

LICENCIATURA EM ENGENHARIA INFORMATICA

Trabalho prático

Matemática Discreta

REALIZADO POR:

8200613 ANDRÉ PINTO

8200615 SERGIO FELIX

8230067 ROBERTO FARIA

Índice

Introdução	2
Contextualização	2
Exercício 1 – Indução	3
Passo Base	3
Hipótese de indução	3
Exercício 2 – Rota do Românico	4
Definição, Classificação e Representação do Grafo	4
Construção da matriz de Adjacência	5
Construção da matriz de Pesos	5
Algoritmo de Dijkstra	0
Fecho Transitivo	0
Aplicação dos algoritmos	0
A) Shortest path to farthest monument	0
B) Caminho mais curto para cada par de vértices	1
C) Circuito Hamiltoniano com menor peso	3
Exercício 3 – Encriptação	4
Encriptação	4
Desencriptação	5
Conclusão	7
Bibliografia	7
Anexos	7

Introdução

Este trabalho, desenvolvido no âmbito da unidade curricular de Matemática Discreta, visa a aplicação prática de conceitos fundamentais desta unidade curricular, tais como demonstrações por indução, algoritmos de Teoria de Grafos e funções de encriptação e descriptação modular.

O propósito principal é aprofundar a compreensão teórica através de desafios concretos que simulam problemas reais, oferecendo uma experiência de aprendizagem integrada e aplicada.

Contextualização

A Matemática Discreta desempenha um papel crucial em várias áreas da Engenharia Informática e Segurança. Este trabalho, pertencente ao ano letivo de 2023/2024, propõe três desafios principais:

Demonstração por Indução ou Aplicação do Algoritmo EGV:

- Os alunos devem selecionar um exercício relevante e apresentar uma demonstração detalhada, utilizando técnicas aprendidas durante as aulas e da forma lecionada. Esta secção visa consolidar o entendimento sobre métodos formais de prova, essenciais na construção e verificação de algoritmos.

Aplicação da Teoria de Grafos na Rota do Românico:

- Com um exercício pratico inspirado na Rota do Românico, um percurso turístico cultural, o trabalho requer a utilização de conceitos de Teoria de Grafos para planejar percursos otimizados entre monumentos históricos e alojamentos. Este desafio aborda a criação de grafos, envolvendo a definição de vértices, arestas, matrizes de adjacência e custo, e a determinação de percursos mínimos baseados em diferentes critérios (distância, tempo, consumo de combustível, etc.).

Função de Encriptação e Descriptação:

- Através de uma função de encriptação modular previamente fornecida, os alunos devem encriptar e descriptar mensagens específicas, percorrendo todos os passos e demonstrando a resolução encontrada. Este exercício ilustra a aplicação prática da criptografia modular, uma área vital para a segurança da informação.

Exercício 1 – Indução

A indução matemática é um método muito utilizado, para demonstrar a veracidade de uma proposição para todos os números naturais. A indução matemática baseia-se na lógica de que se uma proposição é verdadeira para o primeiro caso e, assumindo que é verdadeira para um caso, é possível provar que também é verdadeira para o próximo caso, então a proposição é verdadeira para todos os números naturais.

O processo de indução matemática é geralmente composto por duas etapas:

- **Passo Base:** Nesta etapa, demonstra-se que a proposição é verdadeira para o menor valor da sequência, geralmente para $n=1$. Esta é a fundação sobre a qual se construirá a prova.
- **Passo Indutivo:** Assumindo que a proposição é verdadeira para um valor arbitrário $n=k$ (hipótese de indução), demonstra-se que a proposição também é verdadeira para o próximo valor $n=k+1$.

Uma vez que estas duas etapas são completadas, conclui-se que a proposição é verdadeira para todos os números naturais

Voltando ao exercício, foi pedido para selecionar aleatoriamente um exercício, que do qual, foi escolhida a seguinte preposição. A soma dos quadrados dos números ímpares naturais positivos dada pela seguinte expressão:

$$P(n): 1^2 + 3^2 + \dots + (2n-1)^2 = \frac{n(2n-1)(2n+1)}{3}, n \in \mathbb{N}^+$$

Passo Base

Será utilizado $n=1$ para o passo base. $\frac{1(2 \cdot 1 - 1)(2 \cdot 1 + 1)}{3} = 1^2 = 1$, é válido!

Hipótese de indução

Agora, através da veracidade $n=k$, assumimos também que $n=k+1$

$$1^2 + 3^2 + \dots + (2k-1)^2 = \frac{k(2k-1)(2k+1)}{3}, k \in \mathbb{N}^+$$

Passo Indutivo

Adicionamos à sucessão o valor de $k+1$, substituindo k por $k+1$. Obtendo a tese de indução.

$$1^2 + 3^2 + \dots + (2k-1)^2 + (2(k+1)-1)^2 = \frac{(k+1)(2k+1)(2k+3)}{3}, k \in \mathbb{N}^+$$

$$\frac{k(2k-1)(2k+1)}{3} + \frac{(2k+2-1)^2}{1} = \frac{k(2k-1)(2k+1) + 3(2k+1)^2}{3}$$

$$= \frac{(2k+1)(k(2k-1) + 3(2k+1))}{3} = \frac{(2k+1)(2k^2 - k + 6k + 3)}{3}$$

$$= \frac{(2k+1)(2k^2 + 5k + 3)}{3} = \frac{(2k+1)(k+1)(2k+3)}{3}$$

$C.A \frac{2k^2 + 5k + 3}{k+1} = (k+1)(2k+3)$

Como demonstramos que a fórmula é verdadeira para $n=1$ (Passo base) e que a veracidade para $n=k$ implica a veracidade para $n=k+1$ (passo indutivo), podemos concluir, pelo princípio da indução matemática, que a fórmula inicial é verdadeira para todo $n \in \mathbb{N}^+$.

Exercício 2 – Rota do Românico

A Rota do Românico abrange 58 monumentos distribuídos pelos concelhos do Vale do Sousa. O objetivo é definir um percurso que inclua 6 monumentos escolhidos pelo grupo, além de um alojamento para pernoitar, dentro da região da rota. O foco principal é aplicar os conceitos de Teoria de Grafos.

Desta forma, para a realização do exercício, foram selecionados os pontos: 4615 Hotel, Igreja de São Vicente de Sousa, Igreja do Salvador de Unhão, Igreja de Santa Maria de Airões, Igreja de São Mamede de Vila Verde, Igreja do Salvador de Avelada e por fim o Mosteiro de Santa Maria de Pombeiro. Nesta escolha tivemos em conta apenas os monumentos da Rota pertencentes a felgueiras, dispersos um pouco pela região.

Por opção, foi decidido realizar todos os cálculos em uma script Python, utilizando a biblioteca networkX. Lá, foram aplicados todos os conceitos necessários para uma posterior análise crítica do grupo.

De modo a não gerar um grafo completo, apenas consideramos as principais estradas que ligam os monumentos e os pesos foram consideradas três variáveis, distancia em quilómetros (kms), tempo em minutos (min) e consumo de combustível em quantidade de litros aos cem quilómetros (l/100kms). Nestes critérios iremos explicar mais à frente.

Definição, Classificação e Representação do Grafo



FIGURA 2 REPRESENTAÇÃO NO MAPA

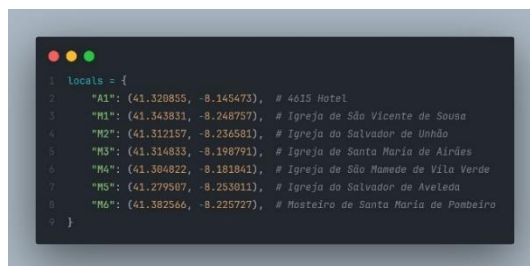


FIGURA 1 DEFINIÇÃO VÉRTICES PYTHON

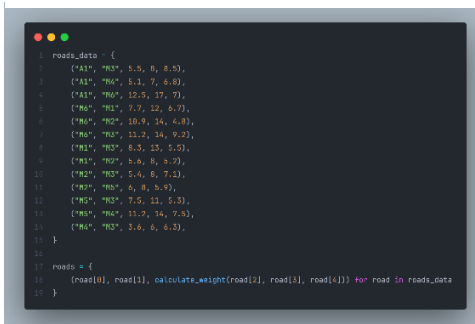


FIGURA 3 DEFINIÇÃO ARESTAS PYTHON

Após a representação do grafo, conseguimos concluir que:

- É não orientado, para representar melhor a realidade, visto que todas as estradas teriam os dois sentidos.
- Conexo, pois, de qualquer vértice, é possível chegar a todos os vértices. (Fecho transitivo).
- Trata-se de um grafo simples, pois não tem lacetes nem arestas múltiplas.

