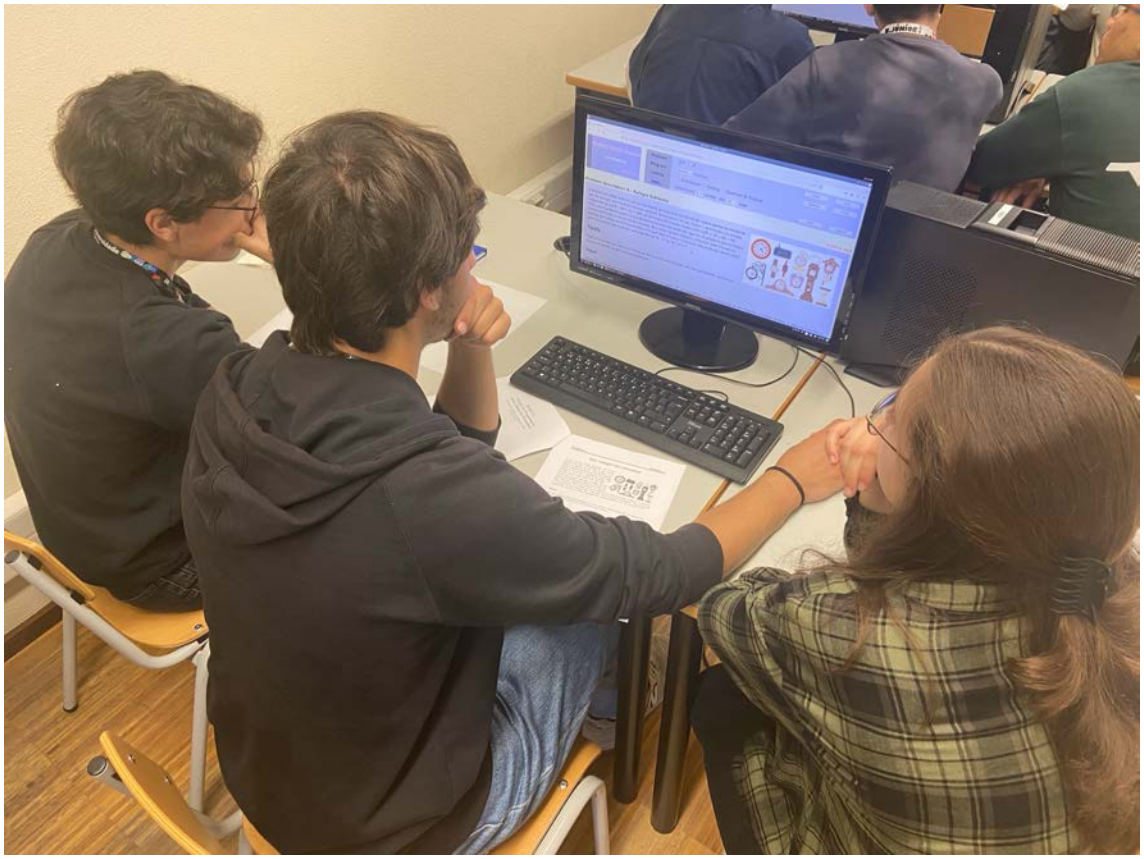
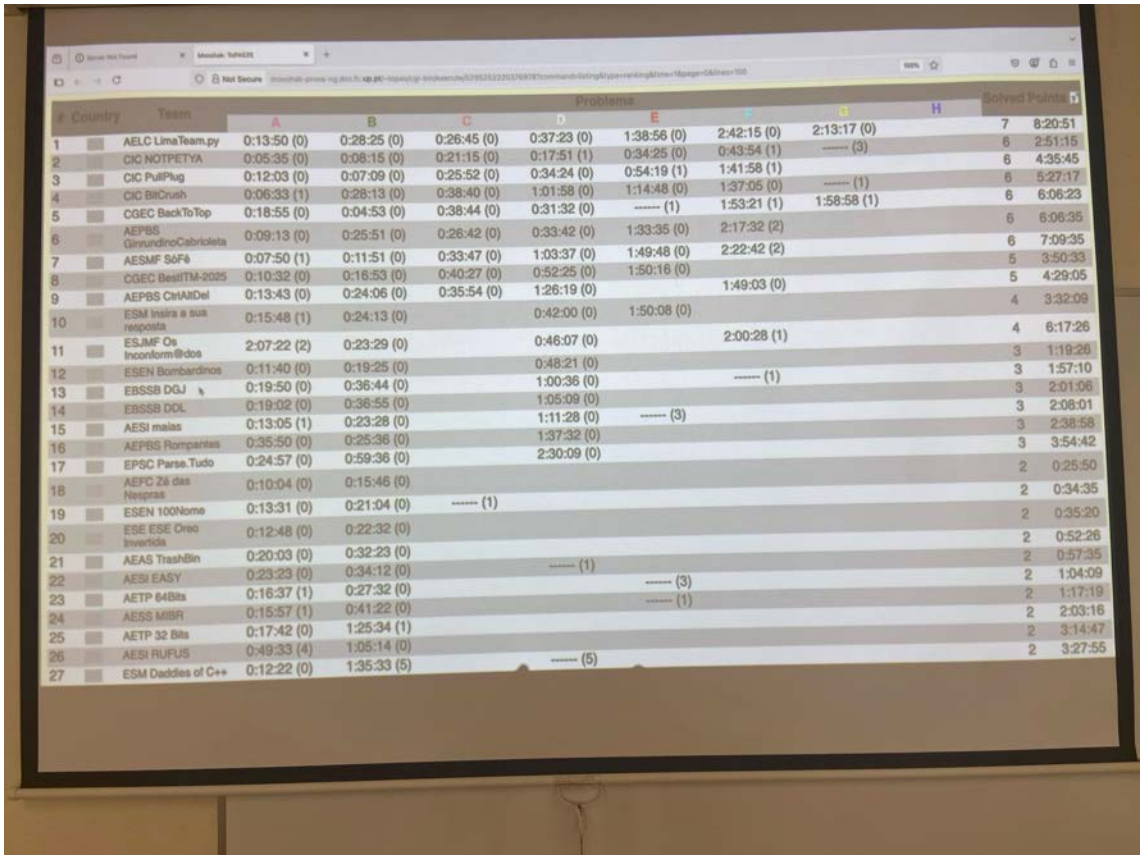
