

**Análise Algorítmica e Otimização**

Ano Letivo de 2016/2017

**Problema da Mochila**

**Joel Pereira 8150138**

**José Silva 8150141**

**José Bernardes 8150148**

**Tiago Bandeira 8150456**

Maio, 2017

**Agradecimentos**

Desejámos exprimir os nossos agradecimentos a todos aqueles que, de alguma forma, nos ajudaram a ultrapassar todos os obstáculos com vista a concluirmos este trabalho com bastante qualidade.

Em primeiro lugar, queremos agradecer ao Mestre Carlos Pereira, nosso professor da unidade curricular de AAO, por toda a sabedoria e conhecimento que nos transmitiu no decorrer do desenvolvimento do trabalho. Com a sua ajuda tornou-se muito mais simples entender e resolver os problemas pedidos e, por isso, deixamos aqui o nosso maior agradecimento.

Em segundo lugar, queremos deixar o nosso agradecimento aos nossos colegas de turma, que de uma forma ou de outra, contribuíram para o sucesso do nosso trabalho e por isso consideramos importante também deixar aqui o nosso agradecimento.

**Resumo**

Este trabalho tem como objetivo estudar o problema da mochila (*Knapsack Problem*) dando uso a um algoritmo de Programação Dinâmica. A explicação do problema será feita nas páginas abaixo. A explicação será feita por passos. Primeiramente, será feita uma apresentação e, posteriormente, será descrito o problema com mais detalhe, apresentando termos técnicos. O algoritmo será apresentado no final juntamente com um output mostrando a solução. É necessário dizer também que, para resolver o problema usamos a linguagem de programação denominada JAVA.

**Índice**

1. Introdução1

1.1. Contextualização1

1.2. Apresentação do Caso de Estudo1

1.3. Motivação e Objetivos1

1.4. Estrutura do Relatório1

**2. Problema da Mochila2**

2.1. O Problema2

2.2. Programação Dinâmica2

2.3. Solução do Problema2

2.4. Metodologia adotada na implementação3

**3.Analise do Algoritmo 4**

3.1. Capacidade da Mochila = 54

3.1. Capacidade da Mochila = 107

3.1. Capacidade da Mochila = 2010

**4. Resultados 13**

**5.Conclusão e Trabalho Futuro 14**

**Referencias WWW 15**

**Lista de Siglas e Acrónimos 16**

**Anexo I 17**

**Índice de Tabelas**

Tabela 1: W=5 & n=54

Tabela 1a: Descrição dos objetos W=5 & n=54

Tabela 2: W=5 & n=10 5

Tabela 2a: Descrição dos objetos W=5 & n=10 5

Tabela 3: W=5 & n=15 6

Tabela 3a: Descrição dos objetos W=5 & n=15 6

Tabela 4: W=10 & n=5 7

Tabela 4a: Descrição dos objetos W=10 & n=5 7

Tabela 5: W=10 & n=10 8

Tabela 5a: Descrição dos objetos W=10 & n=10 8

Tabela 6: W=10 & n=15 9

Tabela 6a: Descrição dos objetos W=10 & n=15 9

Tabela 7: W=20 & n=5 10

Tabela 7a: Descrição dos objetos W=20 & n=5 10

Tabela 8: W=20 & n=10 11

Tabela 8a: Descrição dos objetos W=20 & n=10 11

Tabela 9: W=20 & n=15 12

Tabela 9a: Descrição dos objetos W=20 & n=15 12

Tabela 10: Resumo dos resultados13

1. **Introdução**
   1. **Contextualização**

Este relatório é parte integrante da análise e desenvolvimento do algoritmo de resolução do problema da mochila.

O problema da mochila é um problema de otimização combinatória, o que o integra no âmbito da cadeira de AAO.

* 1. **Apresentação do Caso de Estudo**

O caso de estudo em questão é o Problema da Mochila.

Suponhamos que existe um conjunto de objetos com um determinado peso e valor e uma mochila com capacidade limitada. O objetivo do problema da mochila é escolher a combinação dos objetos a introduzir, tendo em consideração que o seu peso não pode exceder a capacidade da mochila e o somatório do valor dos mesmos seja o máximo possível.

* 1. **Motivação e Objetivos**

Uma das motivações que nos levou a estudar o problema em questão foi a enorme quantidade de problemas do dia-a-dia que podem ser resolvidos, se for implantada uma metodologia como a que permite resolver o problema da mochila.

Os objetivos deste trabalho são a implementação de um algoritmo para a resolução do problema apresentado e posterior análise e estudo da complexidade do mesmo.

* 1. **Estrutura do Relatório**

Este relatório está estruturado de uma forma sequencial para ser de fácil compreensão por parte da pessoa que o esteja a ler.

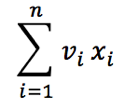
Inicialmente, será apresentado o problema de modo a dá-lo a conhecer ao leitor. Será descrito também a metodologia que adotamos e a forma que ela funciona. De seguida, serão apresentadas as tabelas com as soluções e, no final desta parte, um resumo para comparar os valores obtidos com diferentes cenários.

Por fim, será apresentado, em anexo, a implementação do algoritmo que usamos.

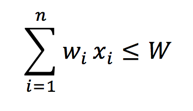
1. **Problema da Mochila**
   1. **O Problema**

O Problema da Mochila é um problema de otimização combinatória em que o objetivo é preencher uma mochila com o maior valor possível, não ultrapassando o peso máximo da mesma [01].

O Problema da Mochila é um problema NP-Completo, isto significa que o problema pode ser resolvido em tempo polinomial. Utilizando programação dinâmica, o problema da mochila torna-se um problema pseudo-polinomial, e assenta no seguinte modelo:



Maximizar



sujeito a onde

Legenda:

corresponde ao objeto

corresponde ao peso do objeto

corresponde ao valor do objeto

corresponde à capacidade da Mochila

corresponde ao número máximo de objetos

* 1. **Programação Dinâmica**

A Programação Dinâmica é uma técnica poderosa que pode ser usada para resolver muitos problemas no tempo O(n2) ou O(n3) para os quais uma abordagem ingénua levaria tempo exponencial (geralmente para obter um tempo de execução abaixo disso - se for possível - seria necessário acrescentar outras ideias também). A Programação Dinâmica é uma abordagem baseada numa fórmula recorrente e um (ou alguns) estados iniciais. Uma sub-solução do problema é construída a partir dos encontrados anteriormente. As soluções DP têm uma complexidade polinomial que garantem um tempo de execução muito mais rápido do que outras técnicas como retrocesso, força bruta, etc.

* 1. **Solução do problema**

Para a resolução do problema foi escolhida a programação dinâmica como metodologia a adotar.

Com a programação dinâmica é possível encontrar uma solução ótima sem que sejam ré- efetuados cálculos, o que agiliza bastante o algoritmo. Por outro lado, par que sejam evitados esses recálculos é necessário utilizar uma matriz para armazenar esses resultados para mais tarde serem utilizados.

Com esta abordagem conseguimos uma melhor qualidade do resultado, já que são poupados recálculos ao longo do algoritmo, no entanto se a quantidade de dados for elevada vai fazer com que a matriz seja consequentemente muito grande, o que pode causar problemas na execução do algoritmo.

* 1. **Metodologia adotada na implementação**

Para implementar uma solução do problema da mochila tendo por base a programação dinâmica é necessário:

1. Indicar a capacidade máxima da mochila, o peso de cada objeto e o seu correspondente valor;
2. Construir uma matriz com o n+1 linhas (n sendo o numero de objetos) e w+1 colunas (w sendo o peso máximo da mochila);
3. Preencher toda a matriz com 0 (como a nossa implementação prática foi efetuada em linguagem java esse passo pode ser omitido no algoritmo pois esta inicializa a matriz já com o valor de 0);
4. Preencher as células de cada linha da matriz com o valor da mesma coluna na linha anterior, se o peso do objeto for menor que o valor dessa coluna. Se o peso do objeto for igual ao valor da coluna, escolher o maior valor entre o valor da correspondente a essa coluna na minha anterior, e a soma do peso do objeto atual com o valor da linha anterior na primeira coluna;
5. O passo 4 é efetuado tantas vezes quanto o numero de linhas da matriz;
6. O valor máximo que o algoritmo calculou, tendo em conta a capacidade da mochila previamente indicada, será o valor da ultima coluna, na ultima linha da matriz;
7. **Analise do Algoritmo**
   1. **Capacidade da Mochila = 5**