- O conjunto dos vértices é constituído por $V(g)=\{A1,M1,M2,M3,M4,M5,M6\}$ e os graus dos mesmos, respetivamente, 3,3,4,6,3,3,4.
- Conjunto das arestas constituído por, $A(g)=\{(A1,M3),(A1,M4), (A1,M6), (M3,M1), (M3,M2), (M3,M4), (M3,M5), (M3,M6), (M4,M5), (M2,M5), (M2,M6), (M1,M2), (M1,M6)\}$
- Não é Euleriano, temos mais do que dois vértices de grau ímpar.
- É Hamiltoniano, mesmo não conseguindo provar o teorema de **DIRAC** pois, por exemplo o grau do vértice A1 (grau 3), é inferior a $(n=7)/2$ e **ORE** pois, por exemplo, a soma do grau do par de vértices não adjacentes A1 (grau 3) e M5 (grau 3) é menor que $n=7$, podemos traçar um circuito hamiltoniano. Exemplo $A1 \rightarrow M4 \rightarrow M3 \rightarrow M5 \rightarrow M2 \rightarrow M1 \rightarrow M6 \rightarrow A1$

```

Dirac's Theorem:
False Vertex A1 violates Dirac's condition

Ore's Theorem:
False Pair (A1, M1) violates Ore's condition

```

FIGURA 4 VIREIFICAÇÃO DIRAC E ORE PYTHON

Construção da matriz de Adjacência

Sendo um grafo não orientado, a nossa matriz de adjacência será simétrica.

```

[0. 0. 0. 1. 1. 0. 1.]
[0. 0. 1. 1. 0. 0. 1.]
[0. 1. 0. 1. 0. 1. 1.]
[1. 1. 1. 0. 1. 1. 1.]
[1. 0. 0. 1. 0. 1. 0.]
[0. 0. 1. 1. 1. 0. 0.]
[1. 1. 1. 1. 0. 0. 0.]

```

FIGURA 5 MATRIZ ADJACENCIAS

Construção da matriz de Pesos

Para realizar este problema foi pedido para serem considerados pelo menos dois critérios para o custo. Sendo assim, o grupo considerou três critérios, distância em quilómetros (kms), tempo em minutos (min) e consumo de combustível em quantidade de litros aos cem quilómetros (l/100kms).

De forma a relacionar estes 2 critérios, foi decidido realizar a média ponderada. Atribuindo 20% à distância, 60% ao tempo de viagem e 20% ao consumo de combustível, originando apenas um valor que relaciona os três critérios através de uma função $F(x, y, z) = 0,2x + 0,6y + 0,2\left(\frac{x}{100}\right)z$. Podemos visualizar na seguinte tabela.

Aresta	Distancia	Tempo	Consumo	Peso
(A1,M3)	5,5	8	8,5	5,99
(A1,M4)	5,1	7	6,8	5,29
(A1,M6)	12,5	17	7	12,88
(M6,M1)	7,7	12	6,7	8,84
(M6,M2)	10,9	14	4,8	10,68
(M6,M3)	11,2	14	9,2	10,85
(M1,M3)	8,3	13	5,5	9,55
(M1,M2)	5,6	8	5,2	5,98
(M2,M3)	5,4	8	7,1	5,96
(M2,M5)	6	8	5,9	6,07
(M5,M3)	7,5	11	5,3	8,18
(M5,M4)	11,2	14	7,5	10,81
(M4,M3)	3,6	6	6,3	4,37

```
1 def calculate_weight(distance, time, consumption):
2     return round((distance * 0.2) + (((distance / 100) * consumption) * 0.2) + (time * 0.6), 2)
```

FIGURA 3 FUNCAO CALCULO PESO PYTHON

[0.	0.	0.	5.99	5.29	0.	12.88]
[0.	0.	5.98	9.55	0.	0.	8.84]
[0.	5.98	0.	5.96	0.	6.07	10.68]
[5.99	9.55	5.96	0.	4.37	8.18	10.85]
[5.29	0.	0.	4.37	0.	10.81	0.]
[0.	0.	6.07	8.18	10.81	0.	0.]
[12.88	8.84	10.68	10.85	0.	0.	0.]

FIGURA 7 MATRIZ DE PESOS PYTHON

Algoritmo de Dijkstra

Para a realização dos próximos exercícios é necessário previamente elaborar a tabela de Dijkstra para que seja possível saber para todos os trajetos os pesos associados. Assim será mais fácil chegar aos resultados pretendidos.

It.	v_d (M)	Mc	A	$v_i, \dots, v_d, v_j \in L(v_j)$	X e Xd	R
0	vazio	A1	{M3,M4,M6}	{M3,M4,M6} {5.99,5.29,12,88}	L(A1,M3)=5.99 L(A1,M4)=5.29 L(A1,M6)=12.88	A1,M3 A1,M4 A1,M6
1	M4	A1,M4	{M3,M5}	{M3,M5} {4,37;10,81}	L(A1,M4,M3)=10,27 L(A1,M4,M5)=16,71	A1,M4,M3 A1,M4,M5
2	M3	A1,M3	{A1,M1,M2,M4,M5,M6}	{M1,M2,M4,M5,M6} {9,55;5,96;4,37;8,18;10,85}	A1,M3,M1 = 15,54 A1,M3,M2 = 11,95 A1,M3,M4 = 10,36 A1,M3,M5 = 14,17 A1,M3,M6 = 16,84	A1,M3,M1 A1,M3,M2 A1,M3,M4 A1,M3,M5 A1,M3,M6
3	M2	A1,M3,M2	{M1,M3,M5,M6}	{M1,M3,M5,M6} {5,98;5,96;6,07;10,68}	A1,M3,M2,M1 = 17,93 A1,M3,M2,M3 = 17,91 A1,M3,M2,M5 = 18,02 A1,M3,M2,M6 = 22,85	A1,M3,M2,M1 A1,M3,M2,M3 A1,M3,M2,M5 A1,M3,M2,M6
4	M1	A1,M3,M2,M1	{M2,M3,M6}	{M2,M3,M6} {5,98;9,55;8,84}	A1,M3,M2,M1,M2 = 23,91 A1,M3,M2,M1,M3 = 27,48 A1,M3,M2,M1,M6 = 26,77	A1,M3,M2,M1,M2 A1,M3,M2,M1,M3 A1,M3,M2,M1,M6
5	M6	A1,M3,M2,M1,M6	{A1,M1,M2,M3}	{A1,M2,M3} {12,88;8,84;10,85}	A1,M3,M2,M1,M6,A1 = 39,65 A1,M3,M2,M1,M6,M1 = 35,61 A1,M3,M2,M1,M6,M3 = 37,62	A1,M3,M2,M1,M6,A1 A1,M3,M2,M1,M6,M1 A1,M3,M2,M1,M6,M3
6	M5	A1,M3,M2,M5	{M2,M3,M5}	{M2,M3,M4} {6,07;8,18;10,81}	A1,M3,M2,M5,M2 = 24,09 A1,M3,M2,M5,M3 = 26,2 A1,M3,M2,M5,M4 = 28,83	A1,M3,M2,M5,M2 A1,M3,M2,M5,M3 A1,M3,M2,M5,M4

Fecho Transitivo

O fecho transitivo, é importante ser calculado sempre que analisamos um grafo pois, a partir deste passo, podemos retirar várias informações sobre a conectividade do grafo. Se realizarmos o fecho transitivo da matriz de adjacência iremos verificar se é conexo ou fortemente conexo, e se realizar o fecho transitivo da matriz de pesos, iremos obter o peso mínimo para cada par de vértices, essencial para resolução do exercício 3.B. Esta operação foi realizada em python através da função “nx.transitive_closure()”

```
[0 0 0 1 1 0 1]
[0 0 1 1 0 0 1]
[0 1 0 1 0 1 1]
[1 1 1 0 1 1 1]
[1 0 0 1 0 1 0]
[0 0 1 1 1 0 0]
[1 1 1 1 0 0 0]
```

```
[1 1 1 1 1 1 1]
[1 1 1 1 1 1 1]
[1 1 1 1 1 1 1]
[1 1 1 1 1 1 1]
[1 1 1 1 1 1 1]
[1 1 1 1 1 1 1]
[1 1 1 1 1 1 1]
```

