­**UNIVERSIDADE FEDERAL DE MINAS GERAIS**

**DCC | SISTEMAS DE INFORMAÇÃO | ALGORITMOS I**

**DOCUMENTAÇÃO**

**TRABALHO PRÁTICO 1 | VIAGEM**

**ALGORITMOS GULOSO E DE PROGRAMAÇÃO DINÂMICA**

**Rômulo Rafael da Silva, 2012055308**

**Outubro / 2019**

Sumário

[1. DESCRIÇÃO DO PROBLEMA 3](#_Toc23111153)

[2. DESCRIÇÕES TÉCNICAS 4](#_Toc23111154)

[2.1. HARDWARE 4](#_Toc23111155)

[2.2. SOFTWARE 4](#_Toc23111156)

[2.3. LINGUAGEM DE PROGRAMAÇÃO 4](#_Toc23111157)

[3. ESTRUTURAS DE PROJETO E ALGORITMOS 4](#_Toc23111158)

[3.1. DIRETÓRIO DO PROJETO 4](#_Toc23111159)

[3.2. PARADIGMAS DE PROGRAMAÇÃO 5](#_Toc23111160)

[3.3. PROBLEMAS ALGORÍTMICOS 5](#_Toc23111161)

[3.4. PRINCÍPIOS DE PROJETOS DA SOLUÇÃO PROPOSTA ALGORÍTMICOS 6](#_Toc23111162)

[4. ANÁLISE DE COMPLEXIDADE 7](#_Toc23111163)

[4.1. STRUCT E CLASSES 7](#_Toc23111164)

[4.2. TEMPO 7](#_Toc23111165)

[4.3. ESPAÇO 7](#_Toc23111166)

[4.2. PROVA DE CORRETUDE 8](#_Toc23111167)

[5. AVALIAÇÃO EXPERIMENTAL 9](#_Toc23111168)

[5.1. MÉDIA E DESVIO PADRÃO 9](#_Toc23111169)

[5.1.1. TABELA DE MÉDIAS 9](#_Toc23111170)

[5.1.2. GRÁFICOS DE MÉDIAS 9](#_Toc23111171)

[5.1.3. TABELA DE DESVIOS PADRÕES 11](#_Toc23111172)

[5.1.4. GRÁFICOS DE DESVIOS PADRÕES 11](#_Toc23111173)

[5.1. BREVE DISCUSSÃO DAS DUAS ABORDAGENS 12](#_Toc23111174)

[6. CONSIDERAÇÕES FINAIS 12](#_Toc23111175)

[7. ANEXOS 13](#_Toc23111176)

[8. BIBLIOGRAFIA 14](#_Toc23111177)

# 1. DESCRIÇÃO DO PROBLEMA

Esta documentação é referente ao trabalho prático 2 da disciplina de Algoritmos 1 do curso de Sistemas de Informação da UFMG.

A motivação do trabalho foi baseada na compreensão e implementação de dois paradigmas de programação algorítmica: (1) Algoritmos gulosos e (2) Programação dinâmica. Conforme aula de orientação da monitoria, a solução implementada considerou o problema da Mochila.

A proposta do trabalho apresentava um cenário hipotético: Luiz e suas amigas são fãs da série La Casa de Papel (Netflix) e um dos episódios é gravado em uma das ilhas do arquipélago de San Blas (Panamá). Elas querem viajar para conhecer essas ilhas paradisíacas, uma vez que as mudanças climáticas ameaçam a existência deste arquipélago nos próximos 80 anos.

Nesse cenário, o grupo de Luiz economizou dinheiro por algum tempo e iniciou um planejamento de quais ilhas elas irão visitar. O planejamento é assim descrito: o grupo estabeleceu notas para cada uma das ilhas (possíveis de visitação) de acordo com o quanto que elas gostariam de conhecer o local. Também, pesquisaram o custo de ficar em cada ilha por dia (transporte, acomodação e alimentação).

