Universidade Federal do Espírito Santo
Centro de Ciências Agrárias – CCA UFES
Departamento de Computação

Tópicos Especiais em Programação

Sumário

- Metodologia;
- Distribuição das notas;
- Referências adotadas;
- Justificativa;
- Sistema de julgamento dos problemas;
- Linguagens de programação;
- Padrão de Entrada e Saída;
- Dicas;
- Tipos elementares de dados;
- Criação das Equipes.

Metodologia

- Em cada semana do semestre serão ministradas duas aulas, sendo:
  - Uma aula teórica abordando a aplicação dos tópicos para a solução dos problemas computacionais referentes;
  - Uma aula prática com aplicação dos tópicos abordados teoricamente. Nessa aula serão abordados:
    - 2 problemas fáceis (F);
    - 1 problema intermediário (I);
    - 1 problema difícil (D).
- Linguagem adotada: C++

Distribuição das Notas

- A nota será uma média ponderada dos problemas resolvidos:

$$\left[\frac{\sum_{2}^{14}\left(0{,}30*F+0{,}30*F+0{,}40*I+0{,}2*D\right)}{\left(14-2+1\right)}\right]*8+2*\left(\begin{array}{c}porcentagem\,de\,acertos\\ na\,MIP\end{array}\right)$$

  - sendo:

    F: problema fácil resolvido;

    I: problema intermediário resolvido;

    D: problema difícil resolvido;

    MIP: maratona interna de programação.

Referências Bibliográficas

[1] HALIM, S.; HALIM, F. *Competitive Programming*: Increasing the Lower Bound of Programming Contests. Lulu, 2010.

[2] SKIENA Steven S.; REVILLA Miguel A. *Programming Challenges*: The Programming Contest Training Manual. Springer: 2003.

[3] MUSSER, David R.; DERGE, Gillmer J.; SAINI, Atul. *C++ Programming with the Standard Template Library*. Addison-Wesley: 2001.

Justificativa

- Os jogos, quebra-cabeças e desafios existentes nos problemas das competições de programação internacionais trazem formas de:
  - Melhorar sua habilidades algorítmicas;
  - Melhorar suas técnicas de programação;
  - Ter prazer em:
    - Entender técnicas já utilizadas em diversos programas existentes;
    - Em obter o máximo de desempenho dos programas;
    - Em obter o conhecimento necessário para descodificar segredos, penetrar sistemas...

Afinal, a questão espiritual para a elegância pode transformar um hacker em um artista.

Sistema de julgamento dos problemas

- Os problemas originam-se da Universidade de Valladolid (UVa).
- Há dois locais de julgamentos:

  <http://www.programming-challenges.com>

  <http://online-judge.uva.es>
- Os problemas serão utilizados para:
  - Ilustrar conceitos importantes na programação;
  - Entender de forma concreta os algoritmos e técnicas utilizadas;
  - Aplicar os conceitos de matemática, geometria e computação em histórias divertidas de se programar.

- Os assuntos são importantes mesmo se você não quiser competir nas maratonas,
  pois tratá métodos úteis em qualquer contexto de programação.

Respostas dos julgamentos

- ***Accepted (AC)***: Programa correto e executa dentro dos limites de tempo e de memória.
- ***Presentation Error (PE)***: A saída do programa está correta, mas não está no formato correto. Você deve verificar espaços, formatação, etc.
- ***Accepted (PE)***: Programa correto, mas com uma apresentação contendo um erro mínimo, como um espaço em branco ou uma nova linha a mais.
- ***Wrong Answer (WA)***: O programa retornou uma resposta incorreta. Deve ser realizada uma correção do código para enviar novamente.
- ***Compile Error (CE)***: O código fonte não pode ser compilado. As mensagens de erro da compilação serão exibidas.

Respostas dos julgamentos

- ***Runtime Error (RE)***: O programa falhou durante a execução devido a uma falha de segmentação, ou exceção de ponto flutuante, ou algum problema similar. Deve-se verificar os limites dos vetores, seu acesso, se houve divisões por zero, etc.
- ***Time Limit Exceeded (TL)***: O programa ultrapassou o tempo limite.
- ***Memory Limit Exceeded (ML)***: O programa tentou utilizar mais memória do que podia.
- ***Output Limit Exceeded (OL)***: O programa tentou imprimir muito mais saídas do que era necessário. Geralmente acontece quando o programