<b>3</b>
2.1	Next Fit Decreasing Height . . . . .	3
2.1.1	Approximation . . . . .	3
2.1.2	Time Complexity . . . . .	5
2.2	First Fit Decreasing Height . . . . .	5
2.2.1	Approximation . . . . .	5
2.2.2	Time Complexity . . . . .	7
<b>3</b>	<b>Testing Results</b>	<b>7</b>
3.1	Runtime Analysis . . . . .	8
3.2	Approximation Analysis . . . . .	8
<b>4</b>	<b>Conclusion</b>	<b>9</b>

# 1 Introduction

The strip packing problem is a 2-dimensional geometric minimization problem. Given a set of axis-aligned rectangles and a strip of bounded width and infinite height, determine an overlapping-free packing of the rectangles into the strip, minimizing its height. And we need to run those algorithm in different situations to test their approximation.

## 2 Algorithms

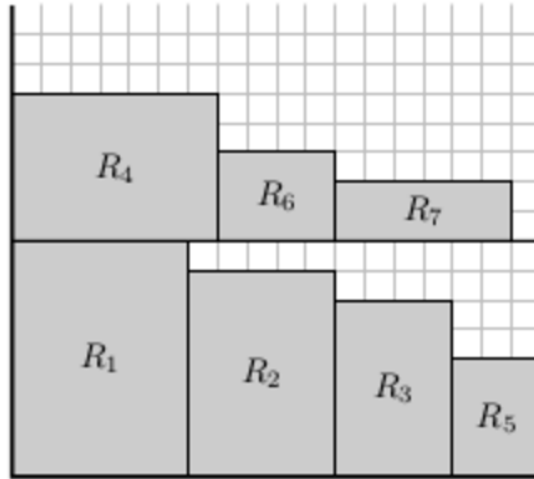
In this section, we use the below example to show how those different algorithms run:

$i$	1	2	3	4	5	6	7
$w(i)$	6	5	4	7	3	4	6
$h(i)$	8	7	6	5	4	3	2

And for convenience, we suppose that the width of the strip is 1 and all items' width less than 1(In implementation, the width is all integers, the proof equals to divide all that width with the total width of the strip.).

### 2.1 Next Fit Decreasing Height

Let  $\mathcal{L}$  be the given items  $R_1, R_2 \dots R_n$ . The NFDH is a quite natural algorithm to store those items. Firstly, we sort all the items in a decreasing order and then put them one by one from the bottom left to the right. And once the next item exceed the right boundary, we put it on a new level and repeat that step until all items are placed. For the given example, the packing would be like the follwing picture:



Graph 1:  $NFDH(\mathcal{L})$

#### 2.1.1 Approximation

It is easy to find that the utility of space in this algorithm is quite low since we never use the level that under the top, so is its approximation. In fact, we have the theory that:

$$NFDH(\mathcal{L}) \leq 2 \times OPTIMAL(\mathcal{L}) + C$$

And the approximation constant 2 is tight.

Now we are going to prove that lemma. We denote  $B_i$  as the  $i$ -th level and  $t$  as the number of levels, or say each first item in the  $i$ -th level has the height of  $B_i$ , which we call it as  $H_i$ . And we define  $A_i$  as the total area of the  $i$ -th level while  $x_i$  and  $y_i$  be the first item's width in  $B_i$  and total width of  $B_i$ . Since we always new a level when last level cannot afford the new item then we have:

$$y_i + x_{i+1} > 1 \quad \text{for } 1 < i < t \quad (1)$$

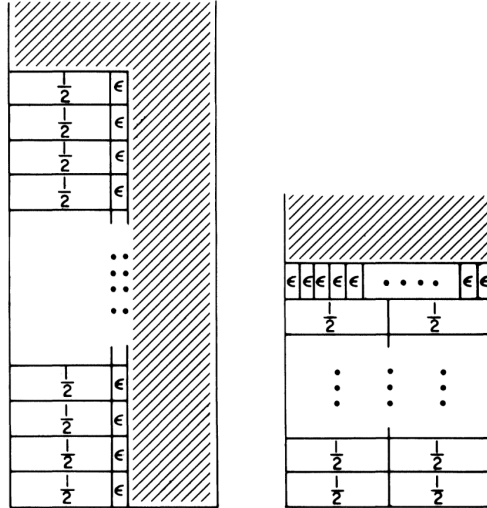
And we know all items in  $B_i$  has a height greater than  $H_{i+1}$ , so:

$$A_i + A_{i+1} \geq H_{i+1}(y_i + x_{i+1}) > H_{i+1} \quad (2)$$

Now we can get the approximation:

$$\begin{aligned} NFDH(\mathcal{L}) &= \sum_{i=1}^t H_i \leq H_1 + \sum_{i=1}^{t-1} A_i + \sum_{i=2}^t A_i \\ &\leq H_1 + 2A \\ &\leq 1 + 2 \, OPTIMAL(\mathcal{L}) \end{aligned} \quad (3)$$

Worst case of the NFDH is we always create a new unnecessary level like the following graph:



Graph 2: Worst case

In this case,  $OPTIMAL(\mathcal{L}) = \frac{1}{4}n + 1$  and  $NFDH(\mathcal{L}) = \frac{n}{2}$ , so we can always find a big enough  $n$  to make approximation arbitrarily close to 2, which mean **NFDH is a 2-approximation algorithm** and the constant 2 is tight.

### 2.1.2 Time Complexity

We use the STL function `sort()`, so the time complexity of sorting is  $\mathcal{O}(\log N)$  and we use `erase()` in the loop:

```

1 Function next_fit_decreasing_height(rectangles_list arr_rect, container_width width):
2     Sort arr_rect by the second element (height) in descending order using cmp_1
3     Set total_height = 0    // To store the total height used
4     Set remaining_width = 0  // To track the remaining width in the current container
5     While arr_rect is not empty:
6         Take the first rectangle t from arr_rect
7         Remove the first rectangle from arr_rect
8         If remaining_width >= t.width:
9             Update remaining_width by subtracting t.width
10        Else:
11            Update total_height by adding t.height
12            Set remaining_width = width - t.width    // Start a new container with this rectangle
13
14    Return total_height

```

So the total complexity equals to the complexity of the loop, which is  $\mathcal{O}(\log N)$

## 2.2 First Fit Decreasing Height

### 2.2.1 Approximation

This algorithm works similar to the NFDH algorithm. However, when placing the next item, the algorithm scans the levels from bottom to top and places the item in the first level on which it will fit. We only new a level when there's no space for the new item. It satisfies:

$$FFDH(\mathcal{L}) \leq 1.7 \text{ OPTIMAL}(\mathcal{L}) + 1 \quad (4)$$

The proof is based on the analogous proof for the First-Fit algorithm of one-dimensional bin-packing, we need to introduce a weight function:

$$W(x) = \begin{cases} \frac{6}{5}x & \text{if } 0 \leq x \leq \frac{1}{6} \\ \frac{9}{5}x - \frac{1}{10} & \text{if } \frac{1}{6} < x \leq \frac{1}{3} \\ \frac{6}{5}x + \frac{1}{10} & \text{if } \frac{1}{6} < x \leq \frac{1}{2} \\ \frac{6}{5}x + \frac{4}{10} & \text{if } \frac{1}{2} < x \leq 1 \end{cases} \quad (5)$$

We extend this function to rectangles by  $W(r) = W(w(r))$ , and set  $A = \sum_{r \in \mathcal{L}} h(r) \cdot W(r)$ . It is proved in that no collection of numbers  $x$  summing to 1 or less can have  $W(x)$  summing to more than 1.7. We can apply this result to our case by cutting the optimal packing into horizontal slices. And summing all the slices we get  $A \leq 1.7 \text{ OPTIMAL}(\mathcal{L})$ .