* 5 Objetos

**Capacidade da Mochila**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 10 | 10 |
| 2 | 0 | 0 | 0 | 0 | 10 | 10 |
| 3 | 0 | 0 | 0 | 0 | 10 | 10 |
| 4 | 0 | 0 | 0 | 0 | 10 | 11 |
| 5 | 0 | 0  Tabela 1: W=5 & n=5 | 0 | 0 | 10 | 11 |

**Itens**

|  |  |  |
| --- | --- | --- |
| Objeto | Peso | Valor |
| 0 | 4 | 10 |
| 1 | 8 | 8 |
| 2 | 6 | 3 |
| 3 | 5 | 11 |
| 4 | 9 | 7 |

Tempo de execução: 7137 ns;

Numero de condições: 28;

Objetos pertencentes á solução ótima: 3;

Tabela 1a: Descrição dos objetos W=5 & n=5

* 10 Objetos

**Capacidade da Mochila**

**Itens**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 10 | 10 |
| 2 | 0 | 0 | 0 | 0 | 10 | 10 |
| 3 | 0 | 0 | 0 | 0 | 10 | 10 |
| 4 | 0 | 0 | 0 | 0 | 10 | 11 |
| 5 | 0 | 0 | 0 | 0 | 10 | 11 |
| 6 | 0 | 0 | 12 | 12 | 12 | 12 |
| 7 | 0 | 3 | 12 | 15 | 15 | 15 |
| 8 | 0 | 3 | 12 | 15 | 15 | 15 |
| 9 | 0 | 3 | 12 | 15 | 15 | 15 |
| 10 | 0 | 3 | 12 | 15 | 15 | 15 |

Tabela 2: W=5 & n=10

|  |  |  |
| --- | --- | --- |
| Objeto | Peso | Valor |
| 0 | 4 | 10 |
| 1 | 8 | 8 |
| 2 | 6 | 3 |
| 3 | 5 | 11 |
| 4 | 9 | 7 |
| 5 | 2 | 12 |
| 6 | 1 | 3 |
| 7 | 10 | 5 |
| 8 | 7 | 9 |
| 9 | 6 | 5 |

Tempo de execução: 12102ns;

Numero de condições: 62;

Objetos pertencentes á solução ótima: 5, 6;

Tabela 2a: Descrição dos objetos W=5 & n=10

* 15 Objetos

**Capacidade da Mochila**

**Itens**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 10 | 10 |
| 2 | 0 | 0 | 0 | 0 | 10 | 10 |
| 3 | 0 | 0 | 0 | 0 | 10 | 10 |
| 4 | 0 | 0 | 0 | 0 | 10 | 11 |
| 5 | 0 | 0 | 0 | 0 | 10 | 11 |
| 6 | 0 | 0 | 12 | 12 | 12 | 12 |
| 7 | 0 | 3 | 12 | 15 | 15 | 15 |
| 8 | 0 | 3 | 12 | 15 | 15 | 15 |
| 9 | 0 | 3 | 12 | 15 | 15 | 15 |
| 10 | 0 | 3 | 12 | 15 | 15 | 15 |
| 11 | 0 | 6 | 12 | 18 | 21 | 21 |
| 12 | 0 | 6 | 12 | 18 | 21 | 21 |
| 13 | 0 | 6 | 12 | 18 | 21 | 22 |
| 14 | 0 | 6 | 12 | 18 | 21 | 22 |
| 15 | 0 | 6 | 12 | 18 | 21 | 27 |

Tabela 3: W=5 & n=15

|  |  |  |
| --- | --- | --- |
| Objeto | Peso | Valor |
| 0 | 4 | 10 |
| 1 | 8 | 8 |
| 2 | 6 | 3 |
| 3 | 5 | 11 |
| 4 | 9 | 7 |
| 5 | 2 | 12 |
| 6 | 1 | 3 |
| 7 | 10 | 5 |
| 8 | 7 | 9 |
| 9 | 6 | 5 |
| 10 | 1 | 6 |
| 11 | 4 | 8 |
| 12 | 3 | 10 |
| 13 | 7 | 7 |
| 14 | 2 | 9 |

Tempo de execução: 12412 ns;

Numero de condições: 101;

Objetos pertencentes á solução ótima: 5, 10, 14;

Tabela 3a: Descrição dos objetos W=5 & n=15

* 1. **Capacidade da Mochila = 10**
* 5 Objetos

**Itens**

**Capacidade da Mochila**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 2 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 3 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 | 13 |
| 4 | 0 | 0 | 0 | 0 | 10 | 11 | 11 | 11 | 11 | 21 | 21 |
| 5 | 0 | 0 | 0 | 0 | 10 | 11 | 11 | 11 | 11 | 21 | 21 |

Tabela 4: W=10 & n=5

|  |  |  |
| --- | --- | --- |
| Objeto | Peso | Valor |
| 0 | 4 | 10 |
| 1 | 8 | 8 |
| 2 | 6 | 3 |
| 3 | 5 | 11 |
| 4 | 9 | 7 |

Tempo de execução: 8998 ns;

Numero de condições: 73;

Objetos pertencentes á solução ótima: 0, 3;

Tabela 4a: Descrição dos objetos W=10 & n=5

* 10 Objetos

**Itens**

**Capacidade da Mochila**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 2 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 3 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 | 13 |
| 4 | 0 | 0 | 0 | 0 | 10 | 11 | 11 | 11 | 11 | 21 | 21 |
| 5 | 0 | 0 | 0 | 0 | 10 | 11 | 11 | 11 | 11 | 21 | 21 |
| 6 | 0 | 0 | 12 | 12 | 12 | 22 | 23 | 23 | 23 | 23 | 23 |
| 7 | 0 | 3 | 12 | 15 | 15 | 15 | 22 | 25 | 26 | 26 | 26 |
| 8 | 0 | 3 | 12 | 15 | 15 | 15 | 22 | 25 | 26 | 26 | 26 |
| 9 | 0 | 3 | 12 | 15 | 15 | 15 | 22 | 25 | 26 | 26 | 26 |
| 10 | 0 | 3 | 12 | 15 | 15 | 15 | 22 | 25 | 26 | 26 | 26 |

|  |  |  |
| --- | --- | --- |
| Objeto | Peso | Valor |
| 0 | 4 | 10 |
| 1 | 8 | 8 |
| 2 | 6 | 3 |
| 3 | 5 | 11 |
| 4 | 9 | 7 |
| 5 | 2 | 12 |
| 6 | 1 | 3 |
| 7 | 10 | 5 |
| 8 | 7 | 9 |
| 9 | 6 | 5 |

Tabela 5: W=10 & n=10

Tempo de execução: 13963ns;

Numero de condições: 152;

Objetos pertencentes á solução ótima: 3, 5, 6;

Tabela 5a: Descrição dos objetos W=10 & n=10

* 15 Objetos

Tabela 6: W=10 & n=15

**Itens**

**Capacidade da Mochila**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 2 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 3 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 | 13 |
| 4 | 0 | 0 | 0 | 0 | 10 | 11 | 11 | 11 | 11 | 21 | 21 |
| 5 | 0 | 0 | 0 | 0 | 10 | 11 | 11 | 11 | 11 | 21 | 21 |
| 6 | 0 | 0 | 12 | 12 | 12 | 22 | 23 | 23 | 23 | 23 | 23 |
| 7 | 0 | 3 | 12 | 15 | 15 | 15 | 22 | 25 | 26 | 26 | 26 |
| 8 | 0 | 3 | 12 | 15 | 15 | 15 | 22 | 25 | 26 | 26 | 26 |
| 9 | 0 | 3 | 12 | 15 | 15 | 15 | 22 | 25 | 26 | 26 | 26 |
| 10 | 0 | 3 | 12 | 15 | 15 | 15 | 22 | 25 | 26 | 26 | 26 |
| 11 | 0 | 6 | 12 | 18 | 21 | 21 | 22 | 28 | 31 | 32 | 32 |
| 12 | 0 | 6 | 12 | 18 | 21 | 21 | 22 | 28 | 31 | 32 | 32 |
| 13 | 0 | 6 | 12 | 18 | 21 | 22 | 28 | 31 | 31 | 32 | 38 |
| 14 | 0 | 6 | 12 | 18 | 21 | 22 | 28 | 31 | 31 | 32 | 38 |
| 15 | 0 | 6 | 12 | 18 | 21 | 27 | 30 | 31 | 37 | 40 | 40 |

|  |  |  |
| --- | --- | --- |
| Objeto | Peso | Valor |
| 0 | 4 | 10 |
| 1 | 8 | 8 |
| 2 | 6 | 3 |
| 3 | 5 | 11 |
| 4 | 9 | 7 |
| 5 | 2 | 12 |
| 6 | 1 | 3 |
| 7 | 10 | 5 |
| 8 | 7 | 9 |
| 9 | 6 | 5 |
| 10 | 1 | 6 |
| 11 | 4 | 8 |
| 12 | 3 | 10 |
| 13 | 7 | 7 |
| 14 | 2 | 9 |