Por fim, dada a limitação do orçamento, o grupo de amigas conversou e decidiu analisar dois tipos de roteiro: (caso 1) dado um orçamento máximo disponível, qual a maior pontuação possível ao escolher um conjunto de ilhas podendo ocorrer repetições e (caso 2) qual a maior pontuação possível, sem repetições de ilhas.

As seções seguintes estão assim estruturadas:

* descrições técnicas sobre hardwares, softwares e linguagem de programação utilizados ao longo do desenvolvimento do trabalho;
* estruturas de dados e dos algoritmos implementados como parte da solução do problema;
* apresentação das análises de complexidade, espaço e tempo, da solução;
* avaliação experimental de testes utilizando os datasets disponibilizados pelos monitores da disciplina e outros criados.

# 2. DESCRIÇÕES TÉCNICAS

## 2.1. HARDWARE

O desenvolvimento e testes do programa foram em um dispositivo cujas configurações são: (1) Intel® Core™ i5-7200U CPU 2.50GHz 2.71GHz, (2) 8GB de memória RAM, (3) Sistema Operacional Windows Pro 10 x64 com Linux Ubuntu rodando como subsistema.

## 2.2. SOFTWARE

Os programas utilizados foram: (1) Visual Studio Code, editor de código-fonte desenvolvido pela Microsoft. Para criar a estrutura do projeto do programa, utilizou-se a extensão do VS (2) Easy C++ projects.

## 2.3. LINGUAGEM DE PROGRAMAÇÃO

Programa desenvolvido em linguagem C++, com utilização somente das bibliotecas padrões disponíveis até a versão C++17 (2014). Não são necessárias bibliotecas adicionais, de terceiros e/ou que exijam a instalação para compilação e execução do programa.

# 3. ESTRUTURAS DE PROJETO E ALGORITMOS

## 3.1. DIRETÓRIO DO PROJETO

A estrutura dos projetos provê:

* dataset: estão os arquivos de entrada (subpasta input) disponibilizados no ambiente Moodle da disciplina, arquivos de saída (subpasta ouput) e um gerador de entradas.
* docs: constam os arquivos do tipo documento (.pdf, .doc etc.) disponibilizado no ambiente Moodle da disciplina e produzidos ao longo do trabalho como, por exemplo, a documentação do trabalho;
* src: constam todos os arquivos de header (.h) e de implementação/corpo (.cpp), que possuem o código dos métodos assinados em .h, bem como o arquivo main.cpp, bem como o makefile;
* pasta raiz: contém os outros arquivos como, por exemplo, arquivos do GitHub (.gitattributes, gitignore e README.md)

## 3.2. PARADIGMAS DE PROGRAMAÇÃO

Como parte da solução deste problema, recorreu-se ao entendimento e implementação dos seguintes paradigmas de programação algorítmica:

|  |  |
| --- | --- |
| **PARADIGMA** | **DESCRIÇÃO (ALTO NÍVEL)** |
| **Guloso** | Algoritmo que sempre faz a escolha que parecer ser a melhor no momento. Ou seja, ele faz uma escolha ótima para as condições locais, na esperança de que essa escolha leve a uma solução ótima para a situação global. Nem sempre produzem soluções ótimas, mas para muitos problemas eles são úteis. |
| **Programação dinâmica** | Algoritmo que resolve problemas combinando as soluções para subproblemas. É aplicável quando os subproblemas não são independentes, isto é, quando os subproblemas compartilham subproblemas. Assim, o algoritmo resolve cada subproblema uma vez só e, então, grava sua resposta em uma tabela, evitando assim o trabalho de recalcular a resposta toda vez que o subproblema é encontrado. |

