Trabalho 2º Bimestre: Técnicas de Projeto e Análise de Algoritmos - Problema do Ensalamento

Eduarda Elger[1], **Ellen Carine Bonafin Marques** [1], **Heloisa Alves**[1]

[1]Colegiado de Ciência da Computação Campus de Cascavel - UNIOESTE

Abstract. *This article presents a comparative study between two approaches to the retraining problem: the greedy strategy and the "TC Greedy" strategy, which combines the Transform and Conquer (TC) algorithm with the greedy strategy. The main objective is to apply and corroborate concepts acquired about these project strategies, seeking to improve the distribution of classes in an academic environment.*

Resumo. *Este artigo apresenta um estudo comparativo entre duas abordagens para o problema de ensalamento: a estratégia gulosa e a estratégia "TC Greedy", que combina o algoritmo Transformar e Conquistar (TC) com a estratégia gulosa. O objetivo principal é aplicar e corroborar conceitos adquiridos sobre estas estratégias de projeto, buscando otimizar a distribuição de aulas em um ambiente acadêmico.*

1. Introdução

A otimização desempenha um papel fundamental na resolução de problemas complexos em diversas áreas, abrangendo desde a ciência da computação até a matemática aplicada. Dentro desses aspectos destacam-se algoritmos e técnicas que visam contribuir para a melhoria dos processos e a busca por soluções ótimas.

Nesse contexto, a estratégia gulosa se destaca pela sua simplicidade e eficiência, transformando-a em uma ferramenta robusta para abordar problemas de otimização em diversos cenários. Sua abordagem iterativa e habilidade em encontrar soluções de forma rápida a tornam atrativa para uma ampla variedade de aplicações.

Entretanto, os algoritmos gulosos baseiam suas decisões unicamente na informação disponível no momento, sem considerar os possíveis efeitos futuros dessas escolhas. Em outras palavras, esses algoritmos não revisam ou reconsideram as decisões tomadas, independentemente das consequências que possam surgir. Devido a essas características, esses algoritmos são, em geral, fáceis de conceber e implementar, demonstrando eficiência quando suas premissas são atendidas [Rocha and Dorini 2004].

Para a melhor visualização e compreensão desse tema, será analisado nesse artigo o comportamento de uma estratégia gulosa, o escalonamento de tarefas, comparando-o com sua versão utilizando o algoritmo de ordenação *QuickSort*.

O trabalho foi subdividido em: descrição do problema (seção 2), implementação (seção 3) contendo informações sobre a linguagem utilizada para a implementação, o ambiente de teste e informações sobre os dos dois algoritmos. Na seção 4 são apresentados a métrica utilizada para a avaliação das implementações propostas e os resultados; e na seção 5 as considerações finais.

2. Problema de Alocação de Salas

O problema de alocação de salas, conhecido como ”ensalamento”, visa distribuir as aulas de maneira a ocupar o menor número possível de salas, considerando as seguintes condições:

- Duas aulas não podem ocupar a mesma sala simultaneamente;
- Não há necessidade de intervalo de tempo entre duas aulas, isto é, uma aula pode iniciar imediatamente após o término de outra.

Foram implementadas duas possíveis soluções, a fim de identificar qual abordagem é mais eficaz para resolver o problema de alocação de salas. Na primeira abordagem, emprega-se a Estratégia Gulosa, escalonamento de tarefas, enquanto na segunda, utiliza-se a estratégia de Transformar para Conquistar (TC). Ambos os algoritmos e as implementações serão detalhadas nas seções subsequentes.

2.1. Estratégia Gulosa

Sendo a primeira solução do problema do ensalamento, a Estratégia Gulosa, chamada de *”Greedy”*, manipula os itens presentes no conjunto de entrada exatamente na mesma posição ao longo de sua execução, buscando distribuí-los da melhor forma possível.

É uma abordagem heurística empregada em problemas de otimização, baseia-se na tomada de decisões locais que parecem ser as melhores em cada etapa. A premissa subjacente é que ao realizar escolhas ótimas localmente, o resultado global também será otimizado. Essa técnica revela-se particularmente eficaz em cenários nos quais uma busca exaustiva por todas as soluções é difícil ou impossível de ser realizada.

Funcionamento da Estratégia Gulosa:

- Escolha Inicial: Inicia-se o processo com uma escolha inicial, que pode ser feita de forma arbitrária ou com base em critérios específicos, dependendo da natureza do problema.
- Critério Ganancioso: Em cada etapa do algoritmo, uma decisão é tomada com base em um critério ganancioso, direcionado a maximizar ou minimizar alguma métrica localmente.
- Atualização da Solução: A solução é ajustada conforme a escolha feita na etapa anterior, e o processo é repetido até que um critério de parada seja satisfeito.
- Não Revisão de Decisões: Uma característica fundamental é que as decisões tomadas nas etapas anteriores não são reavaliadas. O algoritmo parte do pressuposto de que as melhores escolhas locais em cada etapa contribuirão para uma solução globalmente ótima.

