Segunda Avaliação

Programação I (INF15927) - UFES
15 de Dezembro de 2022

Instruções

- A prova é **individual**. Todas as questões serão testadas e plágio não será tolerado
- Guarde o seu celular na sua mochila ou algo similar. O uso de celular durante a prova não será tolerado
- Não custa relembrar, mas todas as questões devem ser desenvolvidas na **linguagem de programação C**
- Organização/indentação e boas práticas de programação é critério de avaliação. Use nome de variáveis de forma inteligente
- **Escolhas de tipo de dados e inclusão de bibliotecas é por sua conta**
- Cada questão possui 10 casos de testes, na qual 4 são públicos e estão disponíveis para vocês. Siga a risca as instruções de cada uma das questões para que o teste automático não falhe
- Seu código será compilado usando o comando `gcc <nome_arquivo.c> -o <nome_arquivo> -lm`
- Seu código será executado em um sistema operacional Linux com o comando `./<nome_arquivo> < <entrada_n> > <saida_n>`
- **Todas as questões tem o mesmo peso**
- **Importante:** a prova encontra-se dentro da pasta `Documents` com o nome `prova2_progI.zip`. Ao extrair o .zip, será criado uma pasta com nome `prova`. Renomeia essa pasta usando seu `<nome_sobrenome>` (exemplo: `andre_pacheco`. Nessa pasta você já encontra uma pasta para cada questão que já contém os testes públicos. **Solucione cada questão dentro da sua respectiva pasta.**

Questão 1: Em uma biblioteca, cada livro possui um identificador único que é representado por um número inteiro de 1 até `N`. Porém, se existir cópias deste livro, elas possui o mesmo identificador. É sempre desejável ter mais de uma cópia de cada livro para que várias pessoas possam usufruir do mesmo. Sendo assim, os funcionários da biblioteca receberam uma lista de identificadores de livros para identificar quais não possuem cópias para solicitar a compra de um novo exemplar. Sua função é criar um programa que identifique quais livros da lista não tem cópia e retorne para o funcionário solicitar a compra.

Funcionamento do programa: seu programa deve receber como entrada um número inteiro que representa a quantidade de livros da lista e a lista propriamente dita. Nessa lista, ele deve identificar quais livros não possuem cópias. Seguindo a ordem crescente de identificadores. **Dica:** você não precisa usar algoritmo de ordenação (a não ser que você queira).

Exemplo de funcionamento: imagine que o programa recebeu a seguinte lista com 8 livros: $[1, 2, 3, 4, 3, 2, 1, 7]$. Seu programa deve retornar que os livros 4 e 7 (nessa ordem) não possuem cópias e necessita ser solicitado a compra dos mesmos.

Limites:

- $1 \leq N \leq 1000$
- Todos os identificadores de livros são números inteiros
- O tamanho máximo da lista de livros é 10000

Padrão de entrada e saída:

- Entrada:
  `<quantidade>`
  `<lista de livros>`
- Saída:
  `<lista de livros únicos>`
  Se não houver livros, você deve imprimir `NENHUM`

Nome do arquivo da questão: `livros.c`

Questão 2: Rede neural artificial

Uma rede neural artificial é um algoritmo muito poderoso capaz de abordar diversos problemas relevantes do nosso dia a dia. Apesar de parecer intimidador, no fundo, uma rede neural artificial nada mais é do que uma série de multiplicações de matrizes. Uma rede de apenas uma camada funciona da seguinte maneira: ela possui uma entrada $\mathbf{x}$ que normalmente é um vetor de tamanho M; esse vetor é multiplicado por uma matriz de pesos W; o resultado dessa multiplicação é o vetor de saída $\mathbf{y}$ que possui tamanho N. Logo, a matriz W possui ordem $M \times N$. Matematicamente, temos: $\mathbf{y} = \mathbf{x} \times W$. Sua tarefa é implementar uma programa que calcule o valor de $\mathbf{y}$ dado $\mathbf{x}$ e W.

Funcionamento do programa: seu programa receberá como entrada todos os valores de $\mathbf{x}$ e os valores inteiros M e N. A matriz de peso deve ser definida de maneira aleatória. Para isso, você também vai receber um valor inteiro s que deve ser usado como semente para um gerador de números aleatórios. Feito isso, você deve calcular $\mathbf{y}$ através a multiplicação $\mathbf{x} \times W$.

Exemplo de funcionamento: imagine que você receba como entrada $M = 4$, $N = 5$, $\mathbf{x} = [1, 3, 7, 4]$ e $s = 42$. Usando a semente igual a 42, seu programa deve gerar a seguinte matriz W:

$$W = \begin{bmatrix} 21 & 71 & 37 & 39 & 59 \\ 74 & 78 & 16 & 69 & 5 \\ 21 & 49 & 4 & 18 & 56 \\ 0 & 59 & 100 & 49 & 83 \end{bmatrix}$$

Por fim, o resultado da multiplicação $\mathbf{x} \times W$ gera $y = [390, 884, 513, 568]$. **Importante**: para que a geração de números aleatórios replique a matriz W acima, o seu loop externo deve percorrer as linhas e o interno as colunas.

