

Projeto 03 - Binpacking e algoritmos aproximados

Arthur Wallace Silva Lopes

¹Instituto Federal de Brasília (IFB)

1. Introdução

O problema *Binpacking* consiste em, dado um conjunto de itens de diferentes pesos p e um cesto(*bin*) de capacidade c , definir o menor número possível de cestos necessários para armazenar todos esses itens. Entretanto, a versão ótima da solução para esse problema é **NP-Completa**, ou seja, não existe, até o presente momento, uma solução polinomial capaz de resolvê-lo completamente com 100% de assertividade. Dessa forma, é possível atingir soluções com assertividades satisfatórias, em relação à resolução ótima, em tempos de execuções razoáveis por meio de algoritmos aproximados, ou heurísticos.

Este projeto visa avaliar a implementação de alguns algoritmos aproximados mais conhecidos para se aproximar das resoluções do problema *Binpacking*, são eles: *First-fit*, *Next-fit* e *Best-fit* e comparar suas respectivas assertividades em relação à resolução ótima e também os tempos e custos de execução para os piores casos. Isso será realizado por meio da utilização dos arquivos de dados fornecidos contendo a quantidade total de itens, a capacidade de cada cesto e o peso de cada um dos itens a serem inseridos.

Dessa forma, ao fim desse trabalho, busca-se:

- Demonstrar que esses algoritmos possuem tempos viáveis de execução para o pior caso, por meio da análise assintótica;
- Provar que os algoritmos citados possuem um raio de aproximação 2 em relação às soluções ótimas;
- Comparar as assertividades desses algoritmos com a solução ótima verificando se o raio de aproximação se confirma e, também, demonstrar que os algoritmos aproximados podem ser uma possível alternativa para resolver, de forma satisfatória, o problema *Binpacking*.

2. Abordagens aproximadas para o Problema BIN-PACKING-OPTM

2.1. First-Fit

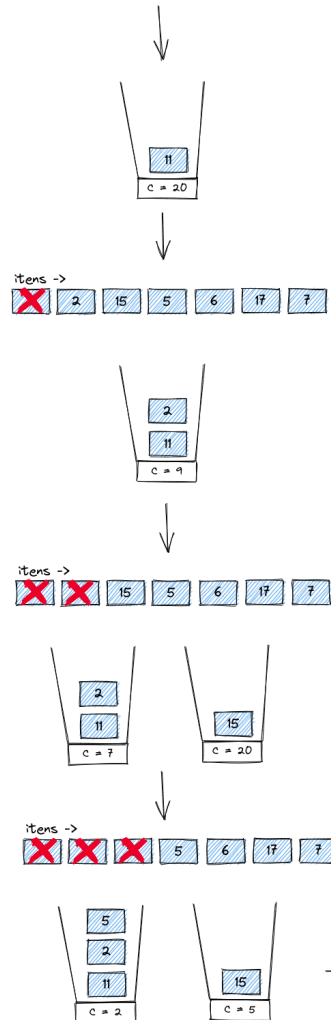
A primeira abordagem utilizada será o método **First-Fit**. Neste método, a lista com os itens não terá sua ordem alterada e cada um dos itens de peso p será alocado no primeiro cesto (o de menor índice) capaz de acomodá-lo, ou seja, onde a diferença entre a capacidade restante e o peso do item seja maior ou igual a zero ($C - p \geq 0$) e, caso ainda não exista esse cesto, um novo será criado e o item será alocado nele.

FIRST-FIT

itens ->

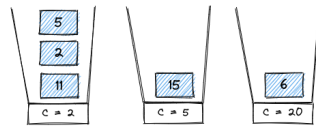
11	2	15	5	6	17	7
----	---	----	---	---	----	---

Os itens serão adicionados no primeiro cesto que possuir espaço disponível em ordem sequencial, da esquerda para a direita, e caso não haja espaço disponível no cesto, será criado um novo.



