1º Trabalho de Projeto e Análise de Algoritmos (PAA)
Métodos de Ordenação
Estratégia Gulosa
Programação Dinâmica

André Luiz Brun[1]

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

Resumo. *Este documento consiste na especificação formal do primeiro trabalho da disciplina de Projeto e Análise de Algoritmos (Csc2152) para o ano letivo de 2024. Aqui são apresentadas as atividades a serem desenvolvidas e como cada processo deverá ser realizado. Além disso, o documento contém as informações sobre a formação das equipes, o objeto de trabalho de cada uma, as datas de entrega e apresentação dos relatórios, bem como critérios de avaliação.*

Introdução

O objetivo do primeiro trabalho da disciplina consiste em comparar o comportamento empírico, principalmente em termos cronológicos, dos métodos de ordenação clássicos frente a alguns cenários e condições variadas. O objetivo é compreender quando e por que motivo tais algoritmos desempenham melhor ou pior em cada contexto.

1. Gabriel Alves Mazzuco
Marco Antonio Damo
Maria Eduarda Crema Carlos

1.1. Objetivo

Aplicar e corroborar conceitos adquiridos sobre estratégias de projeto do grupo Transformar para Conquistar (TC) e Estratégia Gulosa. Para tanto, deverão ser implementadas duas soluções para o problema da mochila binária.

Na primeira solução, que chamaremos de “Clássica”, os itens presentes no conjunto de entrada permanecem na mesma posição ao longo da execução do algoritmo guloso, que busca encher a mochila da melhor forma possível.

Já para a segunda solução “Mochila TC”, que usa uma estratégia TC, o conjunto de itens presentes no vetor de entrada deverá ser ordenado do melhor para o pior, de forma que no início do conjunto fiquem aqueles com o melhor custo/benefício. O processo de seleção segue então a mesma estrutura gulosa da primeira abordagem.

1.2. Critérios de avaliação

Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

1.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis no link a seguir: Link Grupo 1;
- Os arquivos de entrada têm o formato especificado na Figura 1. A primeira linha contém a capacidade da mochila. Não segunda linha são apresentados n valores correspondentes ao benefício de cada um dos n itens presentes na mochila. Por fim, na linha três são apresentados os custos dos n produtos.

105	Capacidade da Mochila
3 42 5 48 42 13 3 20 12 37	Benefícios dos Itens
2 35 13 29 9 25 2 14 4 17	Custos dos Itens

Figura 1. Estrutura dos arquivos de entrada para o problema da mochila

2. Fabrício Cordeiro Marcos
Fernando Schumaker Fiedler
Marlon Pereira

2.1. Objetivo

Aplicar e corroborar conceitos adquiridos sobre estratégias de projeto do grupo Transformar para Conquistar (TC) e Estratégia Gulosa. Para tanto, deverão ser implementadas duas soluções para o problema ensalamento.

A ideia é distribuir as aulas de forma a ocupar o menor número possível de salas, para tanto considerem que:
• Duas aulas não podem ocupar a mesma sala no mesmo momento;
• Não há necessidade de intervalo de tempo entre duas aulas, ou seja, no momento em que uma aula termina, outra pode iniciar imediatamente;

Na primeira solução, que chamaremos de "Greedy", os itens presentes no conjunto de entrada permanecem na mesma posição ao longo da execução do algoritmo guloso, que busca distribuir as aulas da melhor forma possível.

Já para a segunda solução, "TCGreedy", que combina uma estratégia TC com a abordagem Gulosa, o conjunto de itens presentes no vetor de entrada deverá ser ordenado de forma crescente do conforme o horário de início do aula e só então a estratégia gulosa será aplicada para distribuir as aulas.

2.2. Critérios de avaliação

Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

2.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis no link a seguir: Link Grupo 2;
- Os arquivos de entrada têm o formato especificado na Figura 2. Na primeira linha constam os n momentos de início das aulas, enquanto a segunda linha contém os respectivos n momentos de conclusão das aulas.

Figura 2. Estrutura dos arquivos de entrada para o problema da distribuição de aulas

3. Eduardo Pimentel dos Santos
Fabio Novack da Silva
Gustavo Magalhães Faino

3.1. Objetivo

Aplicar e corroborar conceitos adquiridos sobre estratégias de projeto que usam Programação Dinâmica (PD) e Estratégia Gulosa. Para tanto, deverão ser implementadas duas soluções para o problema da mochila binária.

