

UNIVERSIDADE FEDERAL DE MINAS GERAIS  
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO  
DCC111 – MATEMÁTICA DISCRETA

## **Trabalho Prático I**

### **Problemas Combinatórios Soma máxima e Quadrado Mágico**

Alunos: Christian Vieira 2014106325  
Professor: Antonio Loureiro

Belo Horizonte  
31 de outubro de 2017

## Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Objetivo . . . . .	1
<b>2</b>	<b>Implementação</b>	<b>1</b>
2.1	Soma Máxima . . . . .	1
2.2	Quadrado Mágico . . . . .	1
<b>3</b>	<b>Ambiente de desenvolvimento</b>	<b>2</b>
3.1	Compilação . . . . .	2
3.2	Execução do programa . . . . .	2
3.3	Avaliação de alocação/desalocação de recursos . . . . .	2
<b>4</b>	<b>Resultados</b>	<b>3</b>
<b>5</b>	<b>Conclusões</b>	<b>4</b>
<b>6</b>	<b>Referências Bibliográficas</b>	<b>4</b>
	<b>Apêndice A Compilação dos programas</b>	<b>5</b>
	<b>Apêndice B Execução dos programas</b>	<b>6</b>

## Lista de Figuras

1	Processo de compilação. . . . .	5
2	Resultado do processo de compilação dos programas. . . . .	5
3	Exemplo de execução do programa <i>soma máxima (somamax)</i> . . . . .	6
4	Exemplo de execução do programa <i>quadrado mágico (qmagico)</i> . . . . .	6

## Lista de implementações

## Lista de Tabelas

## List of Algorithms

## 1 Introdução

A *Análise Combinatória* é a parte da *Matemática* que estuda os métodos de contagem. Esses estudos inicialmente foram desenvolvidos por *Tartaglia* em 1500 – 1557, *Fermat* em 1601 – 1665 e *Pascal* em 1632 – 1662. O objetivo principal de estudar *Análise Combinatória* é desenvolver métodos que permitam contar de uma forma indireta o número de elementos de um dado conjunto, estando os elementos agrupados sob certas condições.

### 1.1 Objetivo

Segundo especificação do trabalho prático: *Problemas Combinatórios: Soma máxima e Quadrado Mágico* (LOUREIRO, 2017), pede-se a implementação de dois programas em *linguagem C*, um para calcular a soma máxima dos elementos contido em um arranjo unidimensional (cujo tamanho do arranjo e elementos são previamente fornecidos) e o outro, para gerar quadrados mágicos cuja dimensão dos arranjos quadráticos é fornecida (no caso, 3, 4 e 5).

A implementação de soma máxima, deverá retornar o valor da soma máxima e os índices no arranjo utilizados para a realização da soma.

A implementação do quadrado mágico, deverá gerar um arranjo quadrático que obedeça a *definição do quadrado mágico* (soma de cada linha e coluna do arranjo bidimensional deverá ter o mesmo valor, assim como a soma dos elementos da diagonal principal e secundária). Elementos não devem ser repetidos no arranjo (os números utilizados devem ser únicos). Deverá ser retornado o valor da dimensão do arranjo, valor da soma da verificação do quadrado mágico (vide definição do quadrado mágico dado anteriormente).

## 2 Implementação

As duas implementações, tal qual requerido no documento de especificação do trabalho prático, foram realizadas na linguagem de programação *C*, em um ambiente computacional *GNU/Linux* compatível com as estações de trabalho disponíveis no *CRC* (Centro de Recursos Computacionais) do Depto. de Ciência da Computação da universidade.

### 2.1 Soma Máxima

A implementação da soma máxima foi realizada tendo em mente a busca linear em arranjo unidimensional, em que durante a “passagem” do indexador de elementos do arranjo, as posições de início e fim de segmento de soma máxima são armazenados. Conforme o valor acumulado da soma máxima ocorre, as posições de início e fim de segmento são atualizadas. Dessa forma, a um custo computacional de  $O(n)$  consegue-se computar a soma máxima.