Entretanto, é essencial reconhecer que a estratégia gulosa não é infalível. Em situações em que o problema apresenta características complexas ou envolve um grande número de variáveis, ela pode conduzir a soluções subótimas. Portanto, ao aplicar essa estratégia, é crucial avaliar cuidadosamente a adequação do método em relação às nuances específicas do problema em questão.

2.2. Transformar para Conquistar

A segunda solução do problema, a estratégia Transformar para Conquistar (TC), chamada de *"TC Greedy"*, combina uma estratégia em conjunto com a abordagem gulosa escalonamento de tarefas. Nessa abordagem, o conjunto de itens presentes no vetor de entrada é ordenado conforme o horário de início da aula, permitindo uma distribuição mais eficiente dos horários disponíveis.

Para a ordenação das entradas foi escolhido o algoritmo de ordenação *QuickSort* devido à sua reputação de ser o mais eficiente entre todos os algoritmos de ordenação. Sua capacidade de dividir o conjunto de dados em subconjuntos menores e ordená-los de forma rápida e eficaz o torna uma escolha ideal para lidar com grandes volumes de informações, proporcionando resultados precisos e tempos de execução reduzidos.

A estratégia TC é uma abordagem frequentemente utilizada em problemas de otimização, especialmente em contextos de escalonamento e alocação de recursos. Essa estratégia considera o momento em que uma determinada atividade ou evento é concluído, e pode ser valiosa ao combinar com algoritmos gulosos.

3. Implementação

Para o desenvolvimento dos códigos foi utilizada a linguagem de programação C, versão 8.1.0 do GCC (compilador C). A realização dos testes utilizou o ambiente de desenvolvimento integrado Visual Studio Code em uma máquina local, com as seguintes especificações:

- Máquina: MacBook Pro.
- SO: macOS Sonoma versão 14.3.
- RAM: 16 GB.
- Disco Rígido: Macintosh HD 995 GB.

A fim de avaliar o impacto no tempo de processamento das estratégias, foi feita uma leitura de arquivos de texto que foram fornecidos via Google Drive. Neles temos arquivos com entradas de tamanho: 10, 25, 50, 100, 200, 300, 500, 750, 1000, 1500, 2000, 2500, 5000, 7500, 10000, 15000, 20000, 30000, 50000, 75000, 100000, 150000, 250000, 350000, 500000, 750000 e 1000000. Todos os arquivos possuem o horário das aulas, sendo a primeira fileira a de início e a segunda de término, como mostra a Figura 1.

Figure 1. Exemplo de Entrada

Para a avaliação prática, serão realizadas 5 repetições para cada estratégia. O comparativo entre elas será feito através do cálculo do tempo médio das 5 execuções. Abaixo estão os dois formatos de implementação utilizados, tanto o *Greedy* (entradas não ordenadas) na Seção 3.1 quanto o *TC Greedy* (entradas ordenadas) na Seção 3.2. Esse método permitirá uma análise robusta e comparativa do desempenho das estratégias em diferentes condições.

3.1. Greddy

Este código implementa um algoritmo guloso para alocar salas de aula para a realização de aulas, utilizando os tempos de início e término das mesmas como critério. Entretanto, é importante destacar que este algoritmo não segue uma ordenação específica. Na busca pela próxima aula a ser alocada, ele seleciona aquela com o menor tempo de início, sem considerar a ordem das aulas no array.

Abaixo é apresentada a implementação da estratégia gulosa utilizada, junto com a sua complexidade:

```
while (n > 0) { // O(n)
    int min_start = __INT_MAX__; // O(1)
    int min_index = -1; // O(1)

    for (int j = 0; j < n; j++) { // O(n)
        if (start_times[j] < min_start) { // O(1)
            min_start = start_times[j]; // O(1)
            min_index = j; // O(1)
        }
    }

    int sala_sequencial = -1; // O(1)
    for (int j = 0; j < *num_salas; j++) { // O(n)
        if (salas_ocupadas[j] <= start_times[min_index]) { // O(1)
            sala_sequencial = j; // O(1)
            break;
        }
    }

    if (sala_sequencial == -1) { // O(1)
        salas_ocupadas[(*num_salas)++] = end_times[min_index]; // O(1)
        sala_sequencial = *num_salas - 1; // O(1)
    } else {
        salas_ocupadas[sala_sequencial] = end_times[min_index]; // O(1)
    }

    for (int i = min_index; i < n - 1; i++) { // O(n)
        start_times[i] = start_times[i + 1]; // O(1)
        end_times[i] = end_times[i + 1]; // O(1)
    }

    n--; // O(1)
}
```

Note que a complexidade de tempo total do algoritmo é dominada pelo loop aninhado, resultando em uma complexidade de tempo de $O(n^2)$.

3.2. TC Greedy

O código implementa o *QuickSort* para ordenar os tempos de início e término das aulas, proporcionando uma base organizada para a distribuição subsequente. Ele é aplicado como uma etapa preliminar, considerando tanto os tempos de início quanto os de término, evitando conflitos em casos de horários de início iguais. Posteriormente, o algoritmo guloso aloca as aulas nas salas de aula de maneira a minimizar o número total de salas utilizadas, garantindo assim uma distribuição otimizada dos horários disponíveis.