Na primeira solução, que chamaremos de "Greedy", os itens presentes no conjunto de entrada são ordenados do melhor para o pior e então executa-se o algoritmo guloso, que busca encher a mochila da melhor forma possível.

Já para a segunda solução, "Dynamic", que usa uma estratégia PD, deverá fazer uso de tabelas para encontrar a melhor solução possível. A ideia é que a estratégia encontre a melhor solução pontual e, em seguida, possa aplicá-la às soluções seguintes.

3.2. Critérios de avaliação

Para realizar a comparação dos métodos deverão ser feitas três análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Na terceira avaliação, também prática, deverão ser comparados os **benefícios obtidos** pelas duas estratégias para cada uma das entradas;
4. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

3.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis no link a seguir: Link Grupo 3;
- Os arquivos de entrada têm o formato especificado na Figura 1. A primeira linha contém a capacidade da mochila. Não segunda linha são apresentados n valores correspondentes ao benefício de cada um dos n itens presentes na mochila. Por fim, na linha três são apresentados os custos dos n produtos.

4. David Antonio Brocardo
Gabriel Santos da Silva

4.1. Objetivo

Aplicar e corroborar conceitos adquiridos com relação ao método de ordenação MergeSort, bem como elucidar se a passagem por referência, trabalhando com um único vetor em memória é mais vantajosa em relação a criar uma cópia do vetor a cada chamada recursiva do algoritmo.

4.2. Tarefa

Implementar as seguintes estratégias de ordenação:

1. Abordagem clássica do MergeSort que aloca um vetor temporário a cada chamada recursiva (solução vista em aula);
2. Abordagem alternativa do MergeSort que, ao invés de alocar um novo vetor a cada chamada, simplesmente sobrepõe o vetor de entrada (que foi passado por referência).

Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

4.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar a estratégia recursiva para os dois casos.
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis nos links a seguir:
  Aleatórios
  Decrescentes
  Crescentes
  Parcialmente ordenados

Devem ser construídos quatro conjuntos de testes, conforme os arquivos disponíveis nos links apresentados acima:

1. As duas estratégias de ordenação trabalhando sobre conjuntos ordenados de forma decrescente;

2. Os dois métodos de ordenação trabalhando sobre conjunto aleatórios;
3. Todos os métodos sobre conjuntos ordenados de forma crescente;
4. Todos os métodos sobre conjuntos parcialmente ordenados de forma crescente.

Além disso, deve ser analisado o comportamento de cada técnica sobre as diferentes entradas conforme descrito a seguir:

1. MergeSort com alocação de novo vetor sobre os quatro tipos de entrada;
2. MergeSort utilizando apenas passagem por referência sobre os quatro tipos de entrada.

5. Gabriel Tadioto Oliveira
Leonardo B. Balan de Oliveira

5.1. Objetivo

Aplicar e corroborar conceitos adquiridos com relação aos métodos de ordenação quadráticos ($O(n^2)$), bem como ilustrar os comportamentos apresentados pelas estratégias em diferentes cenários.

5.2. Tarefa

Implementar as seguintes estratégias de ordenação:

1. BubbleSort;
2. InsertionSort;
3. SelectionSort.

Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

5.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo;

- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis nos links a seguir:
  Aleatórios
  Decrescentes
  Crescentes
  Parcialmente ordenados

Devem ser construídos quatro conjuntos de testes, conforme os arquivos disponíveis nos links apresentados acima:

1. Os três métodos de ordenação trabalhando sobre conjunto aleatórios;
2. Os três métodos de ordenação trabalhando sobre conjuntos ordenados de forma decrescente;
3. Todos os métodos sobre conjuntos ordenados de forma crescente;
4. Todos os métodos sobre conjuntos parcialmente ordenados de forma crescente.

Além disso, deve ser analisado o comportamento de cada técnica sobre as diferentes entradas conforme descrito a seguir:

1. BubbleSort sobre os quatro tipos de entrada;
2. SelectionSort sobre os quatro tipos de entrada;
3. InsertionSort sobre os quatro tipos de entrada.

6. Augusto Barella Dal Pra
Gabriel Rodrigues dos Santos
Gustavo Portela Rautenberg

6.1. Objetivo

Aplicar e corroborar conceitos adquiridos sobre estratégias de projeto do grupo Estratégia Gulosa. Para tanto, deverão ser implementadas duas soluções para o problema ensalamento.