### 2.2 Quadrado Mágico

Na implementação do quadrado mágico, devido ao baixo número de quadrados mágicos requeridos e pela ausência de informação no documento que descreve o trabalho prático relativo ao modo como se dá a geração dos quadrados mágicos, optou-se pela implementação *hardcoded*. Entretanto, foi implementada a verificação que se dado um arranjo bidimensional quadrado, determina se é um quadrado mágico ou não. Por definição, um quadrado mágico deve ter a soma de cada uma de suas linhas e colunas iguais, a soma dos elementos da diagona principal, deve ser igual ao da diagona secundária e por sua vez, igual ao valor calculado da soma dos elementos de cada linha e cada coluna. Há de se observar ainda que a unicidade dos elementos deve ser respeitada (um elemento só poderá ser utilizado uma única vez). O custo computacional para verificação do quadrado mágico é  $O(n^2)$ , mesmo custo computacional para a impressão da matriz na saída padrão. Por

ser *hardcoded*, o custo computacional das matrizes quadrada é de  $O(1)$ , uma vez que os elementos são dispostos na memória em tempo de compilação.

### 3 Ambiente de desenvolvimento

O ambiente de desenvolvimento utilizado para a composição, compilação, execução e testes é dado abaixo:

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.2 LTS"
gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4)
```

Conforme poderá ser visto no arquivo `makefile`, o compilador utilizado foi o compilador *GCC C*, padrão C99, o uso da diretiva `-std=c99` especifica o padrão e versão da linguagem utilizada.

#### 3.1 Compilação

Para compilar o programa, basta ir ao diretório onde encontra-se o código fonte e através do utilitário `make`, executar o comando `$ make`. Automaticamente o utilitário fará a leitura do arquivo `makefile` o qual contém uma espécie de “receita” para a compilação do programa. O processo de compilação bem como o resultado da compilação podem ser vistos respectivamente nas figuras: 1 e 2 do apêndice: A.

#### 3.2 Execução do programa

A execução do programa no ambiente *GNU/Linux* é dada da seguinte forma:

- *soma máxima*: `$ .\somamax n {m}`,  $3 \leq n \leq 20$  e  $m \in \mathbb{Z}$ ,  $m$  representa um conjunto de números inteiros, dentro do domínio de representação de números inteiros da máquina alvo, separados por espaço. A quantidade dos números é dada pelo primeiro parâmetro  $n$ .
- *quadrado mágico*: `$ .\qmagico n`,  $3 \leq n \leq 5$ , em que  $n$  representa a dimensão do quadrado mágico a ser gerada (3x3, 4x4 ou 5x5).

No primeiro caso (soma máxima),  $n$  indica o número de argumentos, os quais correspondem aos elementos do arranjo unidimensional para cálculo da soma máxima. Cada elemento poderá ser qualquer inteiro cuja representação esteja no domínio da máquina alvo.

No segundo caso,  $n$  representa a dimensão do arranjo bidimensional de cada quadrado mágico. Tal qual proposto no enunciado, somente os arranjos quadrados 3x3, 4x4 e 5x5 são gerados.

Exemplos de execuções podem ser encontradas no apêndice: B, onde as figuras: ??, 3 e 4 exibem uma execução típica e os resultados obtidos para os três casos: *permutação com repetição*, *permutação sem repetição* e *combinação com repetição*.

#### 3.3 Avaliação de alocação/desalocação de recursos

Afim de avaliar a correta alocação e desalocação de recursos bem como vazamento de memória, a ferramenta de *software valgrind* (<http://valgrind.org/>) (SEWARD et al., 2016) foi utilizada. Os resultados obtidos inicialmente apresentavam vazamentos de memória. Após a verificação e identificação das anormalidades, as correções foram realizadas e os resultados das execuções mostraram o correto gerenciamento dos recursos utilizados ao longo das execuções dos programas.

As saídas do verificador para as implementações: *somamax* e *qmagico* (respectivamente *soma máxima* e *quadrado mágico*) são dadas abaixo:

```
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$
valgrind --tool=memcheck --leak-resolution=high --leak-check=full ./qmagico 4
==28924== Memcheck, a memory error detector
==28924== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==28924== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==28924== Command: ./qmagico 4
==28924==
n = 4, Soma = 34
12  1 14  7
13  8 11  2
 3 10  5 16
 6 15  4  9
==28924==
==28924== HEAP SUMMARY:
==28924==      in use at exit: 0 bytes in 0 blocks
==28924==    total heap usage: 2 allocs, 2 frees, 1,088 bytes allocated
==28924==
==28924== All heap blocks were freed -- no leaks are possible
==28924==
==28924== For counts of detected and suppressed errors, rerun with: -v
==28924== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$
valgrind --tool=memcheck --leak-resolution=high --leak-check=full
./somamax 10 31 -41 59 26 -53 58 97 -93 -23 104
==28962== Memcheck, a memory error detector
==28962== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==28962== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==28962== Command: ./somamax 10 31 -41 59 26 -53 58 97 -93 -23 104
==28962==
Soma: 187
Indices: 3 a 7
==28962==
==28962== HEAP SUMMARY:
==28962==      in use at exit: 0 bytes in 0 blocks
==28962==    total heap usage: 2 allocs, 2 frees, 1,064 bytes allocated
==28962==
==28962== All heap blocks were freed -- no leaks are possible
==28962==
==28962== For counts of detected and suppressed errors, rerun with: -v
==28962== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Na seção Resultados 4, há maiores informações acerca de cada execução realizada para comprovação do funcionamento de cada uma das três implementações.

Conforme esperado, não foi encontrado nenhum tipo de vazamento de memória ou falha de segmentação, confirmando assim a alocação e desalocação correta dos recursos dinâmicos utilizados ao longo da execução dos programas.

## 4 Resultados

Os resultados obtidos estão em conformidade com os exemplos dados no documento de especificação do trabalho prático: (LOUREIRO, 2017). Exemplos de execuções e resultados para cada programa: *somamax* e *qmagico* exibidos no apêndice: B; figuras: 3, e 4.

## 5 Conclusões

O trabalho prático: *Problemas Combinatórios: Soma máxima e Quadrado Mágico* possibilitou o entendimento da composição (programação) e organização (estruturação) de dois pequenos programas de computador para obtenção do valor da soma máxima em um arranjo unidimensional (cujo custo computacional da implementação foi de  $O(n)$ ) e a geração e verificação dos quadrados mágicos de dimensões 3x3, 4x4 e 5x5. Convém ressaltar que a implementação do quadrado mágico através da técnica *hardcoded* deu-se devido número e intervalo baixo dos quadrados mágicos requeridos ( $3 \leq n \leq 5$ ). Entretanto, a verificação dos quadrados mágicos foi feita de forma a ser utilizada para quadrados mágicos de quaisquer dimensão, tendo o custo computacional típico de  $O(n^2)$ . A implementação da verificação dos quadrados mágicos foi a mais trabalhosa, uma vez que envolve a verificação da soma de todas as linhas e colunas do arranjo bidimensional, além da verificação da soma da diagonal principal e secundária. Foi verificado também a unicidade dos números utilizados, dado que por definição de construção, em um quadrado mágico não pode haver número repetido.

O trabalho possibilitou uma entrada ao fascinante campo da *Combinatória*, um dos ramos da *Matemática* e da *Ciência da Computação*.