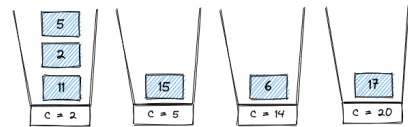
itens ->

X	X	X	X	6	17	7
---	---	---	---	---	----	---



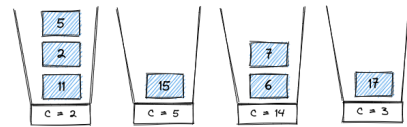
itens ->

X	X	X	X	X	17	7
---	---	---	---	---	----	---



itens ->

X	X	X	X	X	X	7
---	---	---	---	---	---	---



itens ->

X	X	X	X	X	X	X
---	---	---	---	---	---	---

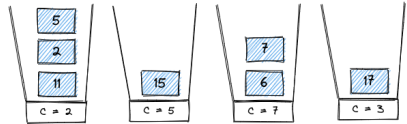


Figura 1. First-Fit exemplo

```
function firstFit(items, numItems, capacity) {  
    let buckets = [];  
  
    for (let i = 0; i < numItems; i++) {  
        let j;  
        for (j = 0; j < buckets.length; j++) {  
            if (buckets[j] > items[i]) {  
                buckets[j] -= items[i];  
                break;  
            }  
        }  
        if (j == buckets.length) buckets.push(capacity -  
        items[i]);  
  
        return buckets.length;  
    }  
}
```

Figura 2. First-Fit - código

Ao analisar a complexidade de execução do algoritmo **First-Fit**, conseguimos, através da análise assintótica, determinar o custo total desse método ao fim dessas iterações.

Primeiramente, é necessário realizar uma iteração sobre cada um dos itens do conjunto de dados, que possuirá custo $\Theta(n)$ para ser executada, onde n é a quantidade total de itens. Logo após, dentro dessa iteração, será realizado um novo loop sobre os cestos (*buckets*) já existentes, que também possuirá custo $\Theta(n)$ para ser executado.

Com isso, temos que o custo total dessa execução será de:

$$\begin{aligned}\Theta(n) * \Theta(n) &= \Theta(n * n) \\ \Theta(n * n) &= \Theta(n^2) \\ \Theta(n^2)\end{aligned}\tag{1}$$

Em relação ao raio de aproximação desse método, temos que ele possuirá uma cota superior de valor 2, já que a quantidade de cestos utilizados nunca será superior a $1.7M$ onde M representa a quantidade mínima de cestos necessários para resolução ótima do problema. Pois, supondo que o método first-fit utilize n cestos, teremos que ao menos $(n - 1)$ cestos estarão com mais da metade da sua capacidade preenchida, já que nunca teremos 2 cestos com menos da metade da sua capacidade preenchida.

2.2. Next-Fit

A próxima abordagem utilizada será o método **Next-Fit**. Neste método, a lista com os itens não terá sua ordem alterada e cada um dos itens de peso p será alocado ou no cesto atual (o último utilizado na iteração anterior) caso ele seja capaz de acomodá-lo ou um novo cesto será criado e o item será alocado nele.

