

Análise dos Problemas do Treino de 21/07/17

Maratona de Programação Unioeste

G – Bafo

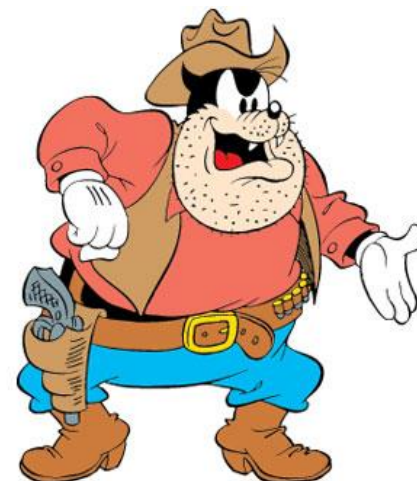
- **Problema**

Dados o número de figurinhas viradas por cada jogador, imprimir que ganhou o jogo.

- **Solução**

Apenas implementar o que é pedido.

```
scanf("%d", &rodadas);  
for(int i = 0; i < rodadas; i++)  
{  
    scanf("%d %d", &a, &b);  
    aldo += a;  
    beto += b;  
}  
if(aldo > beto) printf("Aldo\n");  
else           printf("Beto\n");
```



I – Vivo ou Morto

- **Problema**

Dadas as ordens do chefe e as ações executadas pelos participantes, dizer quem ganhou o jogo.

- **Solução**

Implementar o que é pedido.

Podemos usar dois vetores: um armazena os jogadores no começo da rodada. No outro, vamos escrevendo os jogadores que avançam para a próxima rodada (também é possível fazer isso no mesmo vetor).

I – Vivo ou Morto

```
for(int r = 0; r < rodadas; r++)
{
    scanf("%d %d", &k, &ordem);

    pos = 0;

    for(int i = 0; i < k; i++)
    {
        scanf("%d", &acao);
        if(acao == ordem)
            v2[pos++] = v1[i];
    }

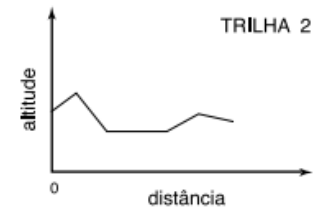
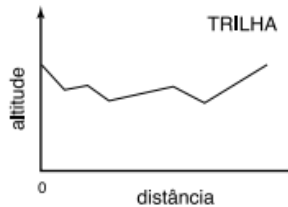
    for(int i = 0; i < pos; i++)
        v1[i] = v2[i];
}

printf("Teste %d\n", teste);
printf("%d\n\n", v1[0]);
```

F - Trilhas

- **Problema**

Dadas as alturas das trilhas, imprimir a trilha que, em algum dos sentidos, exige o menor esforço de subida – o menor trecho de maior desnível.



- **Solução**

Implementar o que é pedido.

Para cada trilha, calculamos a maior diferença entre dois elementos consecutivos de frente pra trás e de trás pra frente.

Armazenamos em outra variável o menor esforço de subida entre todas as trilhas já vistas.

F - Trilhas

```
for(int t = 0; t < trilhas; t++)
{
    scanf("%d", &tam);
    for(int i = 0; i < tam; i++)
        scanf("%d", &alt[i]);

    int maior = 0;

    for(int i = 1; i < tam; i++)
        maior = max(maior, alt[i] - alt[i-1]);

    for(int i = tam-2; i >= 0; i--)
        maior = max(maior, alt[i] - alt[i+1]);

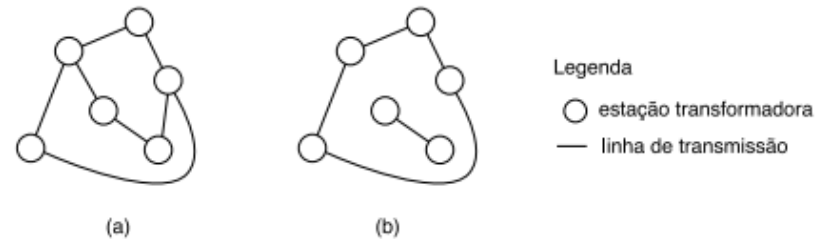
    if(maior < melhor)
    {
        melhor = maior;
        resp = t;
    }
}

printf("%d\n", resp);
```

H – Transmissão de Energia

- **Problema**

Dada a configuração do sistema de transmissão de energia, dizer se o mesmo está normal (todas as estações alcançam todas as outras) ou em falha.



- **Solução**

Este é um problema de grafos que pede para verificar se o grafo dado na entrada é conexo.

Podemos resolvê-lo com um algoritmo de percorrer grafos (BFS ou DFS) ou ainda com Disjoint Set-Union.

H – Transmissão de Energia

- Resolvendo com DSU:
- Inicializamos cada estação em um conjunto sozinha.
- Ao ler uma ligação entre a estação X e a estação Y , fazemos união do conjunto de X com o conjunto de Y (as estações no conjunto de X passarão a alcançar as estações no conjunto de Y e vice-versa).
- Ao final, todas as estações devem estar no mesmo conjunto: todas devem ter o mesmo 'pai' do conjunto.

H – Transmissão de Energia

```
scanf("%d %d", &estacoes, &linhas);

DSU_init(estacoes);

for(int i = 0; i < linhas; i++)
{
    scanf("%d %d", &a, &b);
    DSU_union(a, b);
}

int pai = DSU_find(1), ok = 1;

for(int i = 0; i < estacoes; i++)
{
    if(DSU_find(i) != pai)
    {
        ok = 0;
        break;
    }
}

printf("%s\n", ok ? "normal" : "falha");
```

Demais problemas

- A serem discutidos futuramente:

Palíndrome – Programação Dinâmica

Pedido de Desculpas – Programação Dinâmica (Problema da mochila)