Segue a implementação da estratégia gulosa utilizada, acompanhada de sua respectiva análise de complexidade:

```
for (int i = 0; i < n; i++) { // O(n)
    int sala_disponivel = -1; // O(1)

    // Procura por sala dispon vel
    for (int j = 0; j < *num_salas; j++) { // O(n)
        if (salas_ocupadas[j] <= start_times[i]) { // O(1)
            sala_disponivel = j; // O(1)
            break;
        }
    }

    // Se n o houver sala dispon vel, cria uma nova sala
    if (sala_disponivel == -1) { // O(1)
        salas_ocupadas[(*num_salas)++] = end_times[i]; // O(1)
        sala_disponivel = *num_salas - 1; // O(1)
    } else {
        salas_ocupadas[sala_disponivel] = end_times[i]; // O(1)
    }
}
```

Como a complexidade para a estratégia gulosa é de $O(n^2)$, temos que adiciona-lá com junto a complexidade do algoritmo de ordenação, abaixo é apresentado a sua implementação, seguido da complexidade:

```
if (left < right) { // O(1)
    int pivotIndex = left + (right - left) / 2; // O(1)
    int pivotValue = arr1[pivotIndex]; // O(1)
    int i = left, j = right; // O(1)

    while (i <= j) { // O(n)
        while (arr1[i] < pivotValue || (arr1[i] == pivotValue && arr2[i
    ] < arr2[pivotIndex])) // O(1)
            i++; // O(1)
        while (arr1[j] > pivotValue || (arr1[j] == pivotValue && arr2[j
    ] > arr2[pivotIndex])) // O(1)
            j--; // O(1)

        if (i <= j) { // O(1)
            trocar(&arr1[i], &arr1[j]); // O(1)
            trocar(&arr2[i], &arr2[j]); // O(1)
            i++; // O(1)
            j--; // O(1)
        }
    }

    quick_sort(arr1, arr2, left, j); // O(n log n)
    quick_sort(arr1, arr2, i, right); // O(n log n)
}
```

Feito o cálculo da análise assintótica do algoritmo *QuickSort* temos uma complexidade de $O(n \log_2 n)$. Para obter a complexidade total, deve ser feita a seguinte analogia: a complexidade de tempo $O(n \log_2 n) + O(n^2)$ simplifica para: $O(n \log_2 n)$. Isso ocorre porque, em notação Big O, nós mantemos o termo que cresce mais rapidamente. Nesse

caso: $n \log n$ cresce mais rapidamente do que n quando n se aproxima do infinito. Portanto, a complexidade de tempo total é $O(n \log_2 n)$.

4. Análises e Resultados

Em geral, algoritmos com complexidade $O(n \log_2 n)$ são mais rápidos do que algoritmos com complexidade $O(n^2)$ para entradas grandes. Isso ocorre porque a complexidade $O(n^2)$ cresce mais rapidamente à medida que o tamanho da entrada aumenta, enquanto a complexidade $O(n \log_2 n)$ cresce de forma mais moderada. Portanto, em situações onde o tamanho da entrada é grande, é provável que um algoritmo com complexidade $O(n \log_2 n)$ seja mais rápido do que um algoritmo com complexidade $O(n^2)$.

Para validar a precisão da sentença matemática, serão fornecidos os tempos médios, em segundos, de execução, resultantes de 5 execuções, para cada um dos métodos em cada conjunto de dados. Informações adicionais sobre os resultados dos algoritmos podem ser encontradas na Seção 6.

4.1. Execução Greedy

Ao executarmos o algoritmo não ordenado (Greedy), foi observado um aumento significativo no tempo médio de execução à medida que o número da aulas aumentam. Isso indica que, em geral, as aulas com números mais altos exigem mais tempo para serem processadas. A Figura 2 apresenta o tempo para cada uma das entradas.

Greedy	
	Tempo de execução médio (segundos)
Aula10	0,0000028
Aula25	0,0000136
Aula50	0,0000158
Aula100	0,0000432
Aula200	0,0001528
Aula300	0,0003176
Aula500	0,0007702
Aula750	0,0014408
Aula1000	0,0021002
Aula1500	0,0037886
Aula2000	0,0055506
Aula2500	0,0080142
Aula5000	0,0313718
Aula7500	0,0690308
Aula10000	0,121577
Aula15000	0,2730828
Aula20000	0,483682
Aula30000	1,091797
Aula50000	3,130329
Aula75000	7,073424
Aula100000	12,531227
Aula150000	28,215415
Aula250000	78,084403
Aula350000	153,225922
Aula500000	312,9365260
Aula750000	704,4808146
Aula1000000	1.343,3544442

Figure 2. Quadro de tempo de execução média

A partir da aula 100, observa-se um aumento considerável no tempo de execução, o que pode ser atribuído a um aumento na complexidade do conteúdo ou na quantidade de dados processados. Entretanto, a partir da aula 5000, nota-se um aumento exponencial no tempo de execução, o que sugere a presença de um gargalo no sistema ou um problema de otimização mais significativo.