Tempo de execução: 16136 ns;

Numero de condições: 240;

Objetos pertencentes á solução ótima: 5, 6, 10, 12, 14;

Tabela 6a: Descrição dos objetos W=10 & n=15

* 1. **Capacidade da Mochila = 20**
* 5 Objetos

**Capacidade da Mochila**

**Itens**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| 2 | 0 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 21 | 21 | 21 | 21 | 21 | 33 | 33 | 33 | 33 |
| 3 | 0 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 24 | 24 | 24 | 24 | 24 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 45 |
| 4 | 0 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 24 | 24 | 24 | 24 | 24 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 45 |
| 5 | 0 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 24 | 24 | 24 | 24 | 24 | 36 | 36 | 36 | 36 | 36 | 36 | 41 | 45 |

Tabela 7: W=20 & n=5

|  |  |  |
| --- | --- | --- |
| Objeto | Peso | Valor |
| 0 | 5 | 12 |
| 1 | 12 | 21 |
| 2 | 8 | 24 |
| 3 | 20 | 19 |
| 4 | 6 | 5 |

Tempo de execução: 11482 ns;

Numero de condições: 154;

Objetos pertencentes á solução ótima: 1, 2;

Tabela 7a: Descrição dos objetos W=20 & n=5

* 10 Objetos

**Capacidade da Mochila**

**Itens**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 |
| 2 | 0 | 0 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 17 | 17 | 29 | 29 | 29 | 29 | 29 | 29 |
| 3 | 0 | 0 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 17 | 17 | 29 | 29 | 29 | 29 | 29 | 29 |
| 4 | 0 | 0 | 12 | 12 | 15 | 15 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 29 | 29 | 32 | 32 | 44 | 44 |
| 5 | 0 | 0 | 12 | 12 | 15 | 15 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 29 | 29 | 32 | 32 | 44 | 44 |
| 6 | 0 | 0 | 12 | 12 | 15 | 15 | 27 | 27 | 27 | 27 | 27 | 33 | 33 | 33 | 33 | 33 | 33 | 33 | 33 | 44 | 44 |
| 7 | 0 | 0 | 12 | 12 | 15 | 15 | 27 | 27 | 27 | 27 | 27 | 33 | 33 | 33 | 33 | 33 | 33 | 38 | 38 | 44 | 44 |
| 8 | 0 | 20 | 20 | 32 | 32 | 35 | 35 | 47 | 47 | 47 | 47 | 47 | 53 | 53 | 53 | 53 | 53 | 53 | 58 | 58 | 64 |
| 9 | 0 | 20 | 20 | 32 | 32 | 35 | 35 | 47 | 47 | 47 | 47 | 47 | 53 | 53 | 53 | 53 | 53 | 53 | 58 | 58 | 64 |
| 10 | 0 | 20 | 20 | 32 | 32 | 35 | 35 | 47 | 47 | 47 | 47 | 47 | 53 | 53 | 53 | 56 | 56 | 56 | 58 | 58 | 64 |

Tabela 8: W=20 & n=10

|  |  |  |
| --- | --- | --- |
| Objeto | Peso | Valor |
| 0 | 13 | 17 |
| 1 | 2 | 12 |
| 2 | 18 | 3 |
| 3 | 4 | 15 |
| 4 | 15 | 1 |
| 5 | 5 | 6 |
| 6 | 11 | 11 |
| 7 | 1 | 20 |
| 8 | 10 | 2 |
| 9 | 8 | 9 |

Tempo de execução: 19860 ns;

Numero de condições: 323;

Objetos pertencentes á solução ótima: 0, 1, 3, 7;