A ideia é distribuir as aulas de forma a ocupar o menor número possível de salas, para tanto considerem que:
• Duas aulas não podem ocupar a mesma sala no mesmo momento;
• Não há necessidade de intervalo de tempo entre duas aulas, ou seja, no momento em que uma aula termina, outra pode iniciar imediatamente;

Na primeira solução, que chamaremos de “Clássica”, os itens presentes no conjunto de entrada permanecem na mesma posição ao longo da execução do algoritmo guloso, que busca distribuir forma possível. Assim, a cada iteração é buscada a aula com menor tempo inicial e atribuída à primeira sala disponível.

Considerando que a solução Clássica pode sobrecarregar as primeiras salas, pois elas são sempre as primeiras a serem testadas, implementem uma solução alternativa que além de aplicar a estratégia gulosa como na primeira solução, ainda faça o balanceamento de carga nas salas, ou seja, a segunda implementação deverá tentar distribuir as aulas nas salas evitando sobrecarregar as primeiras salas.

6.2. Critérios de avaliação

Para realizar a comparação dos métodos deverão ser feitas três análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Na terceira análise, deverão ser comparadas as cargas de aulas destinadas a cada sala, indicando se houve ou não uma melhoria na distribuição;
4. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

6.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo, ou seja, a equipe que definirá o critério de distribuição das aulas em cada sala para tentar manter o balanceamento;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis no link a seguir: Link Grupo 6;
- Os arquivos de entrada têm o formato especificado na Figura 2. Na primeira linha constam os n momentos de início das aulas, enquanto a segunda linha contém os respectivos n momentos de conclusão das aulas.

7. Bruno Stafuzza Maion
Lucca Abbado Neres
Rafael Roberto Hoffmann

7.1. Objetivo

Aplicar e corroborar conceitos adquiridos com relação aos métodos de ordenação de ordem $O(n \log_2 n)$, bem como ilustrar os comportamentos apresentados pelas estratégias em diferentes cenários.

7.2. Tarefa

Implementar as seguintes estratégias de ordenação:

1. MergeSort;
2. QuickSort;
3. HeapSort.

Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

7.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis nos links a seguir:
  Aleatórios
  Decrescentes
  Crescentes
  Parcialmente ordenados

Devem ser construídos quatro conjuntos de testes, conforme os arquivos disponíveis nos links apresentados acima:

1. Os três métodos de ordenação trabalhando sobre conjunto aleatórios;
2. Os três métodos de ordenação trabalhando sobre conjuntos ordenados de forma decrescente;
3. Todos os métodos sobre conjuntos ordenados de forma crescente;
4. Todos os métodos sobre conjuntos parcialmente ordenados de forma crescente.

Além disso, deve ser analisado o comportamento de cada técnica sobre as diferentes entradas conforme descrito a seguir:

1. QuickSort sobre os quatro tipos de entrada;
2. MergeSort sobre os quatro tipos de entrada;
3. HeapSort sobre os quatro tipos de entrada.

8. João Vitor Biederman
Luiz Felipe Fonseca Rosa
Pedro Henrique Ferreira Zoz

8.1. Objetivo

Aplicar e corroborar conceitos adquiridos sobre técnicas de projeto do grupo Estratégia Gulosa. Para tanto, deverão ser implementadas duas soluções para o problema da mochila.

Na primeira solução, que chamaremos de "Bin", os itens presentes no conjunto de entrada não podem ser fracionados, ou seja, trata-se da **Mochila Binária**.

Já para a segunda solução, "Frac", que usa a mesma estratégia da primeira abordagem. No entanto, para esta solução os itens podem ser fracionados (**Mochila Fracionária**), ou seja, um item pode ser colocado parcialmente na mochila caso não caiba inteiramente.

8.2. Critérios de avaliação

Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Na terceira análise deverão ser comparadas as soluções obtidas em termos de **Qualidade**, já que as abordagens podem chegar a benefícios distintos;
4. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

8.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis no link a seguir: Link Grupo 8;
- Os arquivos de entrada têm o formato especificado na Figura 1.

9. Gustavo Macedo
Heloisa Aparecida Alves
Luiz Fernando Becher de Araujo

9.1. Objetivo

Aplicar e corroborar conceitos adquiridos com relação ao método de ordenação Quick-Sort, bem como ilustrar o comportamento apresentado pela estratégia em diferentes cenários.

