MARATONA DE PROGRAMAÇÃO

InterFatecs

Taquaritinga - 2017

Final - 26/08/2017

Caderno de Soluções

Patrocínio:

Realização:

www.interfatecs.com.br

6ª MARATONA DE PROGRAMAÇÃO
InterFatecs2017

Problema A
Frequência

Arquivo fonte: frequencia.{ c | cpp | java }
Autor: Érico Veriscimo (ETEC Guaianazes)

Problema simples de cálculo de porcentagens, mas que possui algumas particularidades que acabou confundindo várias equipes. Considerando D, F e R como variáveis do tipo inteiro, o cálculo da frequência, em porcentagem, sairia da seguinte forma:

```
freq = (float)((float)(F+R)/D)*100;
```

Note a necessidade dos casts (float), pois freq deve ser declarada como float. Outro cálculo necessário é a quantidade de dias que Iago já faltou:

```
jaFaltou = D - (R+F);
```

A variável jaFaltou deve ser declarada como inteiro.

Agora entra a parte onde decidimos em que caso Iago se encaixa. No caso da frequência (freq) ser maior que 77, temos que calcular quanto Iago ainda pode faltar:

```
podeFaltar = (int) (D*0.25) - jaFaltou;
```

Note a presença do cast (int), pois a variável pode faltar deve ser inteira. Neste ponto é que temos a particularidade do problema, pois não podemos simplesmente imprimir o valor calculado acima (podeFaltar), é preciso verificar se a quantidade de dias restantes (R) é suficiente. Assim, se podeFaltar for maior que R, ele só poderá faltar R dias.

Problem B
Influência

Arquivo fonte: influencia.{ c | cpp | java }
Autor: Lucio Nunes de Lira (Fatec São Paulo)

Um problema com dificuldade média que pode ser resolvido com busca em matrizes e usando recursão ou uma pilha implementada pelo próprio programador. Estilo de problema recorrente nas últimas edições da competição.

Primeiramente devem ser lidos os valores inteiros que representam o número de fileiras e cadeiras da sala, que podem ser representados por linhas e colunas de uma matriz bidimensional de inteiros (atenção para o tamanho máximo possível!). Na sequência devem ser lidos os valores com a localização da fileira e cadeira de Luiz.

Pulando detalhes de implementação (como fazer uma borda envolta da matriz, uma das possíveis estratégias para resolver esse tipo de problema com a abordagem recursiva), deve-se ler os valores que preencherão a matriz com os níveis de influência de cada aluno.

Com o uso de recursividade, a partir do ponto onde está Luiz, verificamos todas as posições que estão em volta (vertical, horizontal e diagonais) e contabilizamos todos com nível de influência inferior ao dele. Repete-se o processo para cada um dos que foram influenciados.

6ª MARATONA DE PROGRAMAÇÃO
InterFatecs2017

Problema C
Camadas Alfabéticas

Arquivo fonte: camadas.{ c | cpp | java }
Autor: Lucio Nunes de Lira (Fatec São Paulo)

Um problema com dificuldade média que pode ser resolvido usando uma matriz que será preenchida camada a camada por letras, portanto uma matriz bidimensional de caracteres.

Dentre os pontos críticos temos o tamanho da matriz, que poderá ter o tamanho de até 105 × 105 elementos (faça o desenho e encontrará a relação $número\ de\ camadas \times 2 + 1$), e a forma de exibição, que foi definida no enunciado com apoio dos exemplos da Figura C.1.

Problem D
Superstition

Source file: superstition.{ c | cpp | java }
Author: Anderson V. de Araujo/Lucas Tsutsui (UFMS)

Este é um problema cuja solução envolve a descoberta de uma recorrência linear.
Seja F(N) a quantidade de maneiras possíveis de responder a uma prova com N questões, seguindo a restrição do professor.
Se N < 3, é fácil perceber que F(1) = 2 e F(2) = 3. Senão, considere as possíveis maneiras de responder a N-ésima questão da prova:

- Caso escolha-se responder a ela com a opção *Verdadeiro*, então a questão N-1 deve ser respondida como *Falso*, para seguir a restrição do professor, e podemos responder à questão N-2 de qualquer forma. Logo, o número de maneiras será igual a F(N-2).
- Caso escolha-se responder a ela com a opção *Falso*, então podemos responder à questão N-1 de qualquer forma. Logo, o número de maneiras será igual a F(N-1).

Considerando os dois casos, temos que F(N) = F(N-1) + F(N-2). Obtemos então a seguinte recorrência:
F(1) = 2
F(2) = 3
F(N) = F(N-1) + F(N-2), se N > 2
Repare que essa recorrência é similar à recorrência que define os números de Fibonacci, possuindo apenas diferentes casos base. Podemos então fazer uso de um algoritmo que calcula o N-ésimo termo de Fibonacci e adaptar para nosso problema, alterando apenas os casos base. É suficiente escolher um algoritmo que realize essa tarefa, no mínimo, em complexidade linear de tempo para passar no tempo limite do problema, dados os limites da entrada.
Ainda é preciso adaptar o algoritmo para que retorne sempre F(N) mod 10^9+7. Para isso, utilizamos a propriedade da adição da aritmética modular:

$$(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$$

Podemos então dizer que:

$$F(N) \bmod 10^9+7 = ((F(N-1) \bmod 10^9+7) + (F(N-2) \bmod 10^9+7)) \bmod 10^9+7$$

Uma otimização (desnecessária) seria pré-calcular e armazenar, em um vetor, todos os valores de F(N) mod 10^9+7 antes de fazer a leitura da entrada. Assim, podemos responder a todos N's da entrada em complexidade de tempo O(1).

Links para estudo:

- Aula sobre recorrências lineares: https://www.youtube.com/watch?v=srXWcQt6q10
- Como encontrar o n-ésimo termo de Fibonacci: http://www.geeksforgeeks.org/program-for-nth-fibonacci-number/

Problema E
Posição de Equilíbrio

Arquivo fonte: equilibrio.{ c | cpp | java }
Autor: Julio Lieira (Fatec Lins)

O problema consiste em perceber que a diferença dos pesos vai diminuindo à medida que se percorre a string do início para o fim. Assim, iniciando do primeiro caractere da string, calcula-se a diferença da soma do lado esquerdo e do lado direito, e compara-se com a diferença anterior, caso seja maior, continua-se até que se obtém uma diferença maior que a anterior. Como consta no enunciado, é importante notar que a cada caractere avançado, quando a posição de equilíbrio a ser calculada é a posição exata de um caractere, o peso do referido caractere é dividido entre os dois lados para a soma dos pesos.

Problema F
Encurtador de Url

Arquivo fonte: url.{ c | cpp | java }
Autor: Julio Lieira (Fatec Lins)

O problema se resume em um problema simples de conversão de base, porém para a base 62.

Problema G
Teorema de Zeckendorf

Arquivo fonte: teorema.{ c | cpp | java }
Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

O problema consiste em determinar os termos da sequência de Fibonacci que, somados e sem repetição, produzem o valor informado. É dada a garantia de que tal sequência existe para qualquer número natural e que essa sequência é única. A solução é bastante simples, obtida por meio de um algoritmo guloso: gera-se a sequência de Fibonacci até que se alcance ou se ultrapasse o valor de N. Colocamos o maior número dessa sequência que seja menor ou igual a N no resultado e o subtraímos do valor de N. Agora buscamos o próximo termo que seja menor ou igual ao novo valor de N e o colocamos

no resultado, subtraindo então de N o termo encontrado. O processamento prossegue até que o N seja igual a zero.
Exemplo:
N = 125
Sequência de Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
Resultado = (89
N = 125 – 89 = 36
Resultado = (89+34
N = 36 – 34 = 2
Resultado = (89+34+2
N = 2 – 2 = 0
Resultado = (89+34+2)

Problema H
Hash Function

Arquivo fonte: hash.{ c | cpp | java }
Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

São dados um valor numérico da chave a ser processada e a quantidade de bits que o valor hash possui. Converte-se o valor da chave para binário e separa-se em tantas partes quantas necessárias considerando a quantidade de bits recebida no segundo parâmetro da entrada do caso de teste. Caso necessário, completa-se a última parte (a mais à esquerda) com zeros iniciais para atingir o comprimento desejado. Faz-se a operação de XOR manualmente, sobre a versão em binário das partes ou então utilizando-se o operador bitwise '^' sobre a versão numérica das partes. O valor produzido é o resultado.

Problema I
Cabeamento

Arquivo fonte: cabos.{ c | cpp | java }
Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

É necessário receber as coordenadas de cada ponto e armazena-las em um vetor. Em seguida é necessário determinar a distância entre cada par de pontos, utilizando a fórmula fornecida no enunciado, colocando-se cada distância em uma matriz ou lista de adjacências. Finalmente, executa-se sobre o grafo algum algoritmo de determinação da árvore geradora mínima, como por exemplo Prim, Kruskal ou Boruvka, somando-se as distâncias das arestas componentes da árvore. Imprime-se o resultado.
Para evitar divergências de precisão, trabalhar com números reais de precisão dupla, conforme indicava o enunciado e receber os valores numéricos como inteiros.

6ª MARATONA DE PROGRAMAÇÃO
InterFatecs2017

Problema J
Racha Cuca

Arquivo fonte: racha.{ c | cpp | java }
Autor: Danilo Ruy Gomes (Fatec Itapetininga)

Problema envolvendo matrizes e cadeias de caracteres. A ideia era percorrer a cadeia na horizontal, vertical e diagonal principal e inversa. A dificuldade no exercício estava ao percorrer as posições de maneira inversa, para isto, uma sugestão de resolução seria criar uma variável que armazenasse cada palavra de maneira invertida, e percorrer a matriz da última posição da linha, coluna ou diagonal para seu início.

Problema K
Paridade Numérica

Arquivo fonte: paridade.{ c | cpp | java }
Autor: Reinaldo Arakaki (Fatec São José dos Campos)

Uma solução simples seria ler o número como uma string, depois percorrer a string caractere por caractere e se o caractere for 0, 2, 4, 6 ou 8 incremente o contador de par, senão incremente o contador de ímpar. Imprima na saída, a soma dos contadores, seguido do contador de par, seguido do contador de ímpar.

Problema L
Don Perrito

Arquivo fonte: perrito.{ c | cpp | java }
Autor: Leandro Luque (Fatec Mogi das Cruzes)

A solução do problema envolve identificar quantas voltas completas os cachorros deram em uma pista circular dada a distância percorrida por eles e o raio da pista. A distância percorrida em uma (1) volta é igual ao perímetro da circunferência da pista (fórmula dada no enunciado): 2*Pi*Raio. Para responder o problema, bastava truncar (função piso) o resultado da razão entre distância e o perímetro da circunferência da pista. Matematicamente: Piso(distancia/(2*Pi*Raio)).