Aula 01

Rodrigo G. Araújo

24/07/2019

Programação Competitiva

Resolver problemas de Ciência da Computação conhecidos, o mais rápido possível.

Steven & Felix Halim (2010)

Programação Competitiva

- Todos os problemas contem souluções existentes
- A velocidade caracteriza a competição
- Formar profissionais capazes de produzir softwares de qualidade
- Trabalho em equipe

Programação Competitiva

Competitive programming combines two topics: (1) the design of algorithms and (2) the implementation of algothms

Antti Laaksonen (2018)

Programação Competitiva

- Resolução de problemas e pensamento matemático
- Combinação de técnicas conhecidas
- Navas interpretações de técnicas
- Habilidade em programação

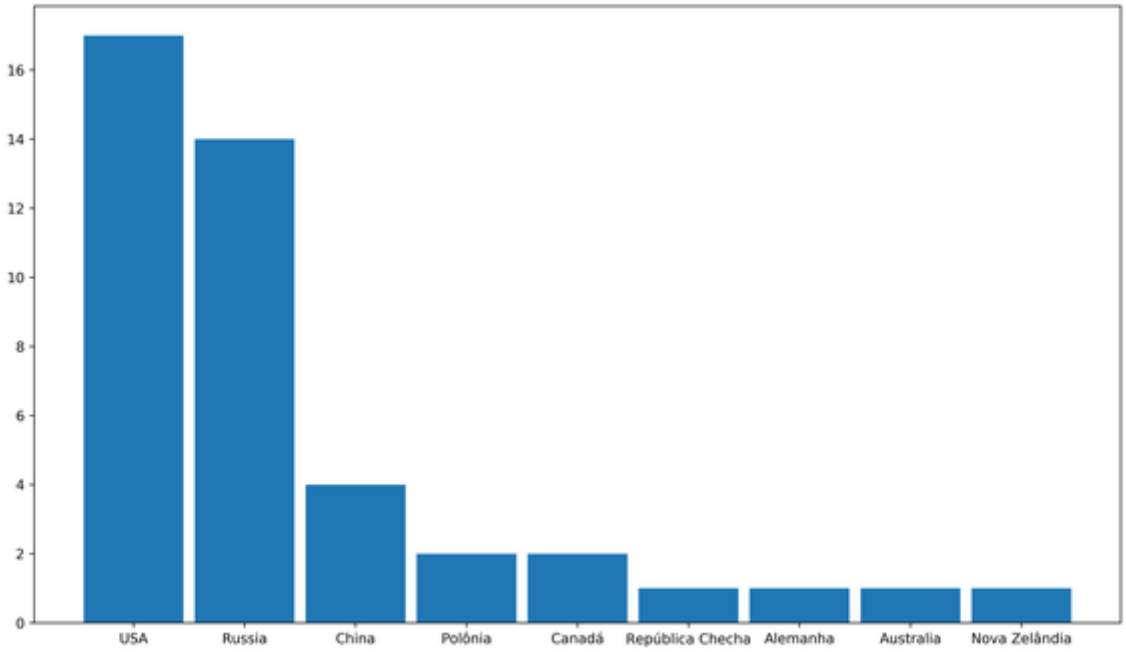
ACM ICPC

- Sub-Regional (14/09 - Centro Universitário IESB)
- Regional (07/11 - Campina Grande)
- Mundial (TDB)
- Equipes de três alunos, um reserva e um coach
- 8 a 14 problemas em 5 horas

Critérios de Vitória

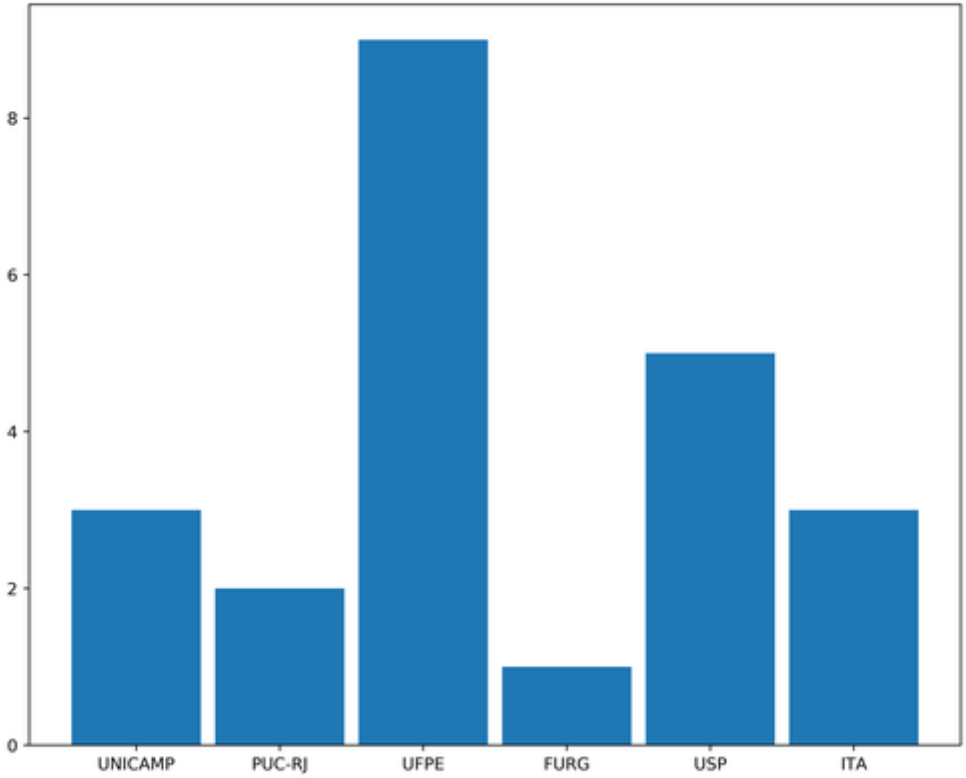
- Maior número de problemas
- Menor tempo total na subimissão de soluções
- O tempo de subimissão é o tempo transcorrido até o momento na competição
- Cada subimissão incorreta antes de acertar gera uma penalidade de 20 minutos
- O tempo total é a soma dos tempos de submissão acressido das penalidades
- Cada problema tem uma cor, e é entregue um balão ao acertar