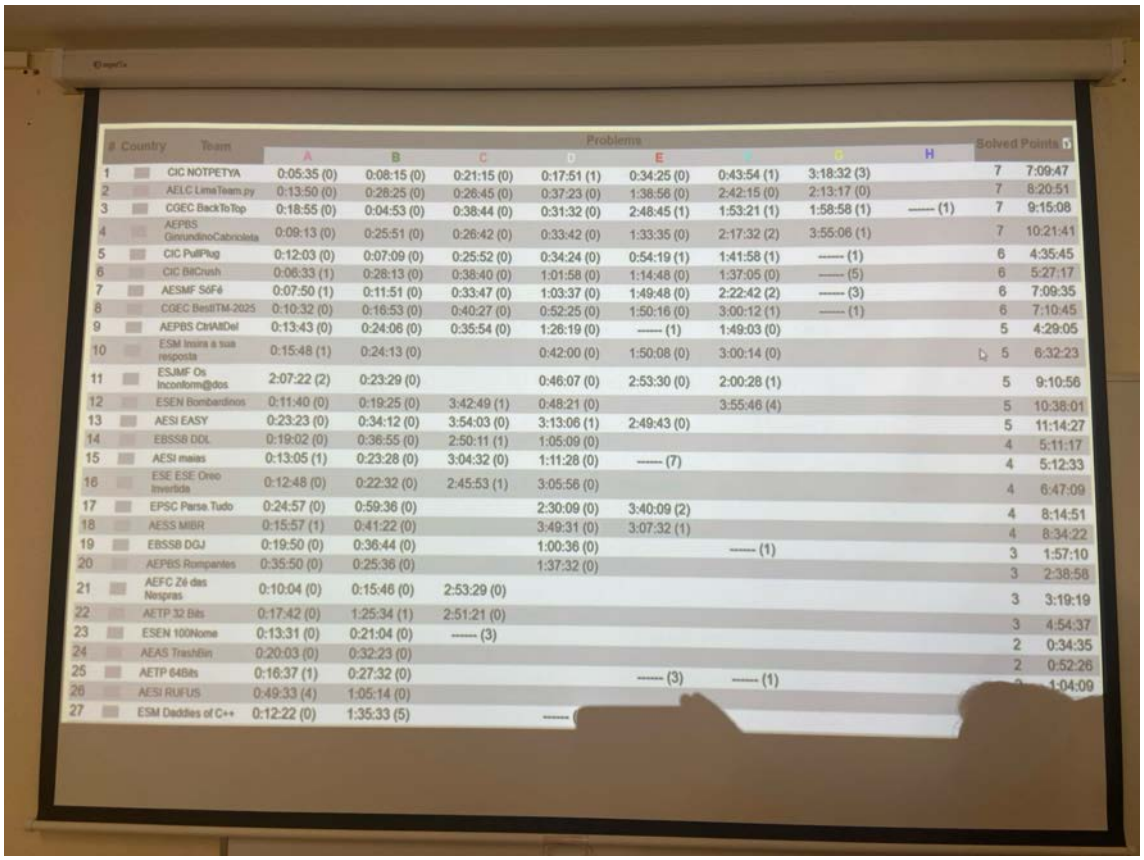