Tabela 8a: Descrição dos objetos W=20 & n=10

* 15 Objetos

**Capacidade da Mochila**

**Itens**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 2 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 |
| 3 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 |
| 4 | 0 | 0 | 0 | 0 | 0 | 3 | 6 | 6 | 6 | 6 | 6 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| 5 | 0 | 0 | 9 | 9 | 9 | 9 | 9 | 12 | 12 | 15 | 15 | 15 | 15 | 15 | 18 | 18 | 18 | 18 | 18 | 18 | 18 |
| 6 | 0 | 0 | 9 | 9 | 9 | 11 | 11 | 20 | 20 | 20 | 20 | 20 | 23 | 23 | 26 | 26 | 26 | 26 | 26 | 26 | 29 |
| 7 | 0 | 0 | 9 | 9 | 9 | 11 | 11 | 20 | 20 | 20 | 20 | 20 | 23 | 23 | 26 | 26 | 26 | 26 | 26 | 26 | 29 |
| 8 | 0 | 17 | 17 | 26 | 26 | 26 | 28 | 28 | 37 | 37 | 37 | 37 | 37 | 40 | 40 | 43 | 43 | 43 | 43 | 43 | 46 |
| 9 | 0 | 17 | 17 | 26 | 38 | 38 | 47 | 47 | 47 | 49 | 49 | 58 | 58 | 58 | 58 | 58 | 61 | 61 | 64 | 64 | 64 |
| 10 | 0 | 17 | 17 | 26 | 38 | 38 | 47 | 47 | 47 | 49 | 49 | 58 | 58 | 58 | 58 | 58 | 61 | 61 | 64 | 64 | 64 |
| 11 | 0 | 17 | 17 | 26 | 38 | 38 | 47 | 47 | 47 | 49 | 50 | 58 | 58 | 58 | 58 | 61 | 61 | 61 | 64 | 64 | 64 |
| 12 | 0 | 17 | 17 | 26 | 38 | 38 | 47 | 47 | 47 | 49 | 50 | 58 | 58 | 58 | 58 | 61 | 61 | 61 | 64 | 64 | 64 |
| 13 | 0 | 17 | 17 | 26 | 38 | 38 | 47 | 47 | 47 | 49 | 50 | 58 | 58 | 58 | 58 | 61 | 61 | 61 | 64 | 64 | 64 |
| 14 | 0 | 17 | 17 | 26 | 38 | 38 | 47 | 47 | 47 | 49 | 50 | 58 | 58 | 58 | 58 | 61 | 61 | 61 | 64 | 64 | 64 |
| 15 | 0 | 17 | 17 | 26 | 38 | 38 | 47 | 47 | 47 | 49 | 50 | 58 | 58 | 58 | 58 | 61 | 61 | 61 | 64 | 64 | 64 |

Tabela 9: W=20 & n=15

|  |  |  |
| --- | --- | --- |
| Objeto | Peso | Valor |
| 0 | 5 | 3 |
| 1 | 13 | 1 |
| 2 | 21 | 20 |
| 3 | 7 | 6 |
| 4 | 2 | 9 |
| 5 | 5 | 11 |
| 6 | 33 | 14 |
| 7 | 1 | 17 |
| 8 | 3 | 21 |
| 9 | 20 | 19 |
| 10 | 4 | 3 |
| 11 | 9 | 2 |
| 12 | 12 | 8 |
| 13 | 17 | 5 |
| 14 | 11 | 10 |

Tempo de execução: 25134 ns;

Numero de condições: 464;

Objetos pertencentes á solução ótima: 0, 4, 5, 7, 8, 10;

Tabela 9a: Descrição dos objetos W=20 & n=15

1. **Resultados**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 5 Objetos | | 10 Objetos | | 15 Objetos | |
| Comparações | Tempo (ns) | Comparações | Tempo (ns) | Comparações | Tempo (ns) |
| C=5 | 28 | 7137 | 62 | 12102 | 101 | 12412 |
| C=10 | 73 | 8998 | 152 | 13963 | 240 | 16136 |
| C=20 | 154 | 11482 | 323 | 19860 | 464 | 25134 |

Tabela 10: Resumo dos resultados das múltiplas execuções

Na tabela anterior estão esquematizadas as durações de execução de cada teste realizado, bem como o respetivo numero de comparações efetuadas pelo algoritmo para ser atingida a solução opima.

É importante referir que todos os testes foram efetuados na mesma maquina e em condições o mais aproximadas possível, visto que os tempos de execução são um parâmetro relativo á maquina em utilização e ás suas próprias condições.

Como podemos observar pela tabela, quando a capacidade da mochila duplica, o numero de comparações efectuadas tambem duplica, aproximadamente, e quando o numero de objetos duplica (5 obj – 10 obj) o numero de comparações tambem duplica.