Campeões Mundiais



Brasil conseguiu 13º lugar no mundial em 2005 com a USP

Campeões Brasileiros



Brasília (UNB) conseguiu o 2º lugar em 1996

IESB nas Maratonas

- 2016 - 11º Lugar - 3 Balões - Programação Orientada a Cebola
- 2017 - 7º Lugar - 5 Balões - Programação Orientada a Cebola
- 2018 - 9º Lugar - 3 Balões - DevLabs

UNB CIC e UNB FGA disputam a primeira posição todos os anos.

Juizes Online

URI - Iniciantes

Codeforces - Intermediário/Avançado

Juizes Online

Cada solução submetida ao juiz retornará um feedback sobre a solução

As linguagens aceitas na Maratona de Programação SBC são: C, C++, Java e Python

Código	Erro	Descrição
WA	*Wrong Answer*	Uma ou mais saídas geradas estão incorretas. O juiz não informa as entradas que geraram o erro nem a resposta correta para tais entradas
PE	*Presentation Error*	As saídas do programa estão corretas, mas a apresentação (formatação, espaçamento, etc) está diferente do que foi especificado
CE	*Compilation Error*	O programa não compila corretamente. Em geral, os juízes listam os parâmetros de compilação utilizados na correção

Código	Erro	Descrição
RE	*Runtime Error*	O programa trava durante a execução, geralmente por conta de falhas de segmentação, divisão por zero, etc
TLE	*Time Limit Exceeded*	Os programas devem gerar as saídas válidas dentro de um limite de tempo especificado. Caso o programa exceda este tempo, esta será a resposta do juiz
MLE	*Memory Limit Exceeded*	O programa requer mais memória em sua execução do que o juiz permite

Imagens obtidas de https://github.com/edsomjr/TEP/

Entrada e Saída

A programação competitiva requer que os problemas sejam solucionados lendo arquivos de entrada especificos e escrevendo a saída em um arquivo da forma solicitada. Os arquivos em sua maioria são obtivos pela entrada e sáida padrão.

Os arquivos de entrada são divididos em quatro tipos:

1. Um único caso do problema
2. N casos do problema, com o valor de N informado na primeira linha
3. N casos do problema, acabando com um valor informado
4. M casos do problema, até que acabe o arquivo (EOF)

Entrada e Saída

Categoria 1: Um único caso do problema

Arquivo de Entrada

X Y

```
x, y = map(int, input().split())
print(x+y)
```

Entrada e Saída

Categoria 2: Varios casos informados no inicio.

Arquivo de Entrada

QTD

X Y

```
qtd = int(input())
for _ in range(qtd):
    x, y = map(int, input().split())
    print(x+y)
```

Entrada e Saída

Categoria 2: Varios casos acabando com um valor.

Arquivo de Entrada

X Y

X Y

X Y

-1 -1

```
while True:
    x, y = map(int, input().split())
    if x == -1 and y == -1:
        break
    print(x+y)
```

Entrada e Saída

Categoria 2: Varios casos até o fim de arquivo.

Arquivo de Entrada

X Y

X Y

X Y

X Y

```
while True:
    try:
        x, y = map(int, input().split())
        print(x+y)
    except EOFError:
        break
```

Categorização de problemas

- Ad-Hoc
- Busca Completa
- Dividir e Conquista
- Gulosos
- Prorgramação Dinâmica
- Grafos
- Matemática
- Geometrica Computacional
- Estrutura de Dados

Classificação Pessoal

- A - Já resolvi um parecido e posso resolver de novo rapidamente
- B - Já vi um parecido e sei que não consigo resolver
- C - Nunca vi

Divida a classificação entre a equipe, deixe somente um, no máximo dois no computador

Execicios

https://codeforces.com/group/uZDbxesr6A/contests

Exercicios

A - Bit++

```
soma = 0
for _ in range(int(input())):
    if input()[1] == '+':
        soma+=1
    else:
        soma-=1
print(soma)
```

Exercicios

B - Beautiful Matrix

```
x, y = 0, 0
for i in range(5):
    for j, number in enumerate(input().split()):
        if number == '1':
            x, y = i, j
tx, ty = abs(x - 2), abs(y - 2)
print(tx+ty)
```

Exercicios

C - Stones on the Table

```
qtd, pedras = int(input()), input()
movimentos = 0
for index, pedra in enumerate(pedras):
    if index+1 < qtd and pedra == pedras[index+1]:
        movimentos+=1
print(movimentos)
```

Exercicios

D - Word

```
lower, upper = 0, 0
texto = input()
for letra in texto:
    if 65 <= ord(letra) <= 90:
        upper += 1
    else:
        lower += 1
print(texto.upper() if upper > lower else texto.lower())
```

Exercicios

E - Fox And Snake

```
n, m = map(int, input().split())
direita = True
for i in range(n):
    if i%2 == 0:
        print("#"*m)
    else:
        if direita:
            print("."*(m-1), "#", sep='')
            direita = False
        else:
            print("#", "."*(m-1), sep='')
            direita = True
```