LimaTeam.py



#	Country	Team	Problems								Solved	Points
			A	B	C	D	E	F	G	H		
1		CIC NOTPETYA	0:05:35 (0)	0:08:15 (0)	0:21:15 (0)	0:17:51 (1)	0:34:25 (0)	0:43:54 (1)			6	2:51:
2		CIC PullPlug	0:12:03 (0)	0:07:09 (0)	0:25:52 (0)	0:34:24 (0)	------ (1)				4	1:19:
3		CGEC BackToTop	0:18:55 (0)	0:04:53 (0)	0:38:44 (0)	0:31:32 (0)					4	1:34:
4		AEPBS GinrundinoCabrioleta	0:09:13 (0)	0:25:51 (0)	0:26:42 (0)	0:33:42 (0)					4	1:35:
5		AELC LimaTeam.py	0:13:50 (0)	0:28:25 (0)	0:26:45 (0)	0:37:23 (0)					4	1:48:
6		CGEC BestITM-2025	0:10:32 (0)	0:16:53 (0)	0:40:27 (0)	0:52:25 (0)					4	2:00:
7		AESMF SóFé	0:07:50 (1)	0:11:51 (0)	0:33:47 (0)						3	1:13:
8		AEPBS CtrlAltDel	0:13:43 (0)	0:24:06 (0)	0:35:54 (0)						3	1:13:
9		ESEN Bombardinos	0:11:40 (0)	0:19:25 (0)		0:48:21 (0)					3	1:19:
10		CIC BitCrush	0:06:33 (1)	0:28:13 (0)	0:38:40 (0)						3	1:33:
11		ESM Insira a sua resposta	0:15:48 (1)	0:24:13 (0)		0:42:00 (0)					3	1:42:
12		AEFC Zé das Nespras	0:10:04 (0)	0:15:46 (0)							2	0:25:
13		ESEN 100Nome	0:13:31 (0)	0:21:04 (0)	------ (1)						2	0:34:
14		ESE ESE Oreo Invertida	0:12:48 (0)	0:22:32 (0)							2	0:35:
15		AEAS TrashBin	0:20:03 (0)	0:32:23 (0)							2	0:52:
16		EBSSB DDL	0:19:02 (0)	0:36:55 (0)							2	0:55:
17		AESI maias	0:13:05 (1)	0:23:28 (0)							2	0:56:
18		EBSSB DGJ	0:19:50 (0)	0:36:44 (0)							2	0:56:
19		AESI EASY	0:23:23 (0)	0:34:12 (0)							2	0:57:
20		AEPBS Rompantes	0:35:50 (0)	0:25:36 (0)							2	1:01:
21		AETP 64Bits	0:16:37 (1)	0:27:32 (0)							2	1:04:
22		ESJMF Os Inconform@dos	------ (2)	0:23:29 (0)		0:46:07 (0)					2	1:09:
23		AESS MIBR	0:15:57 (1)	0:41:22 (0)							2	1:17:
24		ESM Daddies of C++	0:12:22 (0)	------ (3)							1	0:12:
25		AETP 32 Bits	0:17:42 (0)	------ (1)							1	0:17:
26		EPSC Parse.Tudo	0:24:57 (0)								1	0:24:
27		ESE ESE Simfadin	------ (3)	0:43:11 (0)							1	0:43:
		AEAS										