Ao analisar a versão gráfica na Figura 3, é perceptível que o crescimento não é totalmente linear. Observam-se ”pontas” nas curvas da linha, devido ao tempo de execução ser menor para determinadas entradas. Essas ocorrências são atribuídas a fatores como o escalonamento de processos e o desempenho da cache, os quais podem oferecer uma pequena otimização, embora não significativa.

Figure 3. Gráfico de tempo de execução média

É crucial destacar que a entrada de 1000000 valores consiste unicamente em números negativos. Devido à ausência de tempo negativo, a equipe optou por considerar o valor absoluto dos dados lidos. Como resultado, o tempo de execução torna-se consideravelmente longo, uma vez que o algoritmo percorre o arquivo n vezes. No entanto, a saída do código mostra apenas uma sala alocada. Isso ocorre devido ao tratamento do valor absoluto dos tempos, permitindo que o zero, um número não negativo, seja alocado em uma sala de aula.

4.2. Execução TC Greedy

Ao executar o algorítimo ordenado (TC Greedy), foi observado um aumento exponencial no tempo de execução à medida que o tamanho da entrada aumenta. Isso indica

que o algoritmo em questão tem uma complexidade de tempo exponencial. A Figura 4 apresenta o tempo para cada uma das entradas.

TC Greedy	
	Tempo de execução média (segundos)
Aula10	0,0000056
Aula25	0,0000082
Aula50	0,0000136
Aula100	0,000025
Aula200	0,0000642
Aula300	0,0001298
Aula500	0,0002546
Aula750	0,000486
Aula1000	0,0007094
Aula1500	0,0012186
Aula2000	0,0018704
Aula2500	0,0023926
Aula5000	0,0073086
Aula7500	0,0147008
Aula10000	0,0250288
Aula15000	0,0551948
Aula20000	0,0958058
Aula30000	0,2087928
Aula50000	0,6577072
Aula75000	1,427966
Aula100000	2,5182298
Aula150000	5,5099982
Aula250000	14,8085982
Aula350000	28,5331198
Aula500000	57,6873984
Aula750000	129,147552
Aula1000000	0,0836758

Figure 4. Quadro de tempo de execução média

Diferente do método Greedy, somente a partir da aula 75000, nota-se um aumento exponencial no tempo de execução, o que sugere a presença de um gargalo no sistema ou um problema de otimização mais significativo.

Ao observar o gráfico na Figura 5, também é evidente que o crescimento não é completamente linear. A presença de ”pontas” nas curvas ainda é notável. Esses picos podem ser atribuídos a diversos fatores, incluindo o escalonamento de processos, o desempenho da cache e até mesmo a disposição dos dados, especialmente porque agora estão ordenados. Embora possam fornecer alguma otimização, é novamente uma melhoria insignificante.

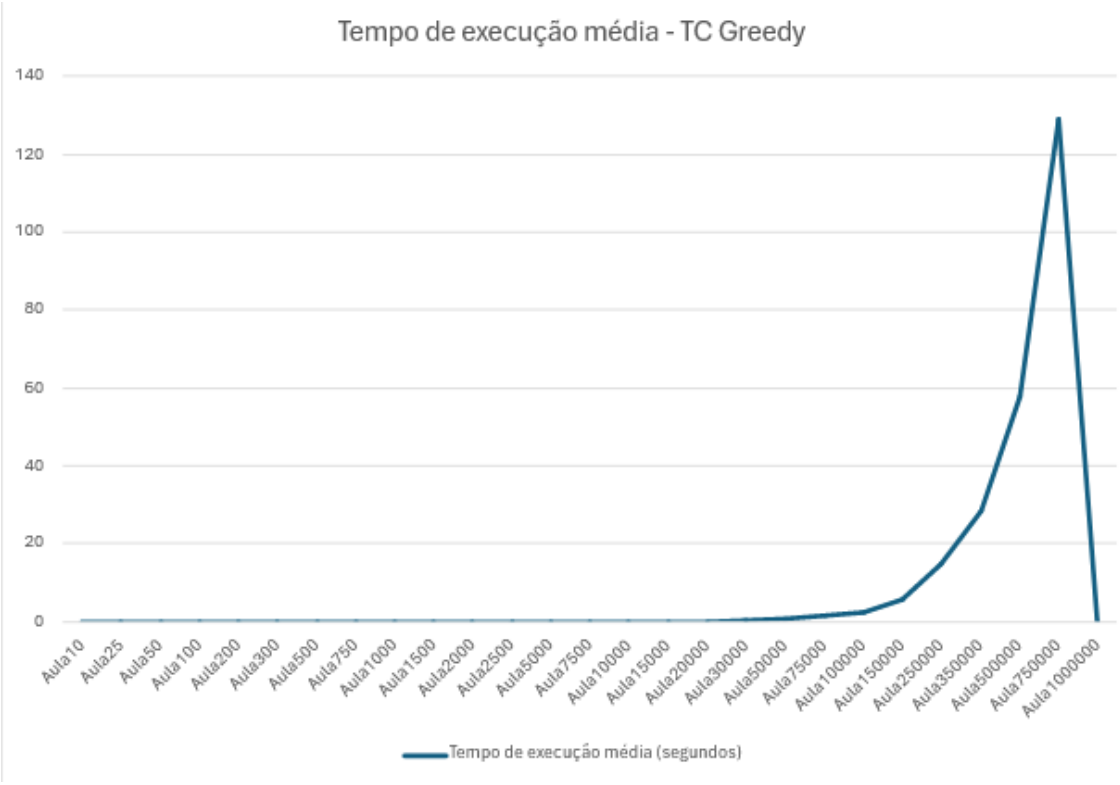


Figure 5. Gráfico de tempo de execução média

Uma observação notável foi a queda abrupta no tempo de execução ao analisar o arquivo de tamanho 1000000. Como mencionado anteriormente, a equipe decidiu não considerar tempos negativos no projeto. Portanto, ao aplicar o método de ordenação no arquivo, o tempo de processamento torna-se significativamente menor. Isso ocorre porque basta acessar a primeira posição do vetor, que estará com valor zero devido ao tratamento de valores absolutos, e alocá-lo imediatamente em uma sala de aula. Essa abordagem simplificada resulta em uma redução drástica no tempo de execução para arquivos com muitos valores negativos.

5. Conclusões

Com base nos resultados obtidos tanto nos cálculos assintóticos quanto nos testes de desempenho, concluímos que, embora o método Greedy com ordenação (TC Greedy) tenha demandado tempo adicional durante o processo de ordenação dos vetores, ele demonstrou uma vantagem em certos aspectos quando comparado ao método Greedy tradicional. O método Greedy sem ordenação apresentou tempos muito significativos em arquivos com entradas menores; no entanto, à medida que o tamanho do arquivo aumenta, seu desempenho é superado pelo método com ordenação. Essa mudança de cenário sugere que a abordagem com ordenação oferece uma melhoria substancial na eficiência do algoritmo em tamanhos de entrada mais amplos.

Além disso, vale ressaltar que a escolha entre os métodos Greedy com e sem ordenação depende das características específicas do problema e das restrições impostas. Enquanto o Greedy sem ordenação pode ser mais eficaz em situações onde a ordem das tarefas não é relevante ou quando os conjuntos de dados são pequenos, o Greedy com ordenação se

destaca em problemas que exigem uma alocação eficiente em larga escala, especialmente quando a ordenação prévia dos dados pode otimizar o processo de alocação. Portanto, ao selecionar o método mais apropriado, é crucial considerar a complexidade do problema, o tamanho dos conjuntos de dados e as limitações de recursos para garantir uma solução eficiente e eficaz.

References

Rocha, A. and Dorini, L. B. (2004). Algoritmos gulosos: definições e aplicações. *Campinas, SP*, 53.

6. Apêndice

Abaixo é apresentada a Planilha de Resultados[1], com os resultados obtidos nos dois métodos em todas as 5 execuções com todas as entradas.

Código sem Ordenação

Arquivo 10					
Execução	1	2	3	4	5
Tempo de Greedy	0,000007	0,000003	0,000001	0,000002	0,000001
Salas Criadas	7				

Arquivo 25					
Execução	1	2	3	4	5
Tempo de Greedy	0.000009	0.000006	0.000007	0.000006	0.000004
Salas Criadas	15				

Arquivo 50					
Execução	1	2	3	4	5
Tempo de Greedy	0.000019	0.000018	0.000013	0.000015	0.000014
Salas Criadas	33				

Arquivo 100					
Execução	1	2	3	4	5
Tempo de Greedy	0,000046	0,000044	0,000043	0,000042	0,000041
Salas Criadas	53				

Arquivo 200					
Execução	1	2	3	4	5
Tempo de Greedy	0.000158	0.000152	0.000151	0.000152	0.000151
Salas Criadas	116				

Arquivo 300					
Execução	1	2	3	4	5
Tempo de Greedy	0.000318	0.000319	0.000319	0.000316	0.000316
Salas Criadas	154				

Arquivo 500					
Execução	1	2	3	4	5
Tempo de Greedy	0.000840	0.000839	0.000745	0.000714	0.000713
Salas Criadas	259				

Arquivo 750					
Execução	1	2	3	4	5
Tempo de Greedy	0.001575	0.001529	0.001367	0.001367	0.001366
Salas Criadas	404				

Arquivo 1000					
Execução	1	2	3	4	5
Tempo de Greedy	0.002339	0.002079	0.002079	0.002080	0.001924
Salas Criadas	481				

Arquivo 1500					
Execução	1	2	3	4	5
Tempo de Greedy	0.004168	0.003887	0.003757	0.003693	0.003438
Salas Criadas	735				

Arquivo 2000					
Execução	1	2	3	4	5
Tempo de Greedy	0.006075	0.005608	0.005492	0.005357	0.005221
Salas Criadas	999				

Arquivo 2500					
Execução	1	2	3	4	5
Tempo de Greedy	0.008026	0.007941	0.008001	0.008051	0.008052
Salas Criadas	1242				