Limites:

- $1 \leq N, M \leq 50$
- s é um valor inteiro
- $\mathbf{x}$ é composto por números inteiros
- W é composto por números inteiros no intervalo fechado de [0, 100]

Padrão de entrada e saída:

- Entrada:
  `<M> <N> <s>`
  `<x>`
- Saída:
  `<y>`

Nome do arquivo da questão: `multiplicacao.c`

Questão 3: Estimativa de alagamento

Problemas com alagamentos são recorrentes em cidades brasileiras. Para tentar tomar tomar medidas preventivas, a prefeitura de Vila Velha encomendou um software para estimar o quanto um foco de alagamento se espalha em um bairro. Esse software recebe uma foto aérea do bairro com um foco inicial de alagamento e deve estimar o quanto essa enchente vai se espalhar ao longo das próximas horas. Sua tarefa é implementar tal software.

Funcionamento do programa: seu programa deve receber uma matriz que representa a uma imagem aérea do foco inicial do alagamento. Nessa matriz, cada posição recebe um valor. O valor 0 representa um terreno seco e o valor 1 representa o terreno alagado. **A cada hora o alagamento expande uma unidade para cima, para baixo, para esquerda e para direita** (não existe expansão para a diagonal). Seu programa deve receber como entrada a quantidade de `horas` que se deseja estimar e retornar a quantidade de terreno seco e alagado naquela hora. Para ler a matriz, você receberá quantidade de `linhas` e `colunas` da mesma. **Observação**: só é possível expandir o alagamento para dentro das dimensões da matriz. Se a evolução estoura o número de linhas ou colunas, esse é o ponto máximo que ele pode chegar.

Exemplo de funcionamento: na Figura 2 é ilustrado um exemplo de funcionamento. A matriz de início é o que seu programa recebe. Cada célula representa uma posição da matriz. Em branco (que representa o valor 0) está o terreno seco. Já em azul (que representa o valor

1) está o terreno alagado. Neste exemplo, pretende-se estimar a condição do bairro em 2h. Sendo assim, na condição inicial, existem 3 terrenos alagados e 61 secos. Após uma hora, são 10 alagados e 54 secos. E por fim, a resposta da estimativa, 21 alagados e 43 secos.

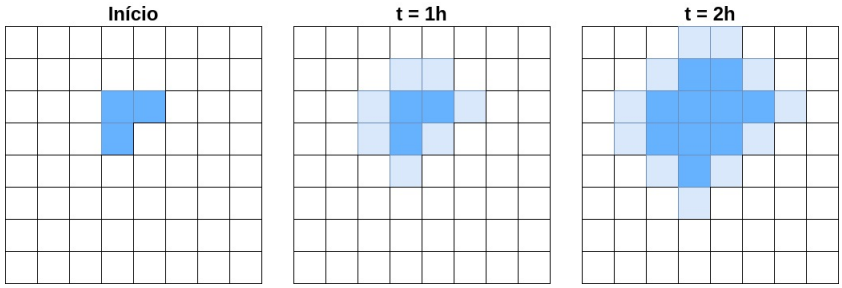

Figura 1: Exemplo da expansão do alagamento ao longo do tempo. Para facilitar a visualização, a expansão do alagamento a cada hora é ilustrada com um azul mais claro.

Limites:

- $0 \leq$ `linhas`, `colunas` ≤ 100
- $1 \leq$ `horas` ≤ 8
- As posições da matriz assumem 1 ou 0

Padrão de entrada e saída:

- Entrada:
  `<horas>`
  `<linhas> <colunas>`
  `<matriz>`
- Saída:
  `<quantidade de terreno alagado> <quantidade de terreno seco>`

Nome do arquivo da questão: `alagamento.c`

Questão 4: Validação de endereço de IP

Endereço IP é um um identificador único que representa um dispositivo na internet ou em uma rede local. Um endereço de IP deve ser dividido em blocos de quatro números separados por ponto (`.`), por exemplo, `127.222.33.15`. Seguindo este padrão, um endereço de IP é válido se ele cumprir os seguintes requisitos:

1. Todos dígitos de cada bloco são números

2. Cada número deve possuir 8 bits, ou seja, estar no intervalo de 0 a 255
3. O IP não pode começar com zero

Sabendo disso, faça um programa que leia um grupo de endereços de IP e retorne se os mesmos são válidos.

Funcionamento do programa: seu programa deve receber como entrada um numero inteiro N que representa a quantidade de IPs. Cada IP é representado por uma string. Seu programa deve retornar se um dado IP é válido ou não.

Exemplo de funcionamento:

- Exemplos de IP válido:
  - `127.1.1.0`
  - `255.21.51.12`
  - `128.128.128.128`
- Exemplos de IP inválido:
  - `500.1.1.0` (viola regra 2)
  - `128.b.128.a` (viola regra 1)
  - `0.128.128.128` (viola regra 3)

Limites:

- Os digitos de um `IP` podem ser caracteres alfa-numéricos (`0-9`, `a-z`, `A-Z`) sem caracter especial ou acento.
- O tamanho máximo de um candidato a `IP` é 50 caracteres
- $1 \leq N \leq 100$

Padrão de entrada e saída:

- Entrada:
  `<N>`
  `<ip_1>`
  ...
  `<ip_N>`
- Saída:
  `<valido>` ou `<invalido>`
  ...
  `<valido>` ou `<invalido>`

Nome do arquivo da questão: `ip.c`

Material suplementar

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciicharstable.com

Figura 2: Tabela ASCII