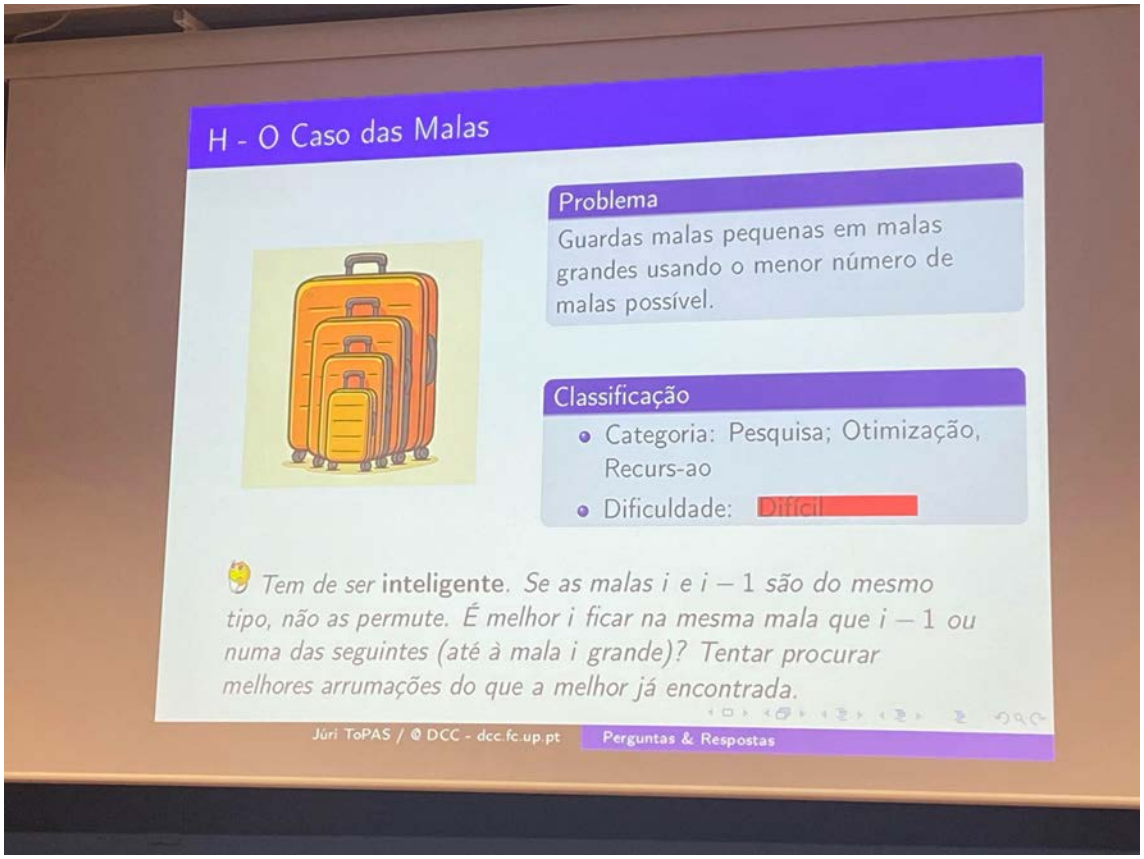


#	Country	Team	Problems								Solved	Points
			A	B	C	D	E	F	G	H		
1		AELC LimaTeam.py	0:13:50 (0)	0:28:25 (0)	0:26:45 (0)	0:37:23 (0)	1:38:56 (0)	2:42:15 (0)	2:13:17 (0)		7	8:20:51
2		CIC NOTPETYA	0:05:35 (0)	0:08:15 (0)	0:21:15 (0)	0:17:51 (1)	0:34:25 (0)	0:43:54 (1)	------ (3)		6	2:51:15
3		CIC PullPlug	0:12:03 (0)	0:07:09 (0)	0:25:52 (0)	0:34:24 (0)	0:54:19 (1)	1:41:58 (1)			6	4:35:45
4		CIC BitCrush	0:06:33 (1)	0:28:13 (0)	0:38:40 (0)	1:01:58 (0)	1:14:48 (0)	1:37:05 (0)	------ (1)		6	5:27:17
5		CGEC BackToTop	0:18:55 (0)	0:04:53 (0)	0:38:44 (0)	0:31:32 (0)	------ (1)	1:53:21 (1)	1:58:58 (1)		6	6:06:23
6		AEPBS GinrundinoCabrioleta	0:09:13 (0)	0:25:51 (0)	0:26:42 (0)	0:33:42 (0)	1:33:35 (0)	2:17:32 (2)			6	6:06:35
7		AESMF SóFé	0:07:50 (1)	0:11:51 (0)	0:33:47 (0)	1:03:37 (0)	1:49:48 (0)	2:22:42 (2)			6	7:09:35
8		CGEC BestITM-2025	0:10:32 (0)	0:16:53 (0)	0:40:27 (0)	0:52:25 (0)	1:50:16 (0)				5	3:50:33
9		AEPBS CtrlAltDel	0:13:43 (0)	0:24:06 (0)	0:35:54 (0)	1:26:19 (0)		1:49:03 (0)			5	4:29:05
10		ESM Insira a sua resposta	0:15:48 (1)	0:24:13 (0)		0:42:00 (0)	1:50:08 (0)				4	3:32:09
11		ESJMF Os Inconform@dos	2:07:22 (2)	0:23:29 (0)		0:46:07 (0)		2:00:28 (1)			4	6:17:26
12		ESEN Bombardinos	0:11:40 (0)	0:19:25 (0)		0:48:21 (0)					3	1:19:26
13		EBSSB DGJ	0:19:50 (0)	0:36:44 (0)		1:00:36 (0)		------ (1)			3	1:57:10
14		EBSSB DDL	0:19:02 (0)	0:36:55 (0)		1:05:09 (0)					3	2:01:06
15		AESI maias	0:13:05 (1)	0:23:28 (0)		1:11:28 (0)	------ (3)				3	2:08:01
16		AEPBS Rompantes	0:35:50 (0)	0:25:36 (0)		1:37:32 (0)					3	2:38:58
17		EPSC Parse.Tudo	0:24:57 (0)	0:59:36 (0)		2:30:09 (0)					3	3:54:42
18		AEFC Zé das Nespras	0:10:04 (0)	0:15:46 (0)							2	0:25:50
19		ESEN 100Nome	0:13:31 (0)	0:21:04 (0)	------ (1)						2	0:34:35
20		ESE ESE Oreo Invertida	0:12:48 (0)	0:22:32 (0)							2	0:35:20
21		AEAS TrashBin	0:20:03 (0)	0:32:23 (0)							2	0:52:26
22		AESI EASY	0:23:23 (0)	0:34:12 (0)		------ (1)					2	0:57:35
23		AETP 64Bits	0:16:37 (1)	0:27:32 (0)			------ (3)				2	1:04:09
24		AESS MIBR	0:15:57 (1)	0:41:22 (0)			------ (1)				2	1:17:19
25		AETP 32 Bits	0:17:42 (0)	1:25:34 (1)							2	2:03:16
26		AESI RUFUS	0:49:33 (4)	1:05:14 (0)							2	3:14:47
27		ESM Daddies of C++	0:12:22 (0)	1:35:33 (5)		------ (5)					2	3:27:55



#	Country	Team	Problems								Solved	Points
			A	B	C	D	E	F	D	H		
1		CIC NOTPETYA	0:05:35 (0)	0:08:15 (0)	0:21:15 (0)	0:17:51 (1)	0:34:25 (0)	0:43:54 (1)	3:18:32 (3)		7	7:09:47
2		AELC LimaTeam.py	0:13:50 (0)	0:28:25 (0)	0:26:45 (0)	0:37:23 (0)	1:38:56 (0)	2:42:15 (0)	2:13:17 (0)		7	8:20:51
3		CGEC BackToTop	0:18:55 (0)	0:04:53 (0)	0:38:44 (0)	0:31:32 (0)	2:48:45 (1)	1:53:21 (1)	1:58:58 (1)	------ (1)	7	9:15:08
4		AEPBS GirirundinoCabricoleta	0:09:13 (0)	0:25:51 (0)	0:26:42 (0)	0:33:42 (0)	1:33:35 (0)	2:17:32 (2)	3:55:06 (1)		7	10:21:41
5		CIC PullPlug	0:12:03 (0)	0:07:09 (0)	0:25:52 (0)	0:34:24 (0)	0:54:19 (1)	1:41:58 (1)	------ (1)		6	4:35:45
6		CIC BitCrush	0:06:33 (1)	0:28:13 (0)	0:38:40 (0)	1:01:58 (0)	1:14:48 (0)	1:37:05 (0)	------ (5)		6	5:27:17
7		AESMF SóFé	0:07:50 (1)	0:11:51 (0)	0:33:47 (0)	1:03:37 (0)	1:49:48 (0)	2:22:42 (2)	------ (3)		6	7:09:35
8		CGEC BestITM-2025	0:10:32 (0)	0:16:53 (0)	0:40:27 (0)	0:52:25 (0)	1:50:16 (0)	3:00:12 (1)	------ (1)		6	7:10:45
9		AEPBS CtrlAltDel	0:13:43 (0)	0:24:06 (0)	0:35:54 (0)	1:26:19 (0)	------ (1)	1:49:03 (0)			5	4:29:05
10		ESM Insira a sua resposta	0:15:48 (1)	0:24:13 (0)		0:42:00 (0)	1:50:08 (0)	3:00:14 (0)			5	6:32:23
11		ESJMF Os Inconform@dos	2:07:22 (2)	0:23:29 (0)		0:46:07 (0)	2:53:30 (0)	2:00:28 (1)			5	9:10:56
12		ESEN Bombardinos	0:11:40 (0)	0:19:25 (0)	3:42:49 (1)	0:48:21 (0)		3:55:46 (4)			5	10:38:01
13		AESI EASY	0:23:23 (0)	0:34:12 (0)	3:54:03 (0)	3:13:06 (1)	2:49:43 (0)				5	11:14:27
14		EBSSB DDL	0:19:02 (0)	0:36:55 (0)	2:50:11 (1)	1:05:09 (0)					4	5:11:17
15		AESI maias	0:13:05 (1)	0:23:28 (0)	3:04:32 (0)	1:11:28 (0)	------ (7)				4	5:12:33
16		ESE ESE Oreo Invertida	0:12:48 (0)	0:22:32 (0)	2:45:53 (1)	3:05:56 (0)					4	6:47:09
17		EPSC Parse.Tudo	0:24:57 (0)	0:59:36 (0)		2:30:09 (0)	3:40:09 (2)				4	8:14:51
18		AESS MIBR	0:15:57 (1)	0:41:22 (0)		3:49:31 (0)	3:07:32 (1)				4	8:34:22
19		EBSSB DGJ	0:19:50 (0)	0:36:44 (0)		1:00:36 (0)		------ (1)			3	1:57:10
20		AEPBS Rompantes	0:35:50 (0)	0:25:36 (0)		1:37:32 (0)					3	2:38:58
21		AEFC Zé das Nespras	0:10:04 (0)	0:15:46 (0)	2:53:29 (0)						3	3:19:19
22		AETP 32 Bits	0:17:42 (0)	1:25:34 (1)	2:51:21 (0)						3	4:54:37
23		ESEN 100Nome	0:13:31 (0)	0:21:04 (0)	------ (3)						2	0:34:35
24		AEAS TrashBin	0:20:03 (0)	0:32:23 (0)							2	0:52:26
25		AETP 64Bits	0:16:37 (1)	0:27:32 (0)			------ (3)	------ (1)				1:04:09
26		AESI RUFUS	0:49:33 (4)	1:05:14 (0)								
27		ESM Daddies of C++	0:12:22 (0)	1:35:33 (5)		------						

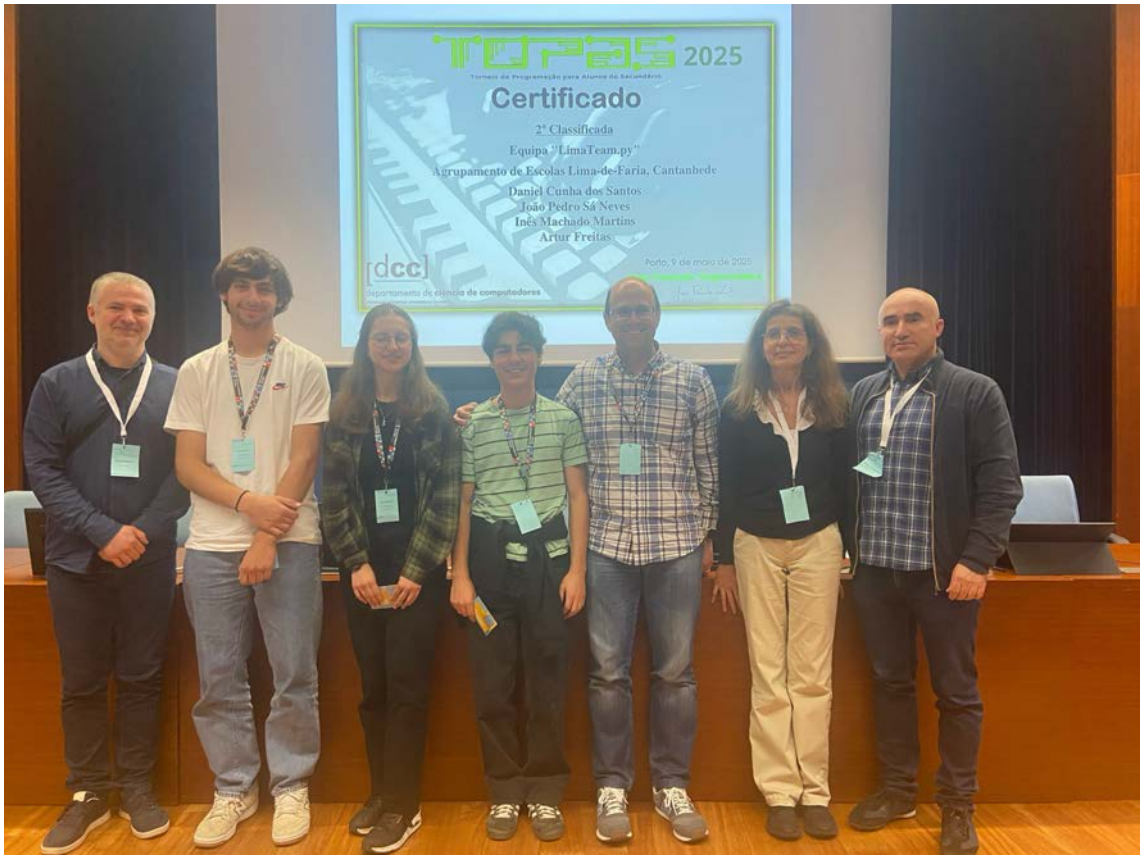

Universidade Júnior
Universidade Júnior


H - O Caso das Malas
Problema
Guardas malas pequenas em malas grandes usando o menor número de malas possível.
Classificação
Categoria: Pesquisa; Otimização, Recurs-ao
Dificuldade: Difícil
Tem de ser inteligente. Se as malas i e i − 1 são do mesmo tipo, não as permute. É melhor i ficar na mesma mala que i − 1 ou numa das seguintes (até à mala i grande)? Tentar procurar melhores arrumações do que a melhor já encontrada.
Júri ToPAS / @ DCC - dcc.fc.up.pt
Perguntas & Respostas

H - O Caso das malas

```
def otima(c,i,N,usadas,malas,frac,bsf,sol):
    if i == N: return usadas
    if usadas == bsf: return bsf
    if i == 0 or c[i] != c[i-1]: j = 0
    else: j = sol[i-1]
    while j <= usadas:
        if malas[j] >= c[i]:
            malas[j] -= c[i];  sol[i] = j
            if j < usadas:
                nsol = otima(c,i+1,N,usadas,malas,frac,bsf,sol)
            else:
                if usadas+1 < bsf:
                    nsol = otima(c,i+1,N,usadas+1,malas,frac,bsf,sol)
                else:  nsol = bsf   # não pode melhorar
            if nsol < bsf:
                bsf = nsol
                if bsf == frac:  return bsf
            malas[j] += c[i]
        j += 1
    return bsf
```

Júri ToPAS / © DCC - dcc.fc.up.pt Perguntas & Respostas




23
CANTANHEDE
sabores de
terra e mar

Voluntários

Os Bancos Alimentares Contra a Fome organizam campanhas semestrais de angariação de alimentos em supermercados e hipermercados. Cada campanha envolve muitos voluntários: uns vão para os supermercados e os hipermercados recolher os donativos, enquanto os outros vão para o armazém separar e embalar os produtos recolhidos. Nenhum voluntário pode ir para dois locais diferentes (ou seja, ninguém recolhe bens em sítios distintos e ninguém recolhe, separa e embala produtos).

Image by brgfx on Freepik

Umas semanas antes de cada campanha, os organizadores perguntam-se se já têm voluntários em número suficiente ou se têm de envidar esforços para recrutar mais alguns. Sabendo:

- em quantos supermercados e em quantos hipermercados haverá recolha de alimentos,
- quantos voluntários são necessários no armazém, em cada supermercado e em cada hipermercado, e
- quantos voluntários já estão assegurados,

consegues calcular quantos voluntários faltam?

Suponhamos que a campanha irá decorrer em 300 supermercados e em 120 hipermercados, e que tem de haver 200 voluntários no armazém, 7 voluntários em cada supermercado e 50 voluntários em cada hipermercado. Portanto, são necessários 8300 voluntários (porque $200 + 300 \times 7 + 120 \times 50 = 8300$). Se já houvesse 8050 voluntários, faltariam 250. Se já houvesse 8345 voluntários, não faltaria nenhum.

Tarefa

Escreva um programa que indique quantos voluntários ainda falta recrutar, sabendo o número de supermercados e o número de hipermercados onde a campanha vai decorrer, quantos voluntários são necessários no armazém, em cada supermercado e em cada hipermercado, e quantos voluntários já estão assegurados.

Input

O input tem três linhas. A primeira linha tem dois inteiros, S e H, que indicam, respetivamente, em quantos supermercados e em quantos hipermercados haverá recolha de alimentos. A segunda linha tem três inteiros, A, M e N, que denotam, respetivamente, quantos voluntários são necessários no armazém, em cada supermercado e em cada hipermercado. A terceira linha tem um inteiro, V, que representa quantos voluntários já estão assegurados.

Ahoy!

Ahoy marujos! Vamos juntos embarcar numa grande aventura, *por mares nunca dantes navegados*! Na distante ilha de Sapot, lá no Mar das Caraíbas, está enterrado um valioso tesouro, perdido desde a trágica expedição do saudoso Barba Negra. Está na hora de o recuperar!

O Barba Negra, minutos antes de naufragar, conseguiu desenhar num mapa da ilha o caminho para o tesouro. Mapa esse que já chegou à nossa posse, Aye! Mas a tormenta era tanta que o Barba Negra teve pouco tempo para verificar o rascunho e com grande probabilidade o mapa pode não levar ao tesouro, mas sim às perigosas armadilhas de Sapot. Aaaarrrrgggghhhh!

A vossa participação nesta viagem é importantíssima! Se vos mostrarmos o mapa, ainda antes da partida, conseguem-nos dizer se o caminho traçado nos leva ao tesouro ou diretos a uma das armadilhas? Se for o último caso, nem zarpamos!

A ilha de Sapot é um perfeito retângulo. Por exemplo, se o mapa deixado pelo Barba Negra for o seguinte:

```
E S A
S S N
A X N
A A A
```

e se assumirmos que atracamos na ilha no canto superior esquerdo, chegamos ao tesouro. A explicação é a seguinte:

1. Ao chegarmos à ilha, o mapa manda-nos em direção a este, já que lemos a letra E.
2. De seguida, continuamos para sul (letra S).
3. Depois, vamos novamente para sul (letra S).
4. E, finalmente, chegamos ao tesouro! Como sempre, a letra X marca o local!