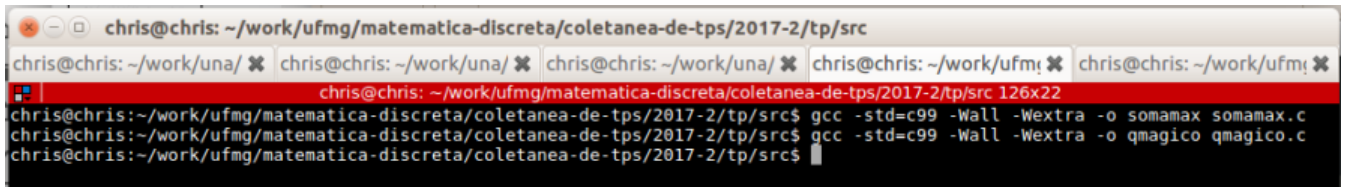
## 6 Referências Bibliográficas

LOUREIRO, A. A. F. *Trabalho Prático – Problemas Combinatórios*. 2017. [Online; acessado em 30/10/2017]. Disponível em: <[http://homepages.dcc.ufmg.br/~loureiro/md/md\\_TP.pdf](http://homepages.dcc.ufmg.br/~loureiro/md/md_TP.pdf)>.

SEWARD, J. et al. *Valgrind*. 2016. [Online; acessado em 07/10/2016]. Disponível em: <<http://valgrind.org>>.

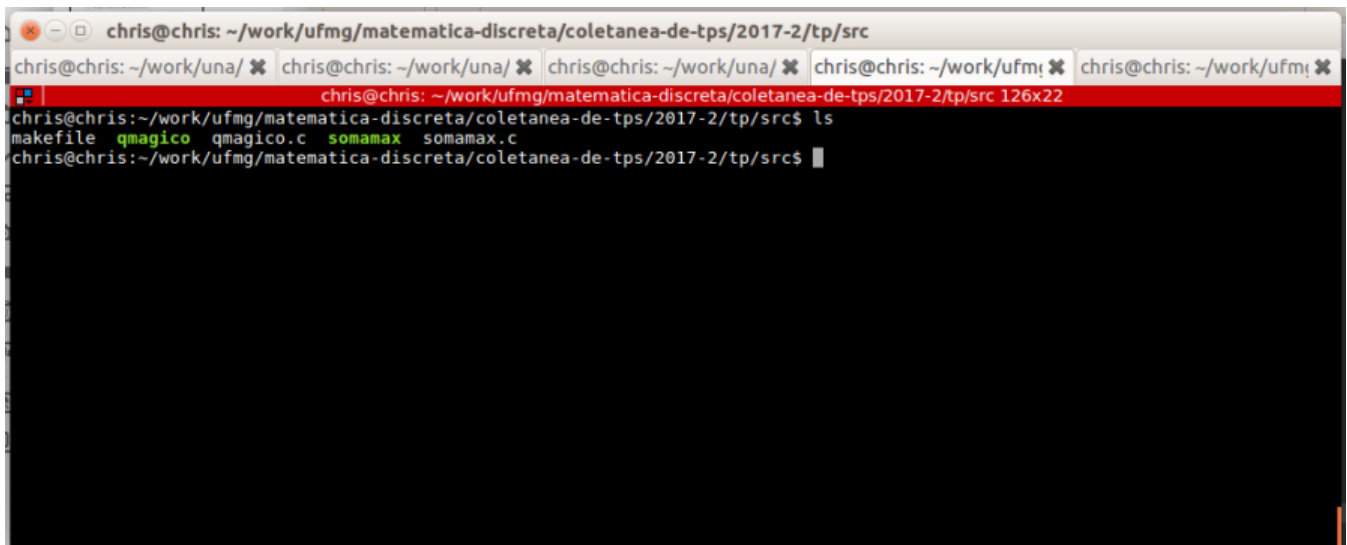
## Apêndice A Compilação dos programas

As figuras 1 e 2 abaixo exemplificam o processo de compilação e os resultados obtidos.



```
chris@chris: ~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src
chris@chris: ~/work/una/ ✖ chris@chris: ~/work/una/ ✖ chris@chris: ~/work/una/ ✖ chris@chris: ~/work/ufm ✖ chris@chris: ~/work/ufm ✖
chris@chris: ~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src 126x22
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$ gcc -std=c99 -Wall -Wextra -o somamax somamax.c
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$ gcc -std=c99 -Wall -Wextra -o qmagico qmagico.c
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$
```

Figura 1 Processo de compilação.