Arquivo 5000					
Execução	1	2	3	4	5
Tempo de Greedy	0.031594	0.031225	0.031333	0.031250	0.031457
Salas Criadas	2347				

Arquivo 7500					
Execução	1	2	3	4	5
Tempo de Greedy	0.069352	0.068933	0.068979	0.068873	0.069017
Salas Criadas	3425				

Arquivo 10000					
Execução	1	2	3	4	5
Tempo de Greedy	0.121328	0.121292	0.121265	0.122749	0.121251
Salas Criadas	4405				

Arquivo 15000					
Execução	1	2	3	4	5
Tempo de Greedy	0.273013	0.273129	0.273084	0.273086	0.273102
Salas Criadas	6597				

Arquivo 20000					
Execução	1	2	3	4	5
Tempo de Greedy	0.483174	0.483200	0.483499	0.484526	0.484011
Salas Criadas	9606				

Arquivo 30000					
Execução	1	2	3	4	5
Tempo de Greedy	1,088689	1,106871	1,089065	1,088207	1,086154
Salas Criadas	12864				

Arquivo 50000					
Execução	1	2	3	4	5
Tempo de Greedy	3,132283	3,143059	3,136035	3,119102	3,121168
Salas Criadas	25090				

Arquivo 75000					
Execução	1	2	3	4	5
Tempo de Greedy	7,109820	7,082978	7,062022	7,072190	7,040111
Salas Criadas	37336				

Arquivo 100000					
Execução	1	2	3	4	5
Tempo de Greedy	12,531124	12,523425	12,531162	12,510119	12,560306
Salas Criadas	49979				

Arquivo 150000					
Execução	1	2	3	4	5
Tempo de Greedy	28,337819	28,309335	28,164962	28,128710	28,136249
Salas Criadas	74868				

Arquivo 250000					
Execução	1	2	3	4	5
Tempo de Greedy	78,069580	78,077717	78,115640	78,092723	78,066354
Salas Criadas	124877				

Arquivo 350000					
Execução	1	2	3	4	5
Tempo de Greedy	153,197741	153,202187	153,216127	153,186977	153,326578
Salas Criadas	175219				

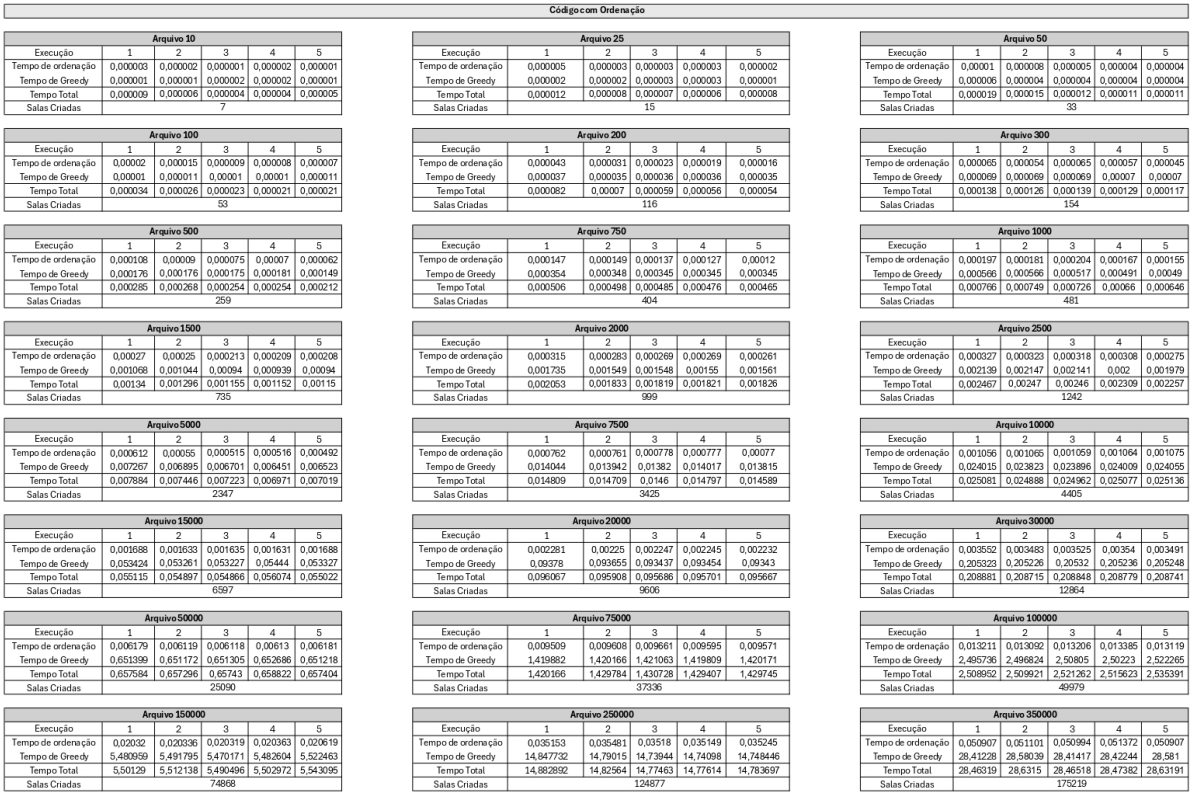
Arquivo 500000					
Execução	1	2	3	4	5
Tempo de Greedy	313,360863	312,783071	312,840740	312,857870	312,840086
Salas Criadas	250175				

