Ano lectivo 2021-2022

[21173] Introdução à Programação

E-fólio A

1.

Epaminondas é um empresário que julga ter descoberto uma solução para um problema complexo. O problema ocorre numa situação em que se tem um conjunto de N strings, todas de tamanho K. Pretende-se encontrar uma string que esteja o mais perto possível de todas as N strings. A distância entre duas strings é o número de letras distintas. Considere K=10 podendo existir strings inválidas até 20 caracteres, e para valor máximo de N considere o valor 50.

Como o algoritmo deve ser mantido em completo segredo antes da publicação, Epaminondas exige que não divulgue nem comente nada do que lhe for solicitado nem do que fizer no âmbito deste projeto.

Epaminondas divide o projeto em quatro fases. Na primeira fase pretende que leia uma string, e retorne 1 se a string tiver 10 caracteres entre 'A' e 'Z', e 0 caso contrário.

Para clarificar o que é pedido, elaborou uma tabela de casos de teste:

Entrada	Saída
ADCFGH	0
acdqcdghjo	0
ABCDEGHSCD	1
YHFVXWRHVD	1
ABCDEGhSCD	0
VXWRHVD	0
1234567890	0
FKVCOKRMGH	1
XOFLRNCQSD	1
XOFLRNCQIII	0

2.

Epaminondas pretende que nesta segunda fase, leia um conjunto de strings, uma por linha, guardando apenas as strings válidas. No final deve retornar o número de strings válidas, seguida das respetivas strings.

Para a entrada de dados:

```
AIDQGDUGBK
QZLXIIUKYE
XDNNTNNRNY
IpKWACYRTU
HXWRDDGJET
PAKZLBVC
LUOWHDDRKR
OVRREQTOEAC
```

O programa deve retornar:

```
5
AIDQGDUGBK
QZLXIIUKYE
XDNNTNNRNY
HXWRDDGJET
LUOWHDDRKR
```

Nota: caso não consiga resolver esta alínea, pode resolver para metade da cotação, mostrando as strings válidas e só depois o número de strings válidas. Nesta situação, apenas os caso de teste ímpares irão ser validados.

3.

Chegando à terceira fase, Epaminondas está bastante satisfeito com os resultados até agora encontrados, e está em condições para entrar no problema propriamente dito. Pretende que leia os dados da mesma forma que na segunda fase, mas que faça a seguinte normalização:

1. processar a primeira letra de todas as N strings introduzidas;
2. de entre as primeiras letras das N strings, determinar a letra mais frequente, a segunda mais frequente, e assim por diante (em caso de empate conta a ordem alfabética);
3. à letra mais frequente, trocar pela letra 'A', à segunda letra mais frequente, trocar pela letra 'B' e assim sucessivamente;
4. repetir os passos 1 a 3 para a segunda letra, terceira letra, e assim sucessivamente até à última letra.

Exemplificando com o primeiro exemplo da segunda fase:

Para a entrada de dados:

```
AIDQGDUGBK
QZLXIIUKYE
XDNNTNNRNY
IpKWACYRTU
HXWRDDGJET
PAKZLBVC
LUOWHDDRKR
OVRREQTOEAC
```

O programa retorna na segunda fase:

```
5
AIDQGDUGBK
QZLXIIUKYE
XDNNTNNRNY
HXWRDDGJET
LUOWHDDRKR
```

Pretende-se que nesta fase faça o seguinte:

1. Primeiras letras das 5 strings: "AQXHL"
2. Sendo a frequência igual, a ordem de frequência é atribuída por ordem alfabética: A, H, L, Q, X
3. Troca de letras: A>>A, H>>B, L>>C, Q>>D, X>>E
4. repetir para as restantes 9 letras. Vamos saltar para a sexta letra, que tem frequências distintas:
    1. Sextas letras: "DINDD"
    2. Ordem considerando a frequência: D, I, N
    3. Troca de letras: D>>A, I>>B, N>>C

No final o programa deve retornar:

```
5
ABABBAABAB
DEBEDBADEA
EACAECDADE
BDECAACCBD
CCDDCABACC
```

Nota: caso não consiga resolver esta alínea completamente, pode resolver para metade da cotação, retornando a string com a letra mais frequente em cada uma das 10 posições. No caso do exemplo, retornaria ADDNDDURBE, sendo a resposta validada nos casos de teste ímpares.

Epaminondas está preocupado porque não está seguro que você consiga implementar o algoritmo solicitado, pelo que deixa-lhe umas sugestões (branco sobre branco, selecionar para ver):

comece por contar quantas letras existem de cada tipo. Tem de ter um contador para a letra 'A', outro para a letra 'B' e assim sucessivamente até à letra 'Z'.
para ordenar, pondere reutilizar/adaptar algoritmos de ordenação que tenha feito nas atividades formativas. Assim poderá obter uma ordem de letras por frequência.
repare que no passo 3, precisa de saber que letra deve colocar para cada uma das letras. Não é o mesmo que ter as letras ordenadas por frequência, precisa ainda de converter essa ordem num mapa de modo a associar cada letra à sua letra correspondente

4

Epaminondas ficou surpreendido com o código da terceira fase, e está agora em condições de avançar para a quarta e última fase. Nesta fase vai-se implementar o seu algoritmo secreto para localizar a string cuja distância maior para todas as N strings, é a menor possível. Para tal tem de começar por implementar a distância, com a seguinte definição:

A distância da string A para a string B, ambas com tamanho K, é o número de letras distintas em cada posição.

Exemplo, distância entre ABCDEABCDE e AAAAABBBBB é precisamente 8. Apenas a letra na primeira posição e na sexta posição é que são iguais (a negrito e vermelho). Notar que o valor da distância será sempre entre 0 e K, uma vez que todas as letras podem ser distintas (nesse caso retorna K), como todas as letras podem ser iguais (retornando 0).

Para encontrar a string cuja distância maior para todas as N strings, é a menor possível, Epaminondas idealizou o seguinte algoritmo:

Iniciar a string de teste atual com AAAAAAAAAA Calcular a distância para as N strings, e identificar as M strings que geram a maior distância Calcular a letra mais frequente, nas M strings, por cada posição (apenas uma só a letra mais >frequente por cada posição, em caso de empate, utilizar a ordem alfabética) Trocar na string de teste atual, as letras pelas mais frequentes, mas apenas para posições cuja frequência seja superior ou igual a M/4 Caso a string de teste seja alterada no passo anterior, e o número de iterações seja inferior a 10, passar para o passo 2. Calcular a distância para as N strings e terminar.

O programa deve mostrar simplesmente a string de teste, juntamente com a distância para as N strings.

No caso dado da terceira fase, o output seria o seguinte:

```
ABABBAABAB
DEBEDBADEA
EACAECDADE
BDECAACCBD
CCDDCABACC
```

Neste caso, a maior distância para a string inicial AAAAAAAAAA é 8, em M=3 strings que têm apenas duas letras A (a verde). Assim, as letras mais frequentes, considerando apenas as strings a verde, são as seguintes:

```
BCBCAAAABA
```

Notar que apenas na sexta letra a letra A ocorre duas vezes, sendo os restantes resultados apenas por ser a letra mais baixa alfabeticamente que ocorre nas strings em causa.

A string de teste atual, atendendo a que M=3 e 3/4 aceitaria frequência 1, é toda a string, BCBCAAAABA.

A sequência de strings neste exemplo continua da seguinte forma:

```
EACAECDADE;
ABABAAABAA;
EACAECDADE;
ABABAAABAA;
...
10 EACAECDADE
```