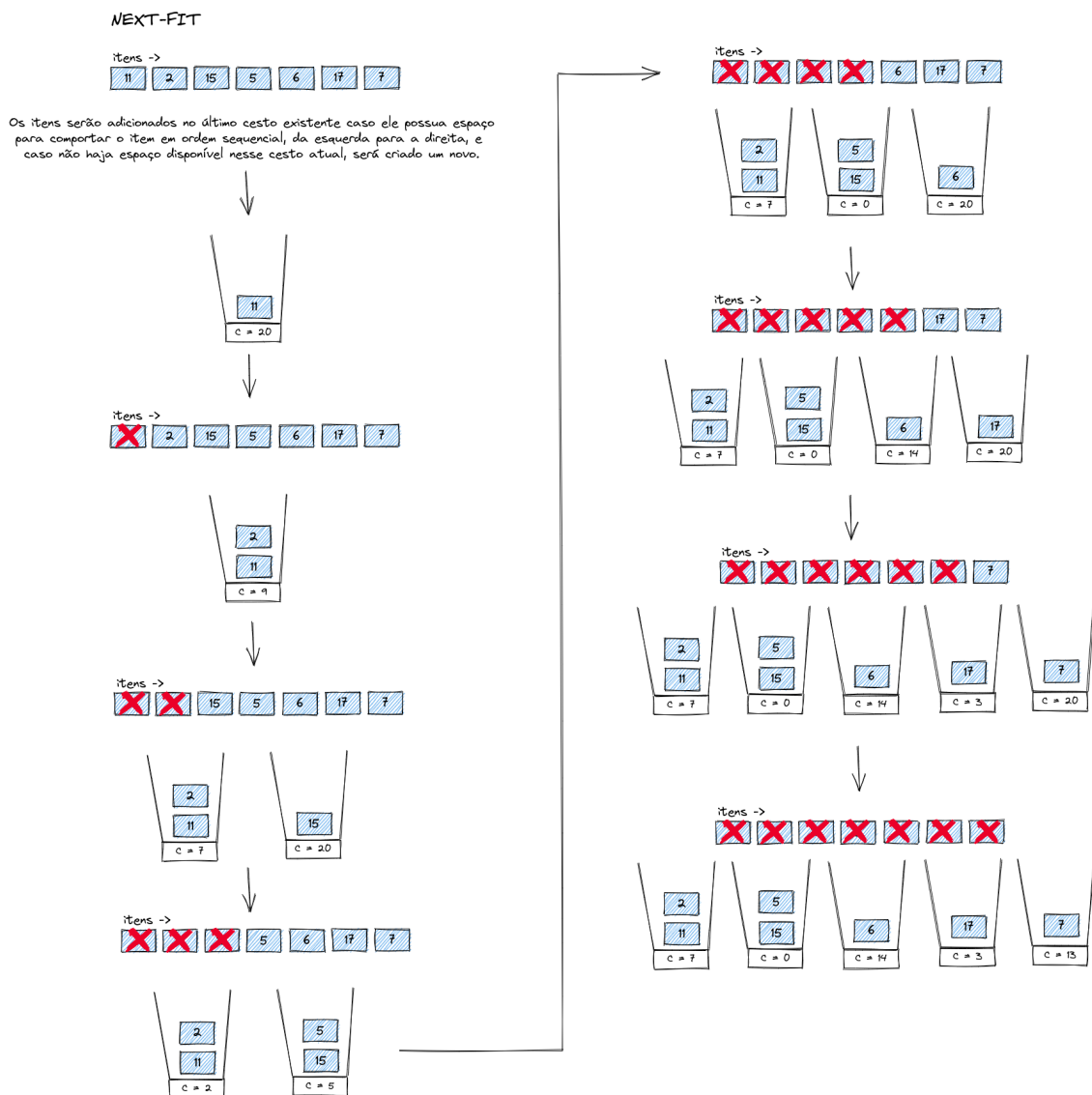
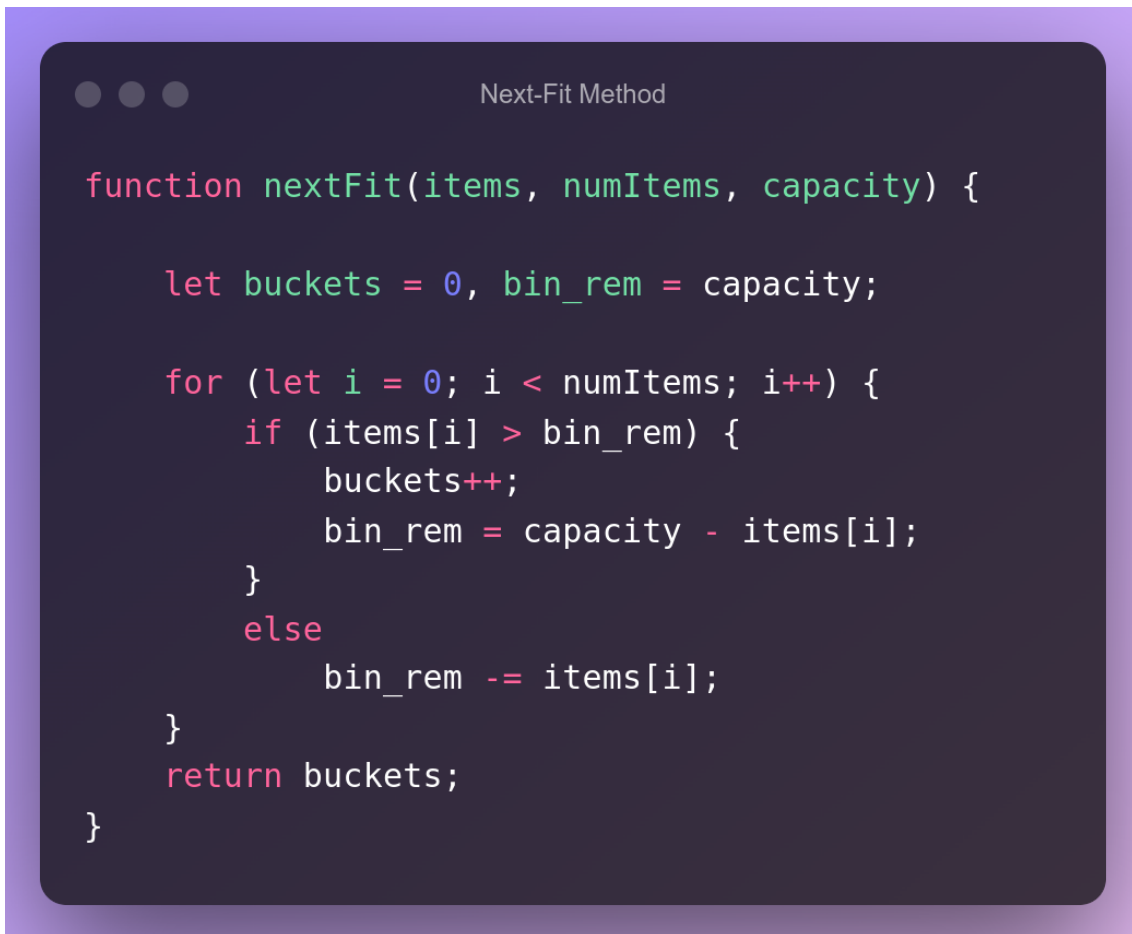