**FIGURA 12 FECHO
TRANSITIVO ADJACÊNCIA**

```
[ 0.    0.    0.    5.99  5.29  0.    12.88]
[ 0.    0.    5.98  9.55  0.    0.    8.84]
[ 0.    5.98  0.    5.96  0.    6.07  10.68]
[ 5.99  9.55  5.96  0.    4.37  8.18  10.85]
[ 5.29  0.    0.    4.37  0.    10.81  0.   ]
[ 0.    0.    6.07  8.18  10.81  0.    0.   ]
[12.88  8.84  10.68  10.85  0.    0.    0.   ]
```

```
[0.    15.54  11.95  5.99  5.29  14.17  12.88]
[15.54  0.    5.98  9.55  13.92  12.05  8.84]
[11.95  5.98  0.    5.96  10.33  6.07  10.68]
[ 5.99  9.55  5.96  0.    4.37  8.18  10.85]
[ 5.29  13.92  10.33  4.37  0.    10.81  15.22]
[14.17  12.05  6.07  8.18  10.81  0.    16.75]
[12.88  8.84  10.68  10.85  15.22  16.75  0.   ]
```

FIGURA 13 FECHO TRANSITIVO PESOS

Aplicação dos algoritmos

A) Shortest path to farthest monument

Utilizamos a seguinte função em python para resolver o problema. Definiu-se o ponto inicial que seria o alojamento e foi encontrado o menor caminho (A1->M3->M1) para o vértice mais distante M1 .

```
1 def shortest_path_from_accommodation_to_furthest_local(network, accommodation):
2     max_distance = 0
3     farthest_monument = None
4
5     for local in locals:
6         distance = nx.shortest_path_length(network, source=accommodation, target=local, weight='weight')
7         if distance > max_distance:
8             max_distance = distance
9             farthest_monument = local
10
11     shortest_path = nx.shortest_path(network, source=accommodation, target=farthest_monument, weight='weight')
12
13     return farthest_monument, max_distance, shortest_path
```


```
2. a)
Farthest monument: M1
Distance: 15.54
Shortest path: ['A1', 'M3', 'M1']
```

FIGURA 9 OUTPUT PYTHON

FIGURA 4 SHORTEST PATH TO FURTHEST LOCAL

B) Caminho mais curto para cada par de vértices

Este exercício foi realizado na mesma forma, utilizando a script Python e tendo encontrado o algoritmo Dijkstra anteriormente criado. Para verificar a veracidade, fomos consultar o fecho transitivo da matriz de pesos.



```
1 def shortest_path_between_every_two_locations(network):
2     shortest_paths = dict(nx.all_pairs_dijkstra_path(network, weight='weight'))
3     shortest_distances = dict(nx.all_pairs_dijkstra_path_length(network, weight='weight'))
4
5     for source in shortest_paths:
6         for target in shortest_paths[source]:
7             print(f"Shortest route from {source} to {target}:")
8             print(f"Path: {shortest_paths[source][target]}")
9             print(f"Total distance: {shortest_distances[source][target]:.2f}")
10            print()
```

FIGURA 10 MENOR DISTÂNCIA ENTRE PARES DE VERTICES

Obtemos os seguintes resultados:

Shortest route from A1 to M4:
Path: ['A1', 'M4']
Total distance: 5.29

Shortest route from A1 to M6:
Path: ['A1', 'M6']
Total distance: 12.88

Shortest route from A1 to M3:
Path: ['A1', 'M3']
Total distance: 5.99

Shortest route from A1 to M5:
Path: ['A1', 'M3', 'M5']
Total distance: 14.17

Shortest route from A1 to M2:
Path: ['A1', 'M3', 'M2']
Total distance: 11.95

Shortest route from A1 to M1:
Path: ['A1', 'M3', 'M1']
Total distance: 15.54

Shortest route from M1 to M6:
Path: ['M1', 'M6']
Total distance: 8.84

Shortest route from M1 to M3:
Path: ['M1', 'M3']
Total distance: 9.55

Shortest route from M1 to M2:
Path: ['M1', 'M2']
Total distance: 5.98

Shortest route from M1 to M5:
Path: ['M1', 'M2', 'M5']
Total distance: 12.05

Shortest route from M1 to M4:
Path: ['M1', 'M3', 'M4']
Total distance: 13.92

Shortest route from M2 to M6:
Path: ['M2', 'M6']
Total distance: 10.68

Shortest route from M2 to M3:
Path: ['M2', 'M3']
Total distance: 5.96

Shortest route from M2 to M5:
Path: ['M2', 'M5']
Total distance: 6.07

Shortest route from M2 to M4:
Path: ['M2', 'M3', 'M4']
Total distance: 10.33

Shortest route from M3 to M6:
Path: ['M3', 'M6']
Total distance: 10.85

Shortest route from M3 to M4:
Path: ['M3', 'M4']
Total distance: 4.37

Shortest route from M3 to M5:
Path: ['M3', 'M5']
Total distance: 8.18

Shortest route from M4 to M5:
Path: ['M4', 'M5']
Total distance: 10.81

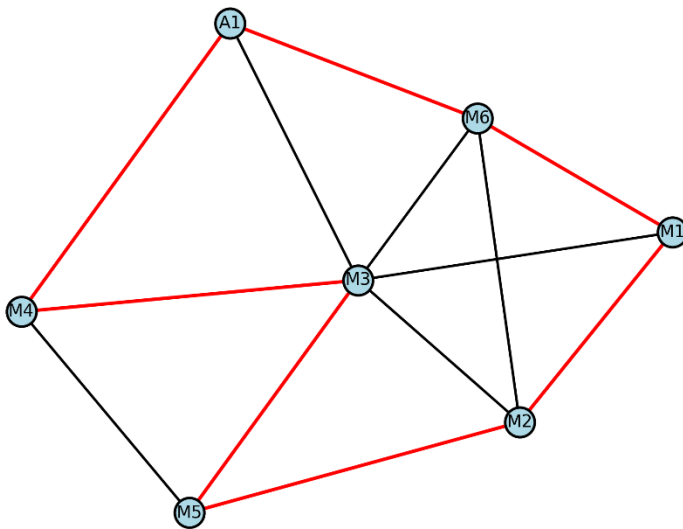
Shortest route from M4 to M6:
Path: ['M4', 'M3', 'M6']
Total distance: 15.22

Shortest route from M5 to M6:
Path: ['M5', 'M2', 'M6']
Total distance: 16.75

C) Circuito Hamiltoniano com menor peso.

Neste exercício é pedido para calcular o percurso com menor peso, com partida no alojamento A1, que passe em todos os locais apenas uma única vez e regresse ao local de partida. Para isso teremos de determinar o circuito hamiltoniano com menor peso.

Obtendo o seguinte output.



```
2. c)
Minimum cost Hamiltonian cycle starting at A1:
Path: ['A1', 'M6', 'M1', 'M2', 'M5', 'M3', 'M4', 'A1']
Total cost: 51.61
```

Exercício 3 – Encriptação

A criptografia é uma técnica essencial na segurança da informação, permitindo codificar dados para proteger sua integridade, confidencialidade e autenticidade.

Utilizada há milhares de anos, a criptografia desempenha um papel cada vez mais vital na sociedade moderna. Sem ela, a informação não poderia ser transmitida ou armazenada de forma segura e confiável, garantindo a proteção contra acessos não autorizados.

Neste contexto, foi solicitado aos alunos que aplicassem os conhecimentos adquiridos sobre criptografia ao longo do semestre. O objetivo é realizar a encriptação de duas frases pré-definidas no enunciado e, posteriormente, que o recetor consiga realizar o processo inverso, ou seja, a desencriptação.

Pressupostos:

- foi definida que a função de encriptação é $F(p) = (\beta p + 2) \bmod 29$, onde $\beta \neq 0$ é o último algarismo do número de aluno de um dos elementos do grupo.
- A seguinte tabela de conversão:

TABELA 1 CONVERSÃO CARACTERES

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	,	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

- Frase 1 “Os estudantes da ESTG: (os 4 primeiros nomes de cada um dos estudantes do grupo), sugerem uma visita a Felgueiras.”
- Frase 2 “As opções de visita incluem: Ver, Escapar, Saborear, Inspirar, Meditar, Encontrar e Viver!”

Encriptação

Numa primeira fase, foi reorganizada toda a informação do exercício e contruída as frases respeitando os pressupostos.