## 3.3. PROBLEMAS ALGORÍTMICOS

Como parte da solução deste problema, recorreu-se ao entendimento e implementação do(s) seguinte(s) problema(s) algorítmico(s):

|  |  |
| --- | --- |
| **PROBLEMAS** | **DESCRIÇÃO (ALTO NÍVEL)** |
| **Mochila** | Problema de otimização combinatória. O nome dá-se devido ao modelo de uma situação em que é necessário preencher uma mochila com objetos de diferentes pesos e valores. O objetivo é que se preencha a mochila com o maior valor possível, não ultrapassando o peso máximo. |
| **Mochila fracionária** | Problema de otimização combinatória, uma extensão do problema da Mochila. O objetivo é encher uma mochila com quantidades fracionárias de diferentes objetos visando maximizar o valor dos materiais selecionados. Ainda, o problema da mochila fracionária pode ser resolvido em tempo polinomial, mostrando como uma mudança aparentemente pequena na estrutura de um problema pode ter um significativo impacto na complexidade computacional. |

## 3.4. PRINCÍPIOS DE PROJETOS DA SOLUÇÃO PROPOSTA ALGORÍTMICOS

A solução proposta perpassa por dois pontos macros:

1. ordenação das entradas por algum critério – implementado um mergesort para isso;
2. solução ótima para cada subproblema de maneira que, ao produzir soluções ótimas para os subproblemas, o problema apresente solução ótima.

Descreve-se como a solução foi pensada e implementada para cada algoritmo.

|  |  |
| --- | --- |
| **PROBLEMAS** | **DESCRIÇÃO (ALTO NÍVEL)** |
| **Algoritmo Guloso** | A partir da definição de algoritmo guloso e dos problemas algorítmicos, buscou-se encontrar uma estrutura do problema, onde a repetição de ilhas é permitida e a solução ótima considerasse esse fato e o orçamento total disponível.  A partir de algumas observações, percebeu-se que o quociente custo\_diário/pontuação para cada ilha permitiria encontrar uma relação do tipo “melhor custo benefício”. Ordenando as ilhas crescentemente, quanto mais à esquerda melhor a relação custo\_diário/pontuação da ilha. De outra forma, quanto mais à direita, pior essa relação.  A cada iteração do algoritmo, adiciona a melhor solução disponível naquele momento, dado que a solução ótima para cada subproblema é adicionar ilhas de “melhor custo benefício” visando, portanto, a maximização da pontuação a partir dessa estratégia de construção do roteiro dentro de um orçamento máximo. |
| **Programação Dinâmica** | A partir da definição de programação dinâmica e dos problemas algorítmicos, buscou-se encontrar uma estrutura do problema, onde não é permitida a repetição de ilhas e a solução ótima considerasse esse fato e o orçamento total disponível.  A partir de algumas observações, percebeu-se que a solução deveria considerar as pontuações definidas para cada ilha de tal forma que, a cada iteração do algoritmo, a ilha com maior pontuação significativa e compatível com o orçamento é adicionada. Uma vez adicionada no conjunto solução (roteiro), ela passa ser desconsiderada e, as próximas iterações do programa visam adicionar outra(s) ilha(s) considerando a estratégia anterior especificada. |

# 4. ANÁLISE DE COMPLEXIDADE

Segundo especificação do trabalho, os algoritmos implementados deveriam ter, cada qual, uma complexidade máxima especificada:

* **Algoritmo guloso:** o tempo de execução para o problema não pode ser superior a O(m log m);
* **Algoritmo de programação dinâmica:** o tempo de execução para o problema não pode ser superior a O(n \* m).

## 4.1. STRUCT E CLASSES

A solução implementa uma struct e uma classe. A seguir, um resumo do “tamanho” destes objetos:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Classe** | **Nome** | **Tipo** | **Atributos** | **Métodos** | | |
| **public** | **protected** | **private** |
| Viagem.h | Ilha | struct | 4 | - | - | - |
| Viagem.h | Viagem | class | - | 14 | - | 12 |

## 4.2. TEMPO

Conforme especificação, a validação do tempo de execução do algoritmo será feito na seção de Avaliação Experimental considerando as complexidades **O(m log m)** para o **Algoritmo Guloso** e **O (n \* m)** para o **Algoritmo de Programação Dinâmica.**