Figura 3. Next-Fit exemplo



```
function nextFit(items, numItems, capacity) {  
  
    let buckets = 0, bin_rem = capacity;  
  
    for (let i = 0; i < numItems; i++) {  
        if (items[i] > bin_rem) {  
            buckets++;  
            bin_rem = capacity - items[i];  
        }  
        else  
            bin_rem -= items[i];  
    }  
    return buckets;  
}
```

Figura 4. Next-Fit - código

Ao analisar a complexidade de execução do algoritmo **Next-Fit**, conseguimos, através da análise assintótica, determinar o custo total desse método ao fim dessas iterações.

Neste método, é realizado apenas uma iteração sobre a quantidade total de itens do conjunto de dados, para percorrer cada um desses itens e realizar as comparações necessárias. Essa iteração possuirá custo $\Theta(n)$ para ser executada, onde n representa a quantidade total de itens.

Com isso, temos que o custo total dessa execução será de: $\Theta(n)$.

Em relação ao raio de aproximação desse método, temos que ele possuirá uma cota superior de valor 2, já que, ao considerar dois cestos adjacentes a soma dos itens nesses dois cestos deve ser maior do que a capacidade dos cestos C . Pois, caso contrário, o método Next-Fit teria adicionado todos os itens do segundo cesto no primeiro. Então, podemos afirmar que no máximo metade do espaço será desperdiçado, ou seja, se usaria no máximo $2M$, onde M representa a quantidade mínima de cestos necessários para resolução ótima do problema.

2.3. Best-Fit

Por fim, a próxima abordagem será utilizando o método **Best-Fit**, que se baseia na premissa de ir adicionando os itens disponíveis em sequência de forma com que se priorize os adicionar nos cestos com a menor capacidade capaz de acomodá-los, ou seja, cestos em que ao colocar o item no seu interior a capacidade restante se aproxime o máximo possível de 0 e caso não exista espaço disponível em nenhum dos cestos existentes será criado um novo e o item será alocado nele.