Then to prove our theory, we only need to prove that  $A \geq FFDH(\mathcal{L}) - 1$ . Let  $T_1$  and  $T_2$  the set of blocks whose first rectangle has width no more than or greater than  $\frac{1}{2}$ . And we define  $H_i$  as the total height of all blocks that satisfy  $T_i$ . Since the first item in block  $B_i$  has height  $H_i$ , and due

to our normalization assumptions, we have

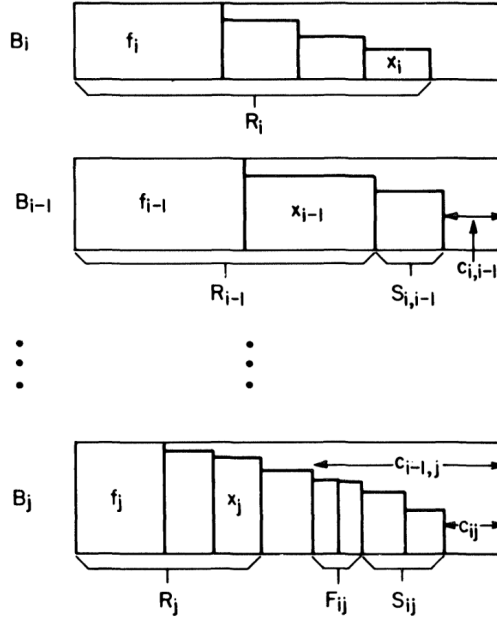
$$\sum_{B \in T_2} \sum_{r \in B} h(r) \cdot W(r) \geq H_2 \quad (6)$$

Now to reach the end, we only need to prove

$$\sum_{B \in T_1} \sum_{r \in B} h(r) \cdot W(r) \geq H_1 - 1 \quad (7)$$

We need to introduce a lot of definitions to help finishing that proof. Let  $L_1$  be the rectangles in blocks of  $T_1$ . Note that the FFDH packing of  $L_1$  would be similar to  $T_1$ . And now we denote  $H_{t+1} = 0$  by convention. And we now divide rectangles into regular and fallback while  $f_i$  and  $g_i$  represents the first and last regular rectangle in  $B_i$  and  $R_i$  be the set of all regular rectangles in  $B_i$ . Moreover, we define  $F_{ij}$  be the set of all fallback rectangles in  $B_j$  before  $f_i$  but after  $g_{i-1}$  ( $0 \leq i \leq t$ ,  $0 \leq j \leq i$ ). Similarly we define  $S_{ij}$  of all fallback items in  $B_i$  after  $f_i$  but before  $g_i$ . Note that  $F_{ij}$  and  $S_{ij}$  are disjoint and  $F_{i,i-1} = \emptyset$ , so we have

$$L_1 = \cup_{i=1}^t [R_i \cup \cup_{j < i} F_{ij} \cup \cup_{j < i} S_{ij}] \quad (8)$$



Graph 3: An illustration figure to show the set relationship

Define  $c_{ij}$  to be the width of the empty space at the bottom right end of  $B_j$ . when the last regular item is packed in  $B_i$ . And in the picture there always holds relations:

$$\begin{aligned} w(f_i) &\geq c_{ij} + \sum_{r \in S_{ij}} w(r), & 1 < i \leq t \text{ and } 1 \leq j < i \\ c_{ij} &= c_{i-1,j} - \sum_{r \in S_{ij} \cup F_{ij}} w(r), & 2 < i \leq t \text{ and } 1 \leq j < i \end{aligned} \quad (9)$$

Define that  $c_i = \max_{j < i} c_{ij}$ , we need to use a lemma that for every  $i \in (1, t)$  exists  $i' < i$  satisfies

$$\sum_{r \in R_{i-1}} W(r) + \frac{6}{5} \sum_{r \in F_{ii'} \cup S_{ii'}} w(r) \geq 1 + \frac{6}{5}(c_{i-1} - c_{ii'}) \quad (10)$$