## 4.3. ESPAÇO

O espaço de memória das principais estruturas implementadas como solução desse problema tem tamanho **polinomial O(n²)**. A seguir, detalham-se essas estruturas:

|  |  |  |  |
| --- | --- | --- | --- |
| **Arquivo** | **Objeto** | **Espaço** | **Descrição** |
| **Main.cpp** | viagem\_guloso | O(n \* s) | Objeto da classe viagem que armazena “n” ilhas da entrada numa estrutura (struct) de “s” campos. |
| **Main.cpp** | viagem\_dinamico | O(n \* s) | Objeto da classe viagem que armazena “n” ilhas da entrada numa estrutura (struct) de “s” campos. |

## 4.2. PROVA DE CORRETUDE

|  |  |
| --- | --- |
| **ALGORITMO** | **PROVA DE CORRETUDE** |
| **Mergesort** | **Passo base:** n = 1. Um conjunto com um único elemento está ordenado.  **Passo indutivo:** Seja S um conjunto de n >= 2 inteiros e x um elemento qualquer de S. Podemos particionar S em dois conjuntos, S1 e S2, de tamanhos n/2 (piso) e n/2 (teto). Como n >= 2, ambos S1 e S2 possuem menos de n elementos.  Por hipótese de indução, sabemos ordenar os conjuntos S1 e S2 . Podemos, então, obter S ordenado intercalando os conjuntos ordenados S1 e S2. |
| **Guloso** | Dado o tamanho da prova e a impossibilidade de reproduzi-la na íntegra aqui, apresenta-se uma prova de corretude para o caso dos objetos estarem ordenados em ordem decrescente de acordo com o valor por peso. A prova está nos anexos.  Observe que o algoritmo implementado ordena em ordem crescente os objetos em termo do valor preço. Observe, ainda que a prova, considera objetos em ordem decrescente. Considere, então, que um Vcrescenete = Vdecrescenteinverso. |
| **Programação Dinâmica** | Dado o tamanho da prova e a impossibilidade de reproduzi-la na íntegra aqui, apresenta-se uma prova de corretude para o caso dos objetos estarem ordenados em ordem decrescente de acordo com o valor por peso. A prova está nos anexos.  Observe que o algoritmo implementado ordena em ordem decrescente por pontuação os objetos. Segue, então, diretamente da prova. |

# 5. AVALIAÇÃO EXPERIMENTAL

Os testes compreenderam a execução de **20 arquivos**, com diferentes tamanhos de entradas: (1) **4 arquivos** são referentes àqueles disponibilizados pelos monitores da disciplina, (2) **7 arquivos** disponibilizados por outros alunos no Moodle e (3) **9 arquivos** próprios gerados.

Os testes foram executados um **total de 10 execuções simultâneas** com **custos, pontuações e número de ilhas variadas**, conforme instruções no arquivo de especificação do trabalho

## 5.1. MÉDIA E DESVIO PADRÃO

### 5.1.1. TABELA DE MÉDIAS

A tabela abaixo apresenta a média (em ms) de 10 execuções consecutivas de cada arquivo de entrada:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input1** | **Input2** | **Input3** | **Input4** | **Input5** | **Input6** | **Input7** | **Input8** | **Input9** | **Input10** |
| 0,000 | 0,000 | 0,000 | 6,250 | 0,000 | 0,000 | 1,563 | 0,000 | 0,000 | 0,000 |
| **Input11** | **Input12** | **Input13** | **Input14** | **Input15** | **Input16** | **Input17** | **Input18** | **Input19** | **Input20** |
| 0,000 | 0,000 | 0,000 | 3,125 | 1,563 | 1,563 | 7,813 | 4,688 | 1,563 | 1,563 |
| **Média Global** | | | | | | | | **1,484** | |

### 5.1.2. GRÁFICOS DE MÉDIAS

O valor “Tm” representa o **tempo médio** de todas as **10 execuções simultâneas realizadas para um mesmo arquivo de entrada.** Graficamente, é possível identificar quais arquivos consumiram maior tempo médio de execução. Destacam-se os arquivos: **input4, input17 e input18.**