- Sendo $\beta \neq 0$ e o último algarismo de um dos alunos dos elementos do grupo, foi definido então $\beta = 3$, onde 3 é o último algarismo do aluno 8200613.
- Frase 1:** “Os estudantes da ESTG: Andre Filipe Martins Pinto, Sergio Luis Lopes Felix e Roberto Cristiano Martins Faria, sugerem uma visita a Felgueiras.”
- Frase 2:** “As opções de visita incluem: Ver, Escapar, Saborear, Inspirar, Meditar, Encontrar e Viver!”
- $F(p) = (3p + 2) \bmod 29$**

Para realizar a encriptação foram executados três passos. Inicialmente a conversão de todos os caracteres para o número inteiro correspondente através da tabela dada no enunciado, **Tabela 1**, de seguida aplicando a fórmula da cifra para cada caractere e no final, voltar a converter o valor inteiro cifrado para uma letra do abecedário, através da mesma tabela de conversão fornecida.

Para tal, foi elaborado um Excel (**Anexo 1**), para que, de forma rápida e visualmente compreensível, realizar estas conversões.

Frase 1

Após ter aplicado todos os passos na frase 1, chegou-se à frase encriptada seguinte:

“P, O,BELCMBO, LC O,BU: CMLYO R.G.SO JCYB.M, S.MBPZ ,OYU.P GE., GPSO, ROG.N O YPFOYBP IY.,B.CMP JCYB.M, RCY.CZ ,EUOYOJ EJC H.,.BC C ROGUEO.YC,W”

Frase 2

Após ter aplicado todos os passos na frase 2, chegou-se à frase encriptada seguinte:

“C, PSIPO, LO H.,.BC .MIGEOJ: HOYZ O,ICSCYZ ,CFPYOCYZ .M,S.YCYZ JOL.BCYZ OMIPMBYCY O H.HOY!”

Descriptação

Tal como pedido no enunciado, será necessário ainda realizar a operação inversa, descriptação das duas frases.

Para conseguir realizar este passo, será necessário encontrar a função inversa da encriptação $F(p) = (3p + 2) \bmod 29$.

Função inversa

Para encontrar a função inversa de $F(p) = (3p + 2) \bmod 29$, precisamos encontrar uma função $f^{-1}(Q)$ tal que $F(f^{-1}(Q)) = Q \bmod 29$. Garantindo assim a bijetividade da função.

$$Q \equiv 3P + 2 \bmod 29$$

$$Q - 2 \equiv 3P \bmod 29$$

Seguidamente será necessário encontrar o inverso multiplicativo $3 \bmod 29$. Para tal é necessário verificar se admite um inverso multiplicativo. Apenas assim, nestes casos em particular, é possível garantir exatamente uma solução. Como estamos a utilizar o mod 29 e 29 é um número primo, então o inverso multiplicativo modular **mod 29** existe para qualquer $\beta \neq 0$ que não seja um múltiplo de 29. Sendo $\beta = 3$, não múltiplo de 29, existe um inverso multiplicativo.

$$\beta \cdot P \equiv 1 \bmod 29 \rightarrow 3 \cdot 10 \equiv 1 \bmod 29$$

É necessário um número P tal que $3P \equiv 1 \bmod 29$, em que 1 é o elemento identidade da multiplicação. Sendo assim $P=10$.

Desta forma chegamos à função inversa. $f^{-1}(Q) = 10(Q - 2) \bmod 29$

Para realizar a descriptação foram executados três passos. Inicialmente a conversão de todos os caracteres para o número inteiro correspondente através da tabela dada no enunciado, **Tabela 1**, de seguida aplicando a fórmula inversa para cada caractere e no final, voltar a converter o valor inteiro cifrado para uma letra do abecedário, através da mesma tabela de conversão fornecida.

Para tal, foi elaborado um Excel (**Anexo 1**), para que, de forma rápida e visualmente compreensível, realizar estas conversões.

No Final, foram obtidas as frases originais.