We observe that

$$\begin{aligned} \sum_{r \in L_1} h(r) \cdot W(r) &\geq \sum_{i=1}^t H_{i+1} \sum_{r \in R_i} W(r) + \sum_{i=1}^t W(f_i)(H_i - H_{i+1}) \\ &\quad + \sum_{i=2}^t H_i \sum_{r \in R_{ii'}} W(r) + \sum_{i=2}^t H_{i+1} \sum_{r \in S_{ii'}} W(r). \end{aligned} \quad (11)$$

Then we use the lemma to get

$$\begin{aligned} \sum_{i=2}^t \left[ H_i \sum_{r \in R_{i-1}} W(r) + \frac{6}{5}(H_i - H_{i+1}) \right] &\left( c_{ii'} + \sum_{r \in S_{ii'}} w(r) \right) + \frac{6}{5} H_i \sum_{r \in F_{ii'}} w(r) + \frac{6}{5} H_{i+1} \sum_{r \in S_{ii'}} w(r) \\ &\sum_{i=2}^t \left[ H_i \sum_{r \in R_{i-1}} W(r) + \frac{6}{5}(H_i - H_{i+1})c_{ii'} + \frac{6}{5} H_i \sum_{r \in F_{ii'} \cup S_{ii'}} \right] \\ &\sum_{i=2}^t [H_i + \frac{6}{5} H_i (c_{i-1} - c_{ii'}) + \frac{6}{5} (H_i - H_{i+1})c_{ii'}], \end{aligned} \quad (12)$$

which means

$$\begin{aligned} \sum_{r \in L_1} h(r) \cdot W(r) &\equiv \sum_{i=2}^t \left[ H_i - \frac{6}{5} H_{i+1} c_{ii'} + \frac{6}{5} H_i c_{i-1} \right] \\ &H(T_1) - H_1 - \frac{6}{5} \sum_{i=2}^t H_{i+1} c_{ii'} + \frac{6}{5} \sum_{i=1}^{t-1} H_{i+1} c_i \\ &H(T_1) - H_1 + \frac{6}{5} \sum_{i=2}^{t-1} H_{i+1} (c_i - c_{ii'}) H(T_1) - 1, \end{aligned} \quad (13)$$

and by our normalization assumption, here comes to the end, which means that **FFDH is a 1.7-approximation algorithm.**

### 2.2.2 Time Complexity

Like the NFDH, FFDH consists of sorting and putting. All details are the same as NFDH except we need to check each level when putting the item, which takes a complexity of constant. So the time complexity is also  $\mathcal{O}(N^2)$

## 3 Testing Results

We design a data genrator that can generates  $N$  rectangles whose width are within the strip width. We choose  $N = 10, 50, 100, 500, 1000, 5000, 10000$  and some different boxes' width  $w = 50, 100, 500$ .