As letras N, S, E e O indicam, respetivamente, que devemos continuar para norte, sul, este e oeste. O local do tesouro está assinalado com a letra X e as várias armadilhas estão assinaladas com a letra A.

Tarefa

Escreva um programa que, dada a dimensão da ilha de Sapot e o mapa do tesouro, indica se a expedição conseguirá chegar ao tesouro ou se irá cair numa das armadilhas. A caça ao tesouro começa sempre pelo ponto mais a norte e a oeste da ilha, isto é, pelo canto superior esquerdo do mapa.

Gata Traquina

A Beatriz tem uma gata muito traquina: se vir um computador portátil aberto na mesa da sala, vai logo caminhar sobre o teclado, introduzindo caracteres aleatórios em qualquer documento aberto!

Na realidade, talvez não sejam mesmo aleatórios: a Beatriz reparou que o seu movimento é muito previsível. A gata caminha sempre da esquerda para a direita do teclado e alterna pressionando teclas de duas linhas diferentes, deixando uma tecla de intervalo em cada linha. Assim, conhecendo o teclado, que está representado na Figura 1, e sabendo apenas as duas primeiras teclas pressionadas, é possível prever todo o texto introduzido.

Figura 1: Teclado

Por exemplo, se as primeiras duas teclas forem Q e Z, por esta ordem, a sequência de caracteres introduzidos será Q, Z, E, C, T, B, U, M e O (como se ilustra na Figura 2).

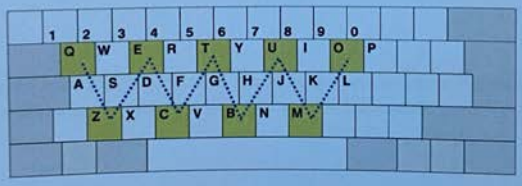


Figura 2: Sequência produzida quando as primeiras teclas pressionadas são Q e Z

Quando acabam as teclas de uma das linhas, a gata continua, pressionando apenas as teclas da outra linha e deixando na mesma uma tecla de intervalo. Por exemplo, se a primeira tecla pressionada for F e a segunda for 1, o texto introduzido será F1H3K579.

Tarefa

Escreva um programa que, dadas as duas primeiras teclas pressionadas, determina o texto produzido pela gata. Assuma que a gata apenas pressiona teclas mostradas na Figura 1, que correspondem aos dígitos de 0 a 9 e às letras de A a Z, em maiúsculas e sem acentos nem cedilhas.

Percurso de Bicicleta

Dois amigos resolveram realizar uma viagem de bicicleta de Lisboa até Santarém, num único dia. O percurso é constituído por 7 troços:

1. Lisboa – Santa Iria de Azóia;
2. Santa Iria de Azóia – Forte da Casa;
3. Forte da Casa – Vila Franca de Xira;
4. Vila Franca de Xira – Carregado;
5. Carregado – Azambuja;
6. Azambuja – Cartaxo;
7. Cartaxo – Santarém.

O final de cada troço é um *local de controlo.*

Os amigos têm de registar os instantes em que partem de Lisboa e em que passam nos locais de controlo. Assuma que, em qualquer um destes locais (partida ou de controlo), os instantes registados pelos dois amigos nunca são iguais. Os amigos também combinaram que só poderiam ultrapassar-se uma única vez em cada troço, ou seja, se o amigo A ultrapassasse o amigo B num dado troço, então B já não poderia ultrapassar A nesse troço.

No final da viagem, os amigos querem saber em que troços do percurso houve ultrapassagens e qual deles ultrapassou o outro, bem como quem chegou primeiro a Santarém.

Tarefa

Escreva um programa que, dados os nomes dos amigos e os instantes registados por eles, indica as ultrapassagens realizadas e quem chegou primeiro a Santarém.

Input

Cada uma das duas primeiras linhas do input tem o nome de um amigo, que é uma sequência não vazia de (no máximo, 50) caracteres. Os dois nomes, N_1 e N_2, são distintos.

A terceira linha tem quatro inteiros, H_{01}, M_{01}, H_{02} e M_{02}, que representam os instantes em que os amigos partiram de Lisboa: o amigo N_1 partiu às H_{01} horas e M_{01} minutos e o amigo N_2 partiu às H_{02} horas e M_{02} minutos. Cada uma das restantes sete linhas tem quatro inteiros, H_{i1}, M_{i1}, H_{i2} e M_{i2}, que representam os instantes em que os amigos chegaram ao fim do i-ésimo troço do percurso (com $i = 1, \ldots, 7$). Assuma que, em cada uma destas oito linhas, os dois instantes especificados são diferentes.

A Dinastia dos Clubes Secretos

A República de Xiangjião rege-se por leis e regras que não lembram a ninguém. Tem de se ser amigo de quem decide, e quem não é amigo, bem... amigo não é.

Décadas deste regime instituíram formas peculiares de se viver em sociedade. Por exemplo, se o João quiser fazer negócio com o Pedro, é bom que seja amigo dele de uma forma ou de outra ou então têm de se tornar amigos. Senão, não há negócio.

Amigo do meu amigo, meu amigo é. Logo, posso estar junto de uma pessoa sem saber que é minha amiga, porque esta amizade não é direta. Mas então, como é que se sabe?