9.2. Tarefa

Implementar as seguintes estratégias de ordenação:

1. QuickSort (Randômico) em que a escolha do pivô é feita de forma totalmente aleatória;
2. QuickSort (Central) em que a escolha do pivô é feita selecionando-se o elemento na posição central do vetor;
3. QuickSort (Mediana) em que para se escolher o pivô são sorteados três elementos do vetor e o pivô será o elemento central dos três valores sorteados.

Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

9.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis nos links a seguir:
  Aleatórios
  Decrescentes
  Crescentes
  Parcialmente ordenados

Devem ser construídos quatro conjuntos de testes, conforme os arquivos disponíveis nos links apresentados acima:

1. As três estratégias de ordenação trabalhando sobre conjuntos ordenados de forma decrescente;
2. Os três métodos de ordenação trabalhando sobre conjunto aleatórios;
3. Todos os métodos sobre conjuntos ordenados de forma crescente;
4. Todos os métodos sobre conjuntos parcialmente ordenados de forma crescente.

Além disso, deve ser analisado o comportamento de cada técnica sobre as diferentes entradas conforme descrito a seguir:

1. QuickSort Randômico sobre os quatro tipos de entrada;
2. QuickSort Central sobre os quatro tipos de entrada;
3. QuickSort Mediana sobre os quatro tipos de entrada;

10. Eduarda Elger
Ellen Carine Bonafin Marques
Lucas David Tomalack de Souza

10.1. Objetivo

Aplicar e corroborar conceitos adquiridos sobre técnicas de projeto do grupo Programação Dinâmica. Para tanto, deverão ser implementadas duas soluções para o problema de identificação da Substring Comum Mais Longa (Longest Common Subsequence - LCS).

Na primeira solução, que chamaremos de "PD", deverá ser empregada a estratégia de adoção de tabelas para resolver o problema localmente e expandir tal solução às instâncias maiores do mesmo problema (exemplo de Programação Dinâmica visto em aula).

Já para a segunda solução, "Rec", deverá implementar a solução recursiva discutida durante as aulas. Nesta abordagem compara-se o último caractere de cada string e, caso sejam iguais, obtém-se um hit, decrementando-se assim as duas strings. Caso contrário, seleciona-se a maior LCS entre:
• A primeira string inteira versus a segunda com a última posição descartada;
• Segunda string inteira versus a primeira com a última posição descartada.

10.2. Critérios de avaliação

Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

10.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis no link a seguir: Link Grupo 10;
- Os arquivos de entrada (ilustrados na Figura 3) contém duas linhas, cada uma contendo uma string. Os caracteres que compõe as strings estão separados por espaços em branco e não haverá presença de caracteres especiais. **Além disso, ambas são compostas pelo mesmo número de caracteres**.

Figura 3. Ilustração de como os arquivos contendo as duas cadeias de caracteres estarão dispostos

11. Caio Hideki Gomes Shimohiro
Gustavo Alberto Ohse Hanke
Vinícius Visconsini Diniz

11.1. Objetivo

Aplicar e corroborar conceitos adquiridos sobre técnicas de projeto do grupo Estratégia Gulosa. Para tanto, deverão ser implementadas duas soluções para o problema da mochila.

Na primeira solução, que chamaremos de "BinGreedy", os itens presentes no conjunto de entrada não podem ser fracionados, ou seja, trata-se da **Mochila Binária**. Além disso, os itens não podem ser pré ordenados. Para esta solução será empregado o algoritmo guloso visto em sala.

Já para a segunda solução, "FB", deverá ser implementada uma abordagem do tipo Força Bruta. Nela todas as possíveis combinações de itens devem ser testadas para descobrir qual delas trará o maior benefício.

11.2. Critérios de avaliação

Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;

2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

11.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis no link a seguir: Link Grupo 11;
- Os arquivos de entrada têm o formato especificado na Figura 1.

12. Isadora Coelho Araujo
Leonardo Calsavara
Ronaldo Drecksler Farias Pachico

12.1. Objetivo

Aplicar e corroborar conceitos adquiridos com relação ao método de ordenação BucketSort combinado com o algoritmo InsertionSort, bem como ilustrar o comportamento apresentado pela estratégia em diferentes cenários.

12.2. Tarefa

Implementar as seguintes estratégias de ordenação:

1. InsertionSort tradicional;
2. Abordagem alternativa do InsertionSort usando a estratégia de BucketSort, ou seja, o conjunto de dados dos vetores de entrada devem ser particionados em baldes e então ordenados pelo método de InsertionSort. 1. Para tanto, deverão ser testadas três abordagens distintas da técnica implementada, cada uma com uma quantidade diferente de baldes, a saber:
   (a) 10 baldes
   (b) 100 baldes
   (c) 1000 baldes.