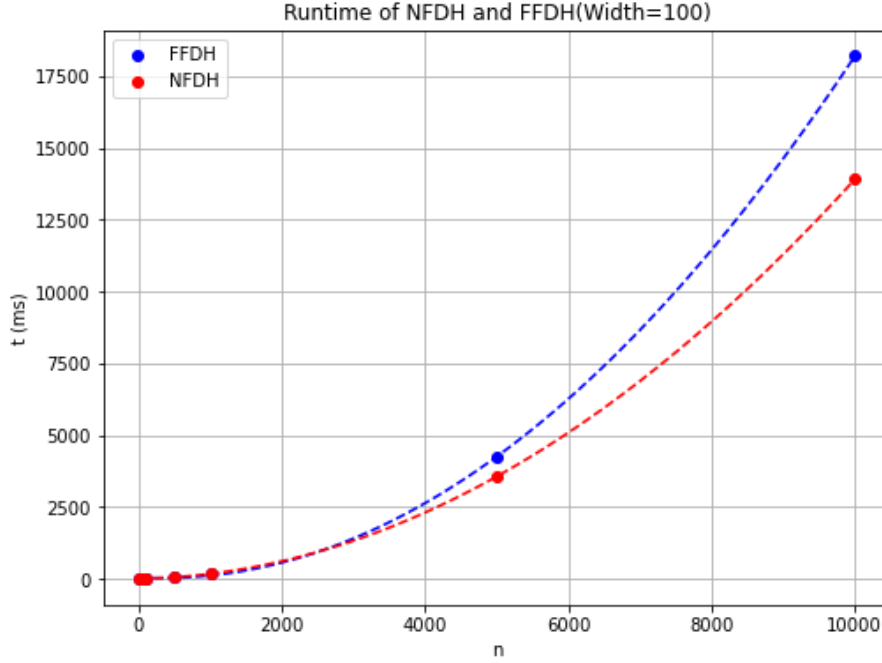
### 3.1 Runtime Analysis

The theoretical complexity of FFDH and NFDH are the same, but it is clear that in FFDH we do branch operations, which means a bigger constant, the testing results show the same trend.

Since the width of the strip won't influence the runtime so much, so here only show the data of a width of 100:

Numbers	10	50	100	500	1000	5000	10000
<b>FFDH/TIME(ms)</b>	9.5	10.5	13.6	71	190	4219	18196
<b>NFDH/TIME(ms)</b>	8.8	9.6	13.2	56	159	3565	13900

Figure 1: FFDH/TIME vs. NFDH/TIME (Transposed)



Graph 4: Runtime comparion(width=100)

### 3.2 Approximation Analysis

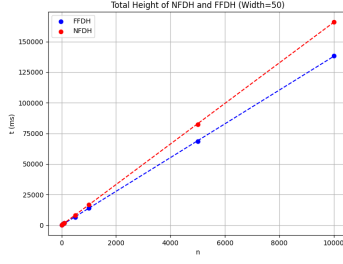
See the data in attached folder, here we only show the figure. To inspect our approximation analysis, we mainly see two point, whether the ascending trend of the algorithm in  $H$  is nearly linear in big  $N$  and whether it satisfies about  $\frac{H(FFDH)}{H(NFDH)} \approx 0.85$  in big  $N$ . (In small  $N$ , FFDH would degenerate into almost NFDH). Here's the result for three widths:

Numbers	10	50	100	500	1000	5000	10000
<b>FFDH/NFDH(width=50)</b>	0.9283	0.8452	0.8468	0.8342	0.8371	0.8313	0.8326
<b>FFDH/NFDH(width=100)</b>	0.8920	0.8678	0.8931	0.8294	0.8231	0.8189	0.8188
<b>FFDH/NFDH(width=500)</b>	0.9649	0.7846	0.8035	0.8077	0.7964	0.8018	0.8005

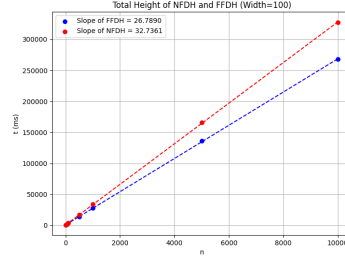
Figure 2: Height Ratio of FFDH and DFDH

And here are the detailed figures:

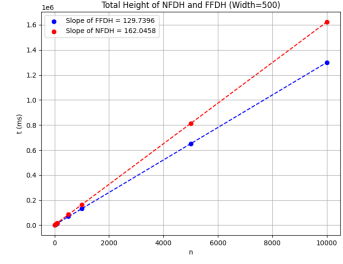




(a) Width=50



(b) Width=100



(c) Width=500

Graph 5: Height of NFDH and FFDH in different widths

## 4 Conclusion

In this project, we do theoretical analysis of the NFDH and FFDH algorithm to partly solve the NP problem - strip packing in a finite time. And by using a data generator to examine our program in datasets of different scales, we check that our theoretical analysis of complexity and approximation is right.