### 5.1.3. TABELA DE DESVIOS PADRÕES

A tabela abaixo apresenta os desvios padrões das 10 execuções de cada arquivo de entrada:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input1** | **Input2** | **Input3** | **Input4** | **Input5** | **Input6** | **Input7** | **Input8** | **Input9** | **Input10** |
| 0,000 | 0,000 | 0,000 | 8,069 | 0,00 | 0,00 | 4,941 | 0,000 | 0,000 | 0,000 |
| **Input11** | **Input12** | **Input13** | **Input14** | **Input15** | **Input16** | **Input17** | **Input18** | **Input19** | **Input20** |
| 0,000 | 0,000 | 0,000 | 6,588 | 4,941 | 4,941 | 8,235 | 7,548 | 4,941 | 4,941 |

### 5.1.4. GRÁFICOS DE DESVIOS PADRÕES

## 5.1. BREVE DISCUSSÃO DAS DUAS ABORDAGENS

As abordagens apresentadas, Gulosa e Dinâmica, visam maximizar a pontuação do roteiro dado um orçamento de viagem.

Caso o grupo viajante considere mais vantajoso aumentar o tempo de estadia no local, a abordagem Gulosa deve ser preferida, uma vez que é possível se manter mais dias viajando, a que pense contra conhecer menos ilhas. Há uma priorização em conhecer ilhas do tipo “melhor custo benefício”.

Caso o grupo viajante considere mais vantajoso conhecer mais lugares, a abordagem de Programação Dinâmica deve ser preferida, dado que não haverá repetição de ilhas, a maximização dos pontos leva a conhecer lugares em que o grupo considera “mais interessante”.

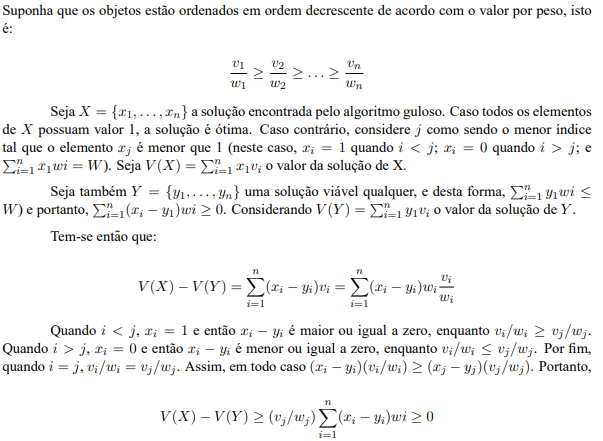
# 6. CONSIDERAÇÕES FINAIS

O trabalho prático 2 foi menos complexo que o trabalho anterior. Considera-se que a complexidade foi assimilar os conceitos dos dois paradigmas de programação utilizados, entender o problema como um todo e, a partir daí, buscar uma solução ótima a partir da identificação da estrutura do problema.

Ao implementar dois algoritmos baseados em paradigmas distintos e que, às vezes, conforme visto na literatura, podem resolver os mesmos problemas, a proposta do trabalho guiou a entender que uma determinada estrutura de problema pode ter uma solução preferível do que outra, focando-se na complexidade de tempo para se obter uma solução. O foco final, então, visa levar a uma solução ótima, a partir da resolução de subproblemas de um problema maior.

# 7. ANEXOS

**Anexo 1 – Prova de Corretude**





# 8. BIBLIOGRAFIA

1. Problema da Mochila. Disponível em: <https://pt.wikipedia.org/wiki/Problema\_da\_mochila> Acesso em: 14 out 2019;
2. Problema da Mochila Fracionada. Disponível em: <https://en.wikipedia.org/wiki/Continuous\_knapsack\_problem> Acesso em: 14 out 2019;
3. CORMEN, T. H et al. Algoritmos Teória e Prática. Tradução Vandenberg Souza. Editora Campus, 4ª tiragem. Disponível em: <<https://www.cin.ufpe.br/~ara/algoritmos-%20portugu%EAs-%20cormen.pdf>> Acesso em: 15 out 2019;