   Para realizar a comparação dos métodos deverão ser feitas duas análises:

1. Na primeira, teórica, deverão ser obtidos os polinômios de **custo assintótico** das duas soluções;
2. Na segunda avaliação, prática, deverão ser comparados os **tempos cronológicos** de execução das soluções;
3. Para a avaliação prática, será necessária a execução de 6 repetições para cada estratégia. Das 6 execuções a primeira será descartada. A ideia é que o comparativo seja feito a partir do tempo médio das 5 execuções restantes. Assim, o gráfico de comparação $Tamanho\ vs\ Tempo$ apresentará os valores médios das duas soluções implementadas.

12.3. Como

- A linguagem utilizada no desenvolvimento é de vossa escolha, desde seja uma das seguintes linguagens: C, C++, Python ou Java. Contudo, a mesma linguagem deve ser adotada para ambos os métodos. Além disso, deve-se empregar as mesmas estratégias para a realização do trabalho. Por exemplo, empregar soluções recursivas para todos ou para nenhum deles;
- O mesmo número de threads deve ser usado para todas as estratégias;
- Os pontos de segmentação entre os baldes é de escolha da equipe;
- A forma com que os métodos serão implementados é determinada pelo grupo;
- A entrada dos dados deve ser feita com base nos arquivos texto disponíveis nos links a seguir:
  Aleatórios
  Decrescentes
  Crescentes
  Parcialmente ordenados

Devem ser construídos quatro conjuntos de testes, conforme os arquivos disponíveis nos links apresentados acima:

1. Os quatro métodos de ordenação trabalhando sobre conjunto aleatórios;
2. As quatro estratégias de ordenação trabalhando sobre conjuntos ordenados de forma decrescente;
3. Todos os métodos sobre conjuntos ordenados de forma crescente;
4. Todos os métodos sobre conjuntos parcialmente ordenados de forma crescente.

Além disso, deve ser analisado o comportamento de cada técnica sobre as diferentes entradas conforme descrito a seguir:

1. InsertionSort clássico sobre os quatro tipos de entrada;
2. InsertionSort combinando com 10 baldes sobre os quatro tipos de entrada;
3. InsertionSort combinando com 100 baldes sobre os quatro tipos de entrada;
4. InsertionSort combinando com 1000 baldes sobre os quatro tipos de entrada;

13. Relatório

Deve ser elaborado um relatório técnico em formato pdf contendo:

- o Descrição de como foi realizado o processo empírico de determinação dos custos: cenário de realização dos experimentos e como foram tomadas as métricas exigidas.
  - Detalhar a configuração usada nos testes (Processador, SO, IDE, etc..).
- Gráficos evidenciando o comportamento dos métodos perante todos os cenários considerando o tamanho dos conjuntos de entrada.
- Análise do comportamento dos métodos durante a execução dos testes.
  - Esta análise deve ser feita com bastante critério e ser esclarecedora, apontando razões para os comportamentos observados.
  - **AS figuras e gráficos devem ser invocadas no texto e explicadas.**

O formato do relatório deve ser a formatação presente neste texto. As regras para tal podem ser obtidas no link download. No arquivo disponível pode-se utilizar a formatação em arquivo .doc ou em latex.

14. Código-fonte

Além do relatório citado, cada equipe deverá enviar os códigos fontes construídos para a execução dos experimentos. Ambos arquivos podem ser compactados e enviados como arquivo único.

Não é necessário o envio do projeto compilado, apenas do código fonte base.

Neste código devem constar os comandos utilizados para tomadas de tempo de execução e contagem das variáveis de interesse.

15. Para quando?

O trabalho deverá ser submetido no link disponibilizado na turma de disciplina dentro do ambiente Microsoft Teams até as **23:59 do dia 02/12/2024.**

As apresentações serão realizadas nas aulas dos dias **02** e **04/12/2024**.

Cada grupo terá 15 minutos para apresentar o trabalho realizado, focando na descrição do problema, nos desempenhos obtidos e nos resultados das análises. **A qualidade da apresentação terá influência na nota do trabalho.**

Além da apresentação as equipes serão avaliadas pelo seu comportamento e presença durante as apresentações dos demais grupos.