Arquivo 750000					
Execução	1	2	3	4	5
Tempo de Greedy	704,347613	704,595426	704,374974	########	704,555373
Salas Criadas	375352				

Arquivo 1000000					
Execução	1	2	3	4	5
Tempo de Greedy	1.342,939315	1.344,312427	1.343,023802	1.342,823686	1.343,672991
Salas Criadas	1				

Figure 6. Resultados Greedy

[1]Disponível em: `https://encurtador.com.br/jopqH`



Código com Ordenação

Arquivo 10					
Execução	1	2	3	4	5
Tempo de ordenação	0,000003	0,000002	0,000001	0,000002	0,000001
Tempo de Greedy	0,000001	0,000001	0,000002	0,000002	0,000001
Tempo Total	0,000009	0,000006	0,000004	0,000004	0,000005
Salas Criadas	7				

Arquivo 25					
Execução	1	2	3	4	5
Tempo de ordenação	0,000005	0,000003	0,000003	0,000003	0,000002
Tempo de Greedy	0,000002	0,000002	0,000003	0,000003	0,000001
Tempo Total	0,000012	0,000008	0,000007	0,000006	0,000008
Salas Criadas	15				

Arquivo 50					
Execução	1	2	3	4	5
Tempo de ordenação	0,00001	0,000008	0,000005	0,000004	0,000004
Tempo de Greedy	0,000006	0,000004	0,000004	0,000004	0,000004
Tempo Total	0,000019	0,000015	0,000012	0,000011	0,000011
Salas Criadas	33				

Arquivo 100					
Execução	1	2	3	4	5
Tempo de ordenação	0,00002	0,000015	0,000009	0,000008	0,000007
Tempo de Greedy	0,00001	0,000011	0,00001	0,00001	0,000011
Tempo Total	0,000034	0,000026	0,000023	0,000021	0,000021
Salas Criadas	53				

Arquivo 200					
Execução	1	2	3	4	5
Tempo de ordenação	0,000043	0,000031	0,000023	0,000019	0,000016
Tempo de Greedy	0,000037	0,000035	0,000036	0,000036	0,000035
Tempo Total	0,000082	0,00007	0,000059	0,000056	0,000054
Salas Criadas	116				

Arquivo 300					
Execução	1	2	3	4	5
Tempo de ordenação	0,000065	0,000054	0,000065	0,000057	0,000045
Tempo de Greedy	0,000069	0,000069	0,000069	0,00007	0,00007
Tempo Total	0,000138	0,000126	0,000139	0,000129	0,000117
Salas Criadas	154				

Arquivo 500					
Execução	1	2	3	4	5
Tempo de ordenação	0,000108	0,00009	0,000075	0,00007	0,000062
Tempo de Greedy	0,000176	0,000176	0,000175	0,000181	0,000149
Tempo Total	0,000285	0,000268	0,000254	0,000254	0,000212
Salas Criadas	259				

Arquivo 750					
Execução	1	2	3	4	5
Tempo de ordenação	0,000147	0,000149	0,000137	0,000127	0,00012
Tempo de Greedy	0,000354	0,000348	0,000345	0,000345	0,000345
Tempo Total	0,000506	0,000498	0,000485	0,000476	0,000465
Salas Criadas	404				

Arquivo 1000					
Execução	1	2	3	4	5
Tempo de ordenação	0,000197	0,000181	0,000204	0,000167	0,000155
Tempo de Greedy	0,000566	0,000566	0,000517	0,000491	0,00049
Tempo Total	0,000766	0,000749	0,000726	0,00066	0,000646
Salas Criadas	481				

Arquivo 1500					
Execução	1	2	3	4	5
Tempo de ordenação	0,00027	0,00025	0,000213	0,000209	0,000208
Tempo de Greedy	0,001068	0,001044	0,00094	0,000939	0,00094
Tempo Total	0,00134	0,001296	0,001155	0,001152	0,00115
Salas Criadas	735				

Arquivo 2000					
Execução	1	2	3	4	5
Tempo de ordenação	0,000315	0,000283	0,000269	0,000269	0,000261
Tempo de Greedy	0,001735	0,001549	0,001548	0,00155	0,001561
Tempo Total	0,002053	0,001833	0,001819	0,001821	0,001826
Salas Criadas	999				

Arquivo 2500					
Execução	1	2	3	4	5
Tempo de ordenação	0,000327	0,000323	0,000318	0,000308	0,000275
Tempo de Greedy	0,002139	0,002147	0,002141	0,002	0,001979
Tempo Total	0,002467	0,00247	0,00246	0,002309	0,002257
Salas Criadas	1242				

Arquivo 5000					
Execução	1	2	3	4	5
Tempo de ordenação	0,000612	0,00055	0,000515	0,000516	0,000492
Tempo de Greedy	0,007267	0,006895	0,006701	0,006451	0,006523
Tempo Total	0,007884	0,007446	0,007223	0,006971	0,007019
Salas Criadas	2347				

