Disciplina: Projeto e Otimização de Algoritmos

Professor: Rafael Scopel

**Trabalho 01**

Arthur Both, Felipe Freitas e Gabriel Ferreira

**Sumário**

[Problema 1 – Juros 1](#_Toc637861944)

[1. O problema 2](#_Toc655978350)

[2. O Algoritmo 2](#_Toc2134490188)

[3. Análise do Algoritmo 2](#_Toc1470668608)

[4. Implementação e Tempo de Execução 2](#_Toc1738429829)

[Problema 2 – Multiplicação de Matrizes 3](#_Toc579155658)

[1. O problema 4](#_Toc1341357880)

[2. O Algoritmo 4](#_Toc501451781)

[3. Análise do Algoritmo 5](#_Toc1546897555)

[4. Implementação e Tempo de Execução 5](#_Toc1937819880)

[Fontes: 6](#_Toc650201906)

# **Problema 1 – Juros**

## **1. O problema**

Vocês estão abrindo uma empresa de modelos generativos e precisam de recursos para desenvolver 'n' diferentes modelos. O membro do time que era responsável por finanças, contratou 'n' empréstimos no valor de $ 1000 cada de vários bancos diferentes. O valor dos empréstimos fica mais caro de acordo com o passar do tempo: em particular, o empréstimo 'j' aumenta por uma taxa de juros 'rj' > 1 em cada mês, onde 'rj' é um determinado parâmetro. Isso significa que se o empréstimo 'j' for pago daqui a 't' meses, vocês terão que devolver ao banco {1000 ∙ '(rj)^t'}.

Assumiremos que todas as taxas de juros são distintas; isto é, 'ri' != 'rj' para taxas 'i' != 'j'.

Dado que a empresa só tem recursos para pagar um empréstimo por mês, em que ordem ela deve pagar os empréstimos para que o valor total gasto seja o menor possível?

## **2. O Algoritmo**

Para resolver o problema, temos de desenhar e implementar um algoritmo que considere as 'n' taxas de juros de preços 'r1', 'r2', […] , 'rn', e calcule uma ordem de pagamento dos empréstimos para que o valor total gasto seja minimizado. Este algoritmo deve ter tempo de execução polinomial em 'n'.

O primeiro passo do algoritmo é ordenar o vetor com os empréstimos, o que tem complexidade de O(n log n).

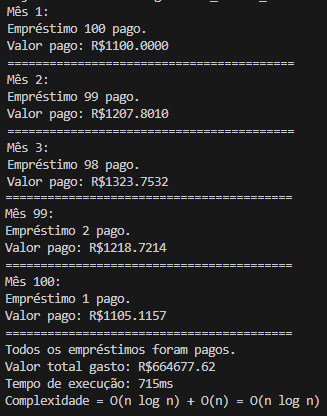
Ordenado o vetor, basta pegar os valores do vetor em ordem.

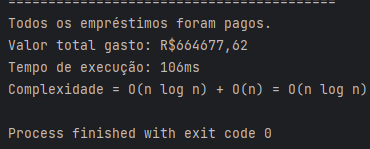
## **3. Análise do Algoritmo**

Sabemos que o algoritmo funciona visto que, dada uma taxa de juros in e outra taxa de juros in+1, como ambas iniciaram no mesmo ponto, é impossível que o empréstimo de menor juros ultrapasse o outro com o passar do tempo. Outro fator que contribui para isso é o fato de todos os empréstimos serem sobre a mesma base e um empréstimo apenas pode ser pago por mês.

## **4. Implementação e Tempo de Execução**

O código foi implementado tendo como base o número de empréstimos sendo 100, representado por “i”, seguindo a fórmula 1.001 + (i x 0.001), em que cada iteração consta um número diferente com base no empréstimo em questão. O programa imprime qual o empréstimo que está sendo pago, ao final de toda operação é mostrado que todos foram pagos e sua velocidade de execução.





Ele opera com a com a complexidade de O(n log n), já que os métodos para encontrar o empréstimo de maior taxa, pagar um empréstimo e avançar o mês, são dependentes do vetor que define o número de empréstimos.