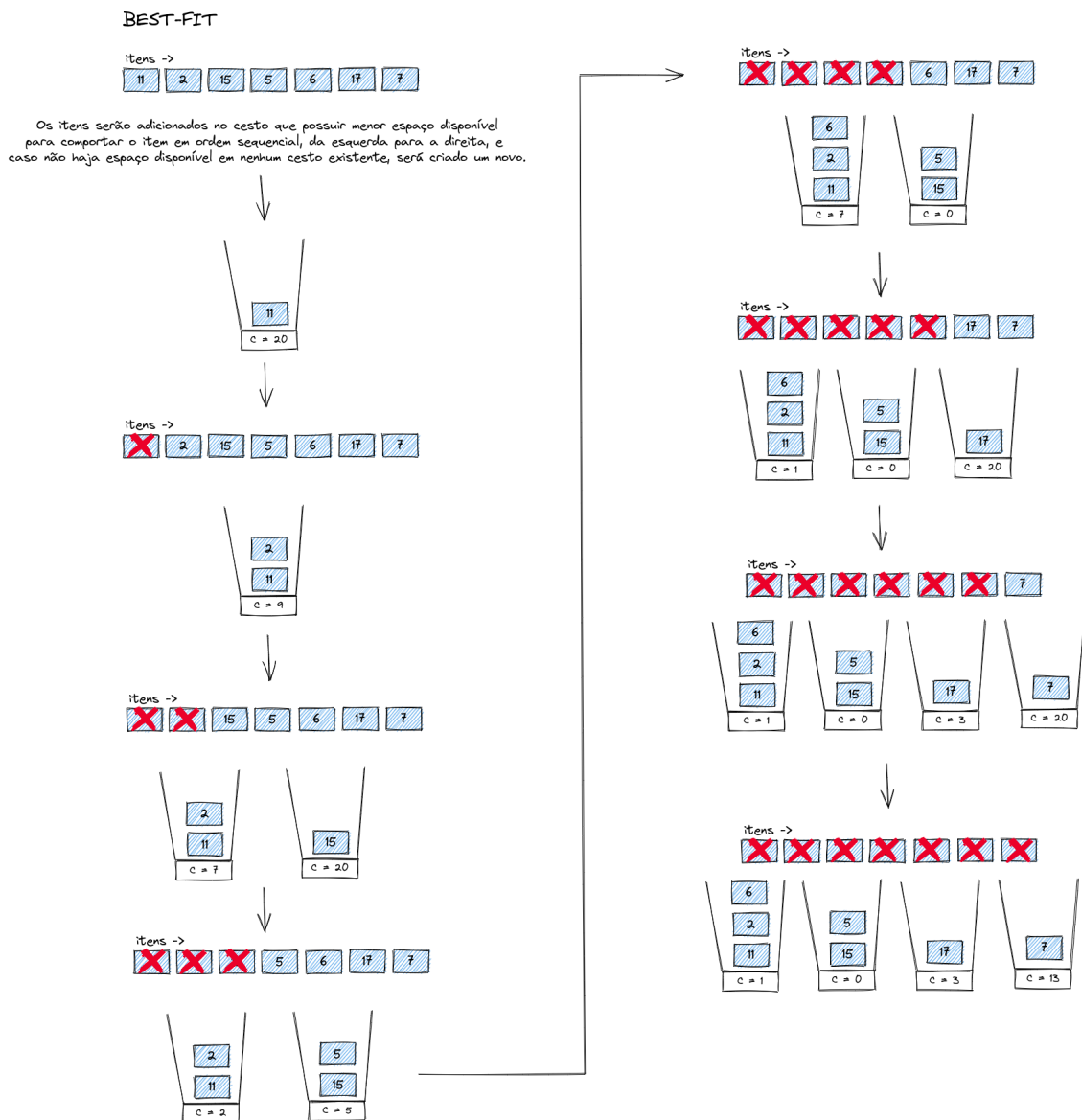


Figura 5. Best-Fit exemplo



```
function bestFit(items , numItems , capacity) {  
  
    let buckets = 0;  
    let bin_rem = Array(numItems).fill(0);  
  
    for (i = 0; i < numItems; i++) {  
  
        let j;  
        let min = capacity + 1, bi = 0;  
  
        for (j = 0; j < buckets; j++) {  
            if (bin_rem[j] >= items[i] && bin_rem[j] - items[i] < min)  
            {  
                bi = j;  
                min = bin_rem[j] - items[i];  
            }  
        }  
  
        if (min == capacity + 1) {  
            bin_rem[buckets] = capacity - items[i];  
            buckets++;  
        } else  
            bin_rem[bi] -= items[i];  
    }  
    return buckets;  
}
```

Figura 6. Best-Fit - código

Ao analisar a complexidade de execução do algoritmo **Best-Fit**, conseguimos, através da análise assintótica, determinar o custo total desse método ao fim dessas iterações.

