Aula 6:
Upsolving do Warmup 2

***Disciplina*: Maratona de Programação 1**

Profs. Edmilson Marmo e Luiz Olmes

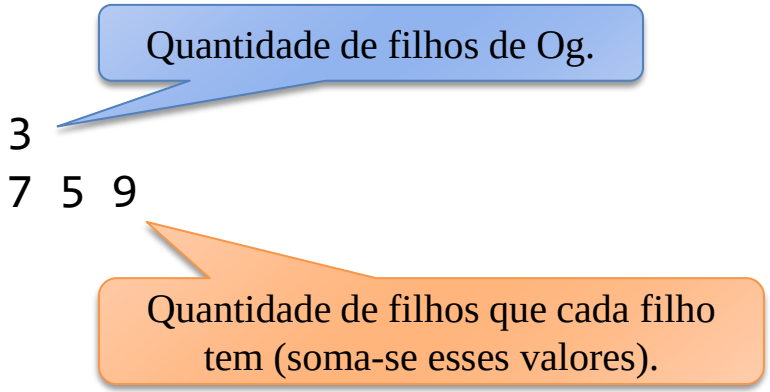
edmarmo@unifei.edu.br, olmes@unifei.edu.br

UNIFEI
UNIVERSIDADE FEDERAL DE ITAJUBÁ

Problema A: Ajudando o Og

- Total de Submissões: 67
- Submissões Aceitas: 53 (79%)

- Este problema é muito simples de ser resolvido: basta ler a entrada e somar os valores.



Problema D: Revisão

- Total de Submissões: 69
- Submissões Aceitas: 34 (49%)

- Esta é uma versão mais simples do problema “1340: Eu posso adivinhar a Estrutura de Dados”, presente na Tarefa 4.

- Basta criar uma `stack` de `int` e, a cada operação do tipo 1, realizar a inserção na pilha. Para cada operação do tipo 2, testa-se o topo da stack com o valor da entrada. Se forem diferentes, não é pilha.

Problema B: Erro dos Professores

- Total de Submissões: 124
- Submissões Aceitas: 22 (18%)

- Neste problema, deseja-se verificar a duplicidade de um par de caracteres. Por exemplo, “a baba não veio trabalhar”.

- Para cada palavra que compõe a frase, pode-se percorrer a string testando se:
  - `str[i] == str[i + 2] && str[i + 1] == str[i + 3]`

- Detalhe adicional: não se deve diferenciar letras maiúsculas de minúsculas. Para isso, basta converter tudo para maiúsculas ou minúsculas. As funções `toupper` / `tolower` pertencem ao arquivo `ctype.h`.

Problema E: Relógio Binário

- Total de Submissões: 26
- Submissões Aceitas: 14 (54%)

- Para resolver este problema, é preciso separar a hora, o minuto e o segundo, convertendo-os para `int`. Depois, cada número deve ser convertido para um vetor de inteiros, contendo seis dígitos binários correspondentes ao respectivo número decimal.
- Por exemplo: 10:37:49
  - $10 \rightarrow 001010$
  - $37 \rightarrow 100101$
  - $10 \rightarrow 110001$

Problema E: Relógio Binário

- Para a resposta horizontal, basta colocar estes resultados em sequência, no mesmo vetor de saída:
  - 00101010010111000 1

- Para montar o vetor vertical, basta saltar proporcionalmente a distância de 6 elementos de cada subvetor:

```
1.  for (int j = 0, x = 0, y = 6, z = 12; j < 18; x++, y++, z++)
2.  {
3.      VER[j++] = HOR[x];
4.      VER[j++] = HOR[y];
5.      VER[j++] = HOR[z];
6.  }
```

Problema E: Relógio Binário

- Este problema também pode ser resolvido utilizando-se a classe `bitset` da biblioteca STL.

- Por exemplo, o seguinte código cria um bitset de 10 elementos:

```
1.  bitset<10> s;
2.  s[1] = 1;
3.  s[4] = 1;
4.  s[7] = 1;
5.
6.  cout << s[4] << "\n"; // 1
7.  cout << s[5] << "\n"; // 0
8.
```

Problema E: Relógio Binário

- O exemplo anterior poderia ser construído a partir do seguinte trecho de código:

```
1.  bitset<6> bsH, bsM, bsS;
2.
3.  bsH = bitset<6>(10);
4.  bsM = bitset<6>(37);
5.  bsS = bitset<6>(49);
```

Problema F: ETs

- Total de Submissões: 16
- Submissões Aceitas: 3 (19%)

- Para resolver esse problema, é necessário identificar os pontos que são colineares entre si.

- Para saber se três pontos $[(x_1, y_1), (x_2, y_2), (x_3, y_3)]$ são colineares, deve-se verificar se o determinante da seguinte matriz é zero:

$$det(A) = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0$$

Problema F: ETs

- Uma vez armazenados todos os pontos em vetor de “pontos”, é preciso encontrar os coeficientes da equação geral da reta, entre dois pontos do conjunto e verificar se existem outros pontos que pertencem à mesma reta.
- Recordando que a equação da reta é dada por:

$$ax + by + c = 0$$

ou

$$(y1 - y2)x + (x2 - x1)y + (x1y2 - x2y1) = 0$$

- Assim:

```
1.  a = pontos[1].y - pontos[2].y;
2.  b = pontos[2].x - pontos[1].x;
3.  c = pontos[1].x * pontos[2].y - pontos[2].x * pontos[1].y;
```

Problema F: ETs

- Para contar quantos pontos estão ao longo de uma mesma linha comum:

```
1.  cont = 0;
2.
3.  for (k = 0; k < npontos; k++)
4.  {
5.      if (a * pontos[k].x + b * pontos[k].y + c == 0)
6.      {
7.          cont++;
8.      }
9.  }
```

Problema C: Soma do Intervalo

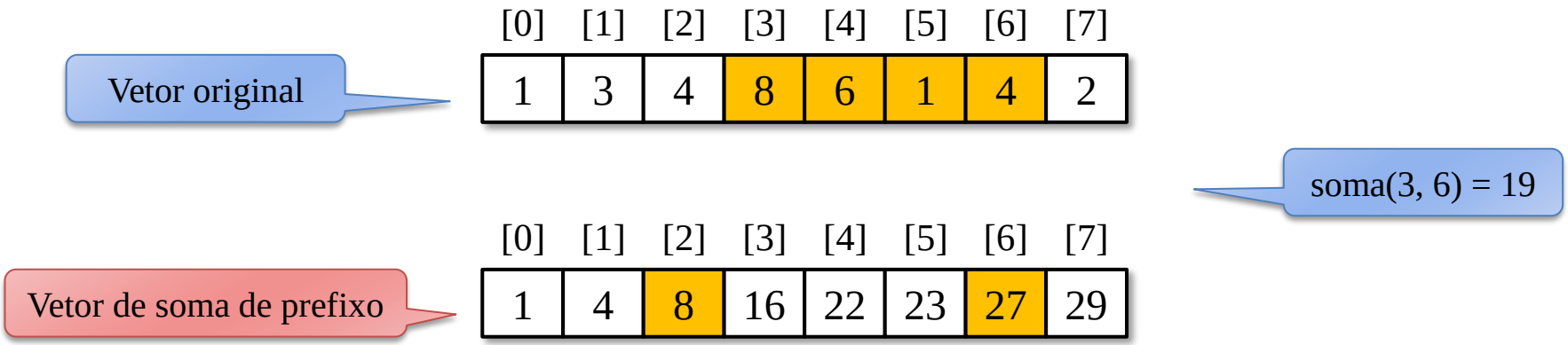
- Total de Submissões: 71
- Submissões Aceitas: 0 (0%)

- O problema pode ser facilmente resolvido através de um vetor de soma de prefixo.

- Cada valor no vetor de prefixo corresponde à soma dos valores anteriores do vetor original, até a posição atual.
  - Isto é, o valor da posição k é soma(0, k).

Problema C: Soma do Intervalo

- Um vetor de soma de prefixo pode ser construído em tempo O(n). A partir daí, pode-se calcular qualquer valor soma(a, b) em tempo O(1), da seguinte maneira:
  - soma(a, b) = soma(0, b) – soma(0, a – 1).
  - onde, por definição, soma(0, –1) = 0.



Problema C: Soma do Intervalo

- No problema do Warmup, há um detalhe, onde a operação 2 deve inserir o elemento no início do vetor.
- Neste caso, seria necessário recomputar o vetor de soma de prefixo a cada operação 2, o que levaria a um julgamento como "Time Limit Exceeded".
- Para evitar que o problema não passe por tempo, pode-se armazenar a entrada em ordem inversa e, a cada operação 2, inserir o novo elemento na última posição do vetor.
- Assim, recomputa-se apenas a última posição do vetor de soma de prefixo.
  - Atentar-se para os índices: o índice 1 é o último elemento!

Aula 6:
Upsolving do Warmup 2

***Disciplina*: Maratona de Programação 1**

Profs. Edmilson Marmo e Luiz Olmes

edmarmo@unifei.edu.br, olmes@unifei.edu.br

UNIFEI
UNIVERSIDADE FEDERAL DE ITAJUBÁ