Arquivo 7500					
Execução	1	2	3	4	5
Tempo de ordenação	0,000762	0,000761	0,000778	0,000777	0,00077
Tempo de Greedy	0,014044	0,013942	0,01382	0,014017	0,013815
Tempo Total	0,014809	0,014709	0,0146	0,014797	0,014589
Salas Criadas	3425				

Arquivo 10000					
Execução	1	2	3	4	5
Tempo de ordenação	0,001056	0,001065	0,001059	0,001064	0,001075
Tempo de Greedy	0,024015	0,023823	0,023896	0,024009	0,024055
Tempo Total	0,025081	0,024888	0,024962	0,025077	0,025136
Salas Criadas	4405				

Arquivo 15000					
Execução	1	2	3	4	5
Tempo de ordenação	0,001688	0,001633	0,001635	0,001631	0,001688
Tempo de Greedy	0,053424	0,053261	0,053227	0,05444	0,053327
Tempo Total	0,055115	0,054897	0,054866	0,056074	0,055022
Salas Criadas	6597				

Arquivo 20000					
Execução	1	2	3	4	5
Tempo de ordenação	0,002281	0,00225	0,002247	0,002245	0,002232
Tempo de Greedy	0,09378	0,093655	0,093437	0,093454	0,09343
Tempo Total	0,096067	0,095908	0,095686	0,095701	0,095667
Salas Criadas	9606				

Arquivo 30000					
Execução	1	2	3	4	5
Tempo de ordenação	0,003552	0,003483	0,003525	0,00354	0,003491
Tempo de Greedy	0,205323	0,205226	0,20532	0,205236	0,205248
Tempo Total	0,208881	0,208715	0,208848	0,208779	0,208741
Salas Criadas	12864				

Arquivo 50000					
Execução	1	2	3	4	5
Tempo de ordenação	0,006179	0,006119	0,006118	0,00613	0,006181
Tempo de Greedy	0,651399	0,651172	0,651305	0,652686	0,652188
Tempo Total	0,657584	0,657296	0,65743	0,658822	0,657404
Salas Criadas	25090				

Arquivo 75000					
Execução	1	2	3	4	5
Tempo de ordenação	0,009509	0,009608	0,009661	0,009595	0,009571
Tempo de Greedy	1,419882	1,420166	1,421063	1,419809	1,420171
Tempo Total	1,420166	1,429784	1,430728	1,429407	1,429745
Salas Criadas	37336				

Arquivo 100000					
Execução	1	2	3	4	5
Tempo de ordenação	0,013211	0,013092	0,013206	0,013385	0,013119
Tempo de Greedy	2,495736	2,498624	2,50805	2,50223	2,522265
Tempo Total	2,508952	2,509921	2,521262	2,515623	2,535391
Salas Criadas	49979				

Arquivo 150000					
Execução	1	2	3	4	5
Tempo de ordenação	0,02032	0,020336	0,020319	0,020363	0,020619
Tempo de Greedy	5,480959	5,491795	5,470171	5,482604	5,522463
Tempo Total	5,50129	5,512138	5,490496	5,502972	5,543095
Salas Criadas	74868				

Arquivo 250000					
Execução	1	2	3	4	5
Tempo de ordenação	0,035153	0,035481	0,03518	0,035149	0,035245
Tempo de Greedy	14,847732	14,79015	14,73944	14,74098	14,748446
Tempo Total	14,882892	14,82564	14,77463	14,77614	14,783697
Salas Criadas	124877				

Arquivo 350000					
Execução	1	2	3	4	5
Tempo de ordenação	0,050907	0,051101	0,050994	0,051372	0,050907
Tempo de Greedy	28,41228	28,58039	28,41417	28,42244	28,581
Tempo Total	28,46319	28,6315	28,46518	28,47382	28,63191
Salas Criadas	175219				

Figure 7. Resultados TC Greedy

Arquivo 500000					
Execução	1	2	3	4	5
Tempo de ordenação	0,075534	0,075071	0,074086	0,074044	0,074391
Tempo de Greedy	57,675538	57,82262	57,6683	57,44359	57,45377
Tempo Total	57,75108	57,8977	57,7424	57,51765	57,52817
Salas Criadas	250175				

Arquivo 750000					
Execução	1	2	3	4	5
Tempo de ordenação	0,114524	0,116016	0,115251	0,118038	0,115761
Tempo de Greedy	128,507862	128,5761	130,3781	129,2888	128,40727
Tempo Total	128,622395	128,6921	130,4934	129,4068	128,523036
Salas Criadas	375352				

Arquivo 1000000					
Execução	1	2	3	4	5
Tempo de ordenação	0,080217	0,080691	0,08054	0,081197	0,080851
Tempo de Greedy	0,002932	0,002909	0,003104	0,002897	0,003006
Tempo Total	0,083154	0,08361	0,083651	0,0841	0,083864
Salas Criadas	1				

Figure 8. Resultados TC Greedy