Primeiramente, é necessário realizar uma iteração sobre cada um dos itens do conjunto de dados, que possuirá custo $\Theta(n)$ para ser executada, onde n é a quantidade total de itens. Logo após, dentro dessa iteração, será realizado um novo loop sobre os cestos (*buckets*) já existentes, que também possuirá custo $\Theta(n)$ para ser executado.

Com isso, temos que o custo total dessa execução será de:

$$\begin{aligned}\Theta(n) * \Theta(n) &= \Theta(n * n) \\ \Theta(n * n) &= \Theta(n^2) \\ &\Theta(n^2)\end{aligned}\tag{2}$$

Em relação ao raio de aproximação desse método, ele será semelhante ao método

First-Fit onde também temos que ele possuirá uma cota superior de valor 2, já que a quantidade de cestos utilizados nunca será superior a $1.7M$ onde M representa a quantidade mínima de cestos necessários para resolução ótima do problema. Pois, supondo que o método best-fit utilize n cestos, teremos que ao menos $(n - 1)$ cestos estarão com mais da metade da sua capacidade preenchida, já que nunca teremos 2 cestos com menos da metade da sua capacidade preenchida.

3. Materiais e Métodos

Para a realização desse projeto, utilizamos a linguagem *NodeJs* para implementação dos algoritmos e fizemos uso dos 17 arquivos de dados fornecidos como referência, os quais possuem cada um a **quantidade de itens**, a **capacidade dos cestos** e os **pesos** de cada um dos itens. Também utilizamos o arquivo contendo as soluções ótimas para cada um dos arquivos de dados a fim de utilizar essas informações para viabilizar a comparação de assertividade para cada um dos métodos de aproximação.

Para realizar os testes, estamos executando o algoritmos por 3 vezes e, a partir dessas execuções, podemos calcular a média dos tempos de execução e também a proximidade alcançada pelos métodos em relação à resolução ótima e gerar, assim, a tabela de assertividade em relação às soluções ótimas e o gráfico de comparação dos tempos de execução para cada um dos métodos.

4. Análise de Dados

Ao analisar os resultados obtidos na realização dos testes, podemos perceber que todos os métodos de aproximação utilizados apresentaram assertividades satisfatórias ao compará-las com as soluções ótimas para cada arquivo, sendo que os métodos **First-Fit** e **Best-Fit** se mostraram muito assertivos com aproximações sempre acima dos **90%** até mesmo para os casos com maiores quantidade de itens, enquanto o método **Next-Fit** sofreu algumas oscilações, variando de assertividades próximas aos **70%** para os piores casos e de **93%** para os melhores casos.

QntItens	FirstFit	NextFit	BestFit
'57'	'100.00%'	'93.75%'	'100.00%'
'60'	'100.00%'	'88.89%'	'100.00%'
'86'	'96.00%'	'72.73%'	'96.00%'
'91'	'95.24%'	'86.96%'	'95.24%'
'92'	'94.12%'	'88.89%'	'94.12%'
'96'	'95.83%'	'82.14%'	'95.83%'
'111'	'96.43%'	'87.10%'	'96.43%'
'114'	'96.55%'	'84.85%'	'96.55%'
'119'	'92.31%'	'92.31%'	'92.31%'
'141'	'91.67%'	'91.67%'	'91.67%'
'142'	'93.75%'	'88.24%'	'93.75%'
'144'	'93.33%'	'93.33%'	'93.33%'
'153'	'94.12%'	'88.89%'	'94.12%'
'163'	'92.31%'	'92.31%'	'92.31%'
'164'	'93.33%'	'93.33%'	'93.33%'
'228'	'92.86%'	'92.86%'	'92.86%'
'239'	'95.24%'	'90.91%'	'95.24%'

Figura 7. Tabela de assertividade dos métodos de aproximação

Nesse sentido, também podemos analisar os tempos de execução que foram necessários para cada um desses métodos através do gráfico de linhas gerado comparando as três alternativas de resolução. Podemos perceber que o tempo de execução do **Next-Fit** se mostra realmente próximo ao linear atingindo, até mesmo nos piores casos, um tempo menor que **0.05 ms**. Já os métodos **First-Fit** e **Best-Fit** tiveram oscilações de tempo significativas de acordo com cada conjunto de itens, chegando a atingir quase **0.25 ms** nos piores casos.

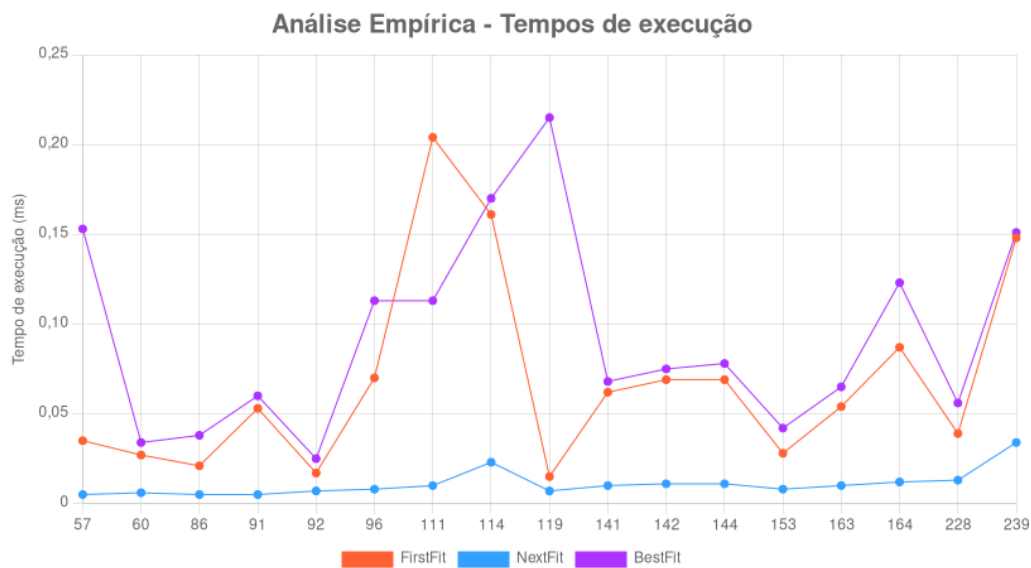


Figura 8. Gráfico de tempos de execução

O código para esse projeto está disponível em: https://github.com/ArthurWallaceIFB/AnaliseAlgoritmos_Projeto3

5. Considerações finais

Por fim, após a realização desse projeto, podemos chegar às seguintes conclusões:

- Com base nos gráficos apresentados e na análise assintótica, foi possível perceber que os algoritmos abordados atingiram resultados satisfatórios se tratando de assertividade em relação aos resultados ótimos. Logo, ao se buscar uma solução polinomial para resolver o problema de *binpacking* esses métodos se provam uma excelente opção a ser utilizada.
- Também foi possível provar que todos os métodos utilizados possuem um raio de aproximação 2 em relação às soluções ótimas, já que o máximo atingido pelos métodos First-Fit e Best-Fit será de $1.7M$ e para o de Next-Fit será de $2M$. Onde M representa a quantidade mínima de cestos para a solução ótima.

Dessa forma, ao confrontar a análise teórica dos métodos com os resultados obtidos, mostra-se evidente que as complexidades de execução entre os métodos são bem diferentes, já que, enquanto o método Next-Fit será dependente apenas da quantidade

de itens (n) linearmente, possuindo complexidade $\Theta(n)$, os métodos First-Fit e Best-Fit possuirão, também, influência apenas da quantidade de itens (n) mas em forma de potenciação, possuindo complexidade $\Theta(n^2)$.

6. Referências Bibliográficas

Algoritmos de Aproximação para o Problema de Empacotamento. [S. l.], 2 nov. 2015. Disponível em: <https://sites.icmc.usp.br/andretta/ensino/aulas/sme0216-5826-2-15/grupo2.pdf>. Acesso em: 6 ago. 2022.

Approximation Algorithms Chapter 9: Bin Packing. [S. l.], 6 mar. 2006. Disponível em: <http://www.cs.ucr.edu/~neal/2006/cs260/piyush.pdf>. Acesso em: 6 ago. 2022.