Início	Num	Formula Encrypt	Encrypt	Texto Encrypt	Formula Decrypt	Decrypt	Fim
O	14	$F(14)=(3*14+2) \bmod 29 = 15$	15	P	$F^{-1}(15)=10(15-2) \bmod 29 = 14$	14	O
s	18	$F(18)=(3*18+2) \bmod 29 = 27$	27	,	$F^{-1}(27)=10(27-2) \bmod 29 = 18$	18	S
e	4	$F(4)=(3*4+2) \bmod 29 = 14$	14	O	$F^{-1}(14)=10(14-2) \bmod 29 = 4$	4	E
s	18	$F(18)=(3*18+2) \bmod 29 = 27$	27	,	$F^{-1}(27)=10(27-2) \bmod 29 = 18$	18	S
t	19	$F(19)=(3*19+2) \bmod 29 = 1$	1	B	$F^{-1}(1)=10(1-2) \bmod 29 = 19$	19	T
u	20	$F(20)=(3*20+2) \bmod 29 = 4$	4	E	$F^{-1}(4)=10(4-2) \bmod 29 = 20$	20	U
d	3	$F(3)=(3*3+2) \bmod 29 = 11$	11	L	$F^{-1}(11)=10(11-2) \bmod 29 = 3$	3	D
a	0	$F(0)=(3*0+2) \bmod 29 = 2$	2	C	$F^{-1}(2)=10(2-2) \bmod 29 = 0$	0	A
n	13	$F(13)=(3*13+2) \bmod 29 = 12$	12	M	$F^{-1}(12)=10(12-2) \bmod 29 = 13$	13	N
t	19	$F(19)=(3*19+2) \bmod 29 = 1$	1	B	$F^{-1}(1)=10(1-2) \bmod 29 = 19$	19	T
e	4	$F(4)=(3*4+2) \bmod 29 = 14$	14	O	$F^{-1}(14)=10(14-2) \bmod 29 = 4$	4	E
s	18	$F(18)=(3*18+2) \bmod 29 = 27$	27	,	$F^{-1}(27)=10(27-2) \bmod 29 = 18$	18	S
d	3	$F(3)=(3*3+2) \bmod 29 = 11$	11	L	$F^{-1}(11)=10(11-2) \bmod 29 = 3$	3	D

FIGURA 11 EXEMPLO DO ANEXO 1 ENCRIPTAÇÃO_FRASE1

Conclusão

O trabalho desenvolvido no âmbito da unidade curricular de Matemática Discreta demonstrou, de forma prática e integrada, a aplicação de conceitos teóricos essenciais, como a demonstração por indução, algoritmos de Teoria de Grafos e funções de encriptação modular. Cada exercício abordado forneceu uma oportunidade valiosa para consolidar e aplicar o conhecimento adquirido.

Exercício 1 – Indução:

Através da indução matemática, verificou-se a veracidade de uma fórmula para a soma dos quadrados dos números ímpares naturais. Este exercício destacou a importância dos métodos formais de prova, imprescindíveis para a construção e verificação de algoritmos.

Exercício 2 – Rota do Românico:

A aplicação da Teoria de Grafos na Rota do Românico permitiu explorar conceitos de grafos, como a definição de vértices e arestas, e a construção de matrizes de adjacência e peso. A utilização do algoritmo de Dijkstra para determinar os percursos mais curtos entre monumentos mostrou a relevância destes conceitos para a resolução de problemas práticos, como a otimização de rotas.

Exercício 3 – Encriptação:

A implementação de funções de encriptação e desencriptação modular enfatizou a importância da criptografia na segurança da informação. Através da encriptação de mensagens específicas e da subsequente desencriptação, tão necessário nos tempos de hoje.

Em conclusão, o trabalho proporcionou uma experiência de aprendizagem aplicada, onde a teoria encontrou a prática. Os desafios apresentados não só reforçaram o entendimento dos conceitos de Matemática Discreta, mas também demonstraram a sua aplicabilidade em contextos reais. Através deste trabalho, ficou claro que a Matemática Discreta é um pilar fundamental para diversas áreas tecnológicas e científicas, contribuindo de forma significativa para o desenvolvimento de soluções inovadoras e eficientes.

Bibliografia

Rosen, K. H. (2012). Discrete Mathematics and Its Applications (7ª ed.). McGraw-Hill.

Omni Calculator. (n.d.). Calculadora Inverso Multiplicativo Modular. Recuperado em 06/2024 de <https://www.omnicalculator.com/pt/matematica/calculadora-inverso-multiplicativo-modular>

GraphOnline. (n.d.). Recuperado em 06/2024, de <https://graphonline.ru/pt/>

Google. (n.d.). Google Maps. Recuperado em 06/2024, de <https://www.google.pt/maps>

Anexos

- Anexo 1 – Criptografia.xlsx
- Anexo 2 – main.py