Nesta república, popularizou-se o uso de cumprimentos secretos. Se João e Pedro se tornarem amigos, combinam um cumprimento secreto, o qual é um inteiro, que será usado pelos seus amigos também. Assim, formam-se clãs na República de Xiangjião e cada clã tem o seu cumprimento secreto único. Quando dois membros de clãs diferentes se tornam amigos, pelas circunstâncias da vida, os dois clãs juntam-se e, segundo as regras da república, adotam como cumprimento o maior dos dois cumprimentos. Os cidadãos desta república bananeira também se zangam com alguma facilidade e as consequências são duras: se dois membros de um clã se zangam, os ânimos exaltam-se tanto que todos quebram as amizades, ficando novamente isolados, sem amigos!

Afinal, há ou não há negócio? Esse é o seu desafio: determinar se os negócios são possíveis.

Tarefa

Faça um programa que processe uma sequência de ações e, para cada tentativa de negócio, escreva YES, se houver forma de negociar (porque os cidadãos são amigos), ou NO, se não houver forma de negociar (porque não são amigos). Por cada ação da forma "`C` i", o programa deverá indicar qual é o cumprimento do cidadão identificado por i.

Se houver N cidadãos em Xiangjião, os cidadãos são identificados pelos inteiros de 0 a $N-1$. Existem quatro tipos de ações, sendo i e j inteiros distintos que representam cidadãos:

`A` i j	O cidadão i (que não é amigo de j) torna-se amigo do cidadão j;
`D` i j	O cidadão i quer fazer negócio, vulgo *Deal*, com o cidadão j;
`Z` i j	Os cidadãos i e j zangam-se; se forem amigos, desfazem-se todas as amizades do clã;
`C` i	O cidadão i indica qual é o cumprimento secreto do seu clã.

No início, não há amizades feitas e cada cidadão tem o seu próprio identificador como cumprimento secreto, ou seja, o cumprimento do cidadão 0 é 0, o cumprimento do cidadão 1 é 1, etc. Assim, por exemplo, se o cidadão 3 tem cumprimento 5 (porque a vida o levou a ser amigo do cidadão 5) e o cidadão 6 tem cumprimento 7 (por ser amigo de 7), então, pela ação `A 3 6`, os cidadãos 3 e 6 tornam-se amigos e o cumprimento acordado será 7. Todos os amigos de 3 e todos os amigos de 6 usarão esse cumprimento, pois assim ditam as leis de Xiangjião.

O Caso das Malas

Em janeiro de 2025, foi notícia um deputado ter sido constituído arguido por suspeitas de furtar bagagens em aeroportos. Dizem que, na casa de banho, escondia as malas furtadas dentro de outras de maior dimensão.

Não queremos resolver um caso de polícia, mas perceber como esconder N malas pequenas em B malas grandes, utilizando o menor número possível das grandes, sem analisar orientações e re-orientações para as encaixar. As malas grandes são identificadas por inteiros consecutivos, a partir de 1, assim como as pequenas. As malas grandes têm todas capacidade C. A mala i pequena ocupa c_i unidades de capacidade, para $1 \leq i \leq N$, sendo $C \geq c_1 \geq c_2 \geq \ldots \geq c_N > 0$, e todos inteiros. O número t de tipos de malas é igual ao número de valores c_i distintos.

Se desfizermos malas, isto é, se pudermos distribuir c_i por várias malas grandes, para conseguir que as malas grandes usadas, exceto talvez uma delas, fiquem totalmente cheias, quantas malas usaria uma tal **arrumação fracionária ótima**? É fácil calcular! E, sem desfazer malas, quantas malas grandes requer uma **arrumação ótima** (não fracionária)? De certo, se t e N forem grandes, até o melhor programa pode demorar um tempo excessivo para responder. Para nos despacharmos, podemos tentar uma **arrumação apressada** em que, simplesmente, colocamos a mala i pequena na primeira grande em que ainda couber, depois de termos escondido as malas numeradas de 1 a $i-1$. Com sorte, tal arrumação é tão boa como a ótima ou, até, como a ótima fracionária, que é sempre a que requer menos malas! Se a fracionária requerer mais do que B malas, é óbvio que a apressada também falha assim como qualquer alternativa, pois B malas não bastariam nunca.

Tarefa

Escreva um programa que, dados inteiros, N, C, B, K, L e c_i, com $i = 1, \ldots, N$, determina a **arrumação apressada**, assumindo que não existem mais do que B malas grandes. Se a arrumação apressada falhar, por requerer mais do que B malas, indica a primeira mala i que não consegue arrumar, escrevendo "`Apressada falha para mala` i". Caso falhe ou suceda, calcula quantas malas requer uma **arrumação fracionária ótima**, para ver se vale a pena tentar procurar uma **arrumação ótima** (não fracionária) ou se, até, já a tem.

Assim, deve calcular uma **arrumação ótima** só se $N \leq L$, $t \leq K$ e não for óbvio que a arrumação apressada é ótima, sendo K e L os dois limites dados. Para cada escolha para as malas 1 a $i-1$, a mala i pode ser colocada numa qualquer mala grande onde ainda caiba. Para cada alternativa, deve procurar completar a arrumação das restantes de forma ótima. Para não sucumbir por exaustão, tem de ser **inteligente**. Se $c_i = c_{i-1}$, isto é, se as malas i e $i-1$ são do mesmo tipo, não as permute, devendo ver se é melhor i ficar na mesma mala que $i-1$ ou numa das seguintes (até à mala i grande). E deve tentar procurar arrumações (não fracionárias) que usem menos malas do que a melhor arrumação já encontrada. Não vale a pena tentar completar arrumações que não seriam melhores.