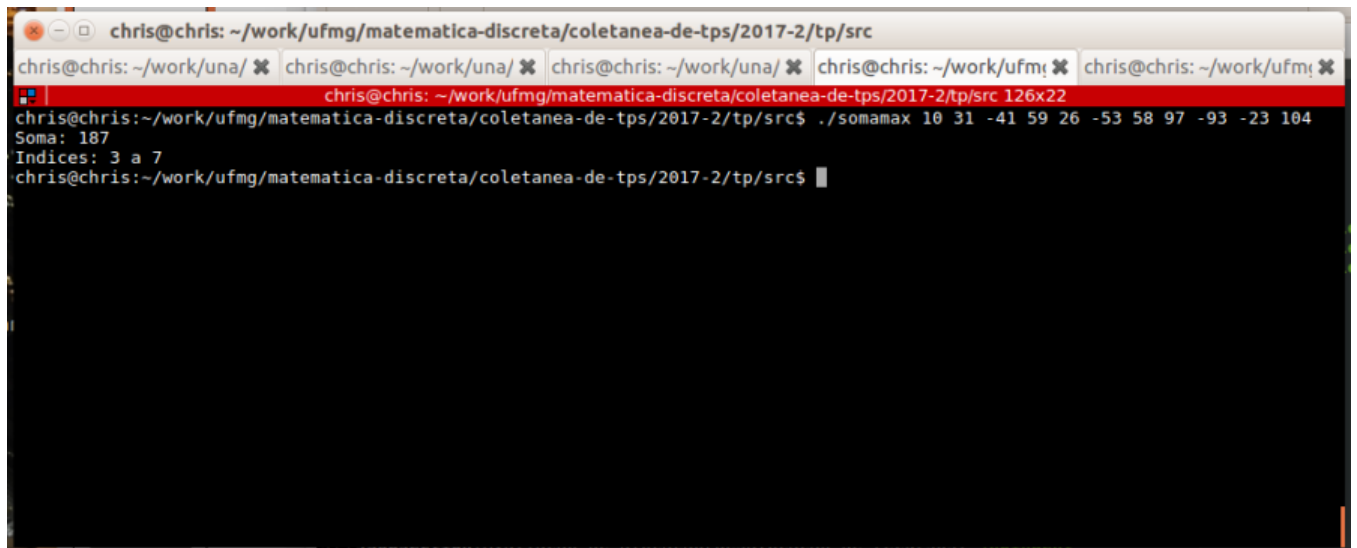


```
chris@chris: ~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src
chris@chris: ~/work/una/ ✖ chris@chris: ~/work/una/ ✖ chris@chris: ~/work/una/ ✖ chris@chris: ~/work/ufm ✖ chris@chris: ~/work/ufm ✖
chris@chris: ~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src 126x22
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$ ls
makefile qmagico qmagico.c somamax somamax.c
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$
```

Figura 2 Resultado do processo de compilação dos programas.


## Apêndice B Execução dos programas

As figuras 3 e 4 exemplificam o processo de execução do programa e exibem os resultados obtidos.



```
chris@chris: ~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src
chris@chris: ~/work/una/ ✖ chris@chris: ~/work/una/ ✖ chris@chris: ~/work/una/ ✖ chris@chris: ~/work/ufmg ✖ chris@chris: ~/work/ufmg ✖
chris@chris: ~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src 126x22
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$ ./somamax 10 31 -41 59 26 -53 58 97 -93 -23 104
Soma: 187
Indices: 3 a 7
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$
```

Figura 3 Exemplo de execução do programa *soma máxima* (*somamax*)



```
chris@chris: ~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src
chris@chris: ~/work/una/ ✖ chris@chris: ~/work/una/ ✖ chris@chris: ~/work/una/ ✖ chris@chris: ~/work/ufmg ✖ chris@chris: ~/work/ufmg ✖
chris@chris: ~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src 126x22
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$ ./qmagico 5
n = 5, Soma = 65
17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9
chris@chris:~/work/ufmg/matematica-discreta/coletanea-de-tps/2017-2/tp/src$
```

Figura 4 Exemplo de execução do programa *quadrado mágico* (*qmagico*)