Com esta constatação podemos dizer que o algoritmo terá uma complexidade de O(n\*W).

Podemos tambem observar que essa fatorização não é muito precisa, na medida em que, quanto mais aumentamos um dos fatores (W ou n), o numero de comparações é maior do que simplesmente o dobro, por isso quase que podemos dizer, e tal como foi teoricamente referido anteriormente, que o algoritmo será pseudo-polinomial, com uma complexidade de O(n\*2^W), mas nos casos de teste implementados o mais adequado será dizer que será O(n\*W).

1. **Conclusão e Trabalho Futuro**

Neste trabalho abordámos o problema da mochila (*KnapSack Problem*) que consiste na inserção de objetos (cada um com um peso e um valor) numa mochila até perfazer a capacidade da mochila, tentando maximizar o valor total da mochila (este valor total representa a soma dos valores de todos os objetos presentes na mochila) e, concluímos que sempre que aumentamos o numero de objetos e/ou a capacidade da mochila, a complexidade do algoritmo e o tempo de execução aumentam. Cumprimos todos os pressupostos que nos foram pedidos, no entanto a compreensão do algoritmo e a escolha da metodologia que deveríamos adotar foi a parte complicada devido ao facto de ser um problema novo e existirem muitas formas de proceder a sua resolução.

Consideramos a realização deste trabalho bastante importante porque elevou o nosso conhecimento e as nossas valências sobre a otimização e a análise de algoritmos, que são uma mais-valia para o nosso futuro profissional.

Para trabalho futuro, pensamos em resolver o problema adotando outras metodologias, analisando todas as componentes (complexidade, tempo de execução), de modo a verificar qual método seria mais eficaz e qual obterá melhores resultados.

**Referencias WWW**

[01] https://pt.wikipedia.org/wiki/Problema\_da\_mochila

[02] https://en.wikipedia.org/wiki/Knapsack\_problem

[03] https://en.wikipedia.org/wiki/Dynamic\_programming

**Lista de Siglas e Acrónimos**

**NP-Completo** Tempo polinomial não determinístico

**AAO** Análise Algorítmica e Otimização

**Anexo I – Implementação**

public class AAO4LIFE {

static int max(int a, int b) {

return (a > b) ? a : b;

}

static int knapsack(int W, int wt[], int val[], int n) {

long startTime = System.nanoTime();

int i, w, complex = 0;

int[][] K = new int[n + 1][W + 1];

for (i = 1; i <= n; i++) {

for (w = 1; w <= W; w++) {

complex++;

if (wt[i - 1] <= w) {

K[i][w] = max(val[i - 1] + K[i - 1][w - wt[i - 1]], K[i - 1][w]);

complex++;

} else {

K[i][w] = K[i - 1][w];

}

}

}

long endTime = System.nanoTime();

System.out.println("Tempo de execucao em nanosegundos: " + (endTime - startTime));

printMatrix(K);

System.out.println("Numero de condições: " + complex);

System.out.println("-----RESULTADO-----");

printSolution(K, n, val, wt, W);

return K[n][W];

}

public static void printMatrix(int[][] matrix) {

System.out.println("Matriz de valores");

for (int[] rows : matrix) {

for (int col : rows) {

System.out.format("%4d", col);

}

System.out.println();

}

System.out.println();

}

public static void printSolution(int[][] matrix, int n, int v[], int w[], int W) { //i

System.out.println("Objetos pertencentes à solução óptima:");

n--;

while (n >= 0) {

if (W - w[n] >= 0) {

if (matrix[n + 1][W] == v[n] + matrix[n][W - w[n]]) {

System.out.format("Objeto %d \n", n);

W -= w[n];

}

}

--n;

}

}

public static void main(String args[]) {

int[] wt = {20, 5, 3, 2, 10, 4, 6, 8, 15, 1};

int[] val = {10, 2, 3, 5, 7, 12, 6, 4, 1, 8};

int W = 20;

int n = wt.length;

System.out.println("Objeto | Peso | Valor");

for (int i = 0; i < n; i++) {

System.out.format(" %3d %3d %3d\n", i, wt[i], val[i]);

}

System.out.println("Valor máximo da mochila para a capacidade de {" + W + "}: " + knapsack(W, wt, val, n));

}

}