# **Problema 2 – Multiplicação de Matrizes**

## **1. O problema**

Multiplicar matrizes é um problema que aparenta ser relativamente simples, e é amplamente utilizado em diversas áreas além da computação. A solução convencional de multiplicar uma matriz m x n por outra matriz n x p era tida como, até 1969, impossível de resolver em um tempo menor que θ(n³), porém, inconformado com a afirmação, Volker Strassen propôs um algoritmo para provar que há um método mais eficiente para resolver este problema.

## **2. O Algoritmo**

O algoritmo original de multiplicação de matrizes pode ser expresso mais ou menos da seguinte maneira (em pseudo-código):

Matriz A = [1 2 3

4 5 6

7 8 9]

Matriz B = [1 2 3

4 5 6

7 8 9]

Matriz C = []

Para i = 1 Até m Faça

Para j = 1 Até p Faça

C[i][j] = 0

Para k = 1 Até n Faça

C[i][j] = C[i][j] + A[i][k] \* B[k][j]

FimPara

FimPara

FimPara

O algoritmo acima apesar de resolver o problema da multiplicação, é muito complexo e possui complexidade O(n³) (por ter 3 laços que se repetem um dentro do outro), o que não é ideal.

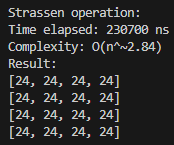
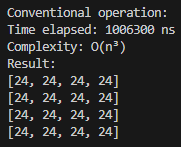
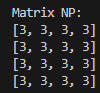
Para resolver este problema, ao menos parcialmente, Strassen idealizou um algoritmo capaz de multiplicar matrizes quadradas com lado de tamanho 2n, isto é, matrizes que possuem o mesmo número de linhas e colunas e que, também, tenham como lado um valor que seja uma potência de 2 (ex: 2, 4, 8, 64, 512). Baseado na técnica de divisão e conquista, faz uma abordagem diferente de multiplicar as matrizes diretamente e opta por fazer várias divisões menores/parciais.

Em alto nível, o algoritmo de Strassen realiza 3 operações principais:

1. Dividir a matriz em 4 submatrizes
2. Calcular 7 produtos intermediários, por meio de chamadas recursivas
3. Combinar os produtos intermediários para obter uma nova matriz com os resultados

## **3. Análise do Algoritmo**

Apesar de realizar menos multiplicações e, por tanto, não ter um custo de O(n³) como o original, o preço a se pagar por isso é o de memória, visto que a recursão em diversas submatrizes menores gera maior necessidade de armazenamento. Rodando o algoritmo com duas matrizes 4x4, a primeira preenchida com 2’s e a segunda com 3’s, obtivemos os seguintes resultados:



## **4. Implementação e Tempo de Execução**

Para atingir uma complexidade de aproximadamente O(nlog2(7)), vamos assumir T(n) como o tempo de execução deste algoritmo para multiplicar duas matrizes n x n. Voltando aos 3 passos anteriores (ver seção 2), sabemos que o passo 1 (Divisão) possui custo O(1), visto que apenas cria referências para as matrizes novas; o passo 2 (7 Cálculos intermediários) possui custo 7\*T(n/2), visto que opera sobre meia matriz; e o passo 3 (Combinação) possui custo O(n²), por realizar certas operações entre matrizes que não são otimizáveis.

Juntando tudo, temos que T(n) = O(1) + 7 \* T(n/2) + O(n²)

Aplicando o Teorema Mestre com as variáveis:

a = 7

b = 2

f(n) = O(n²)

Temos que

c = 2

logba = log27 ~= 2.81

Portanto, o algoritmo de Strassen executa em tempo inferior à O(n³), provando-se, nos casos em que é aplicável, uma ótima alternativa a multiplicação convencional.

# **Fontes:**

Slides de Aula

<https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_strassens_matrix_multiplication.htm#:~:text=Strassen's%20Matrix%20Multiplication%20is%20the,takes%20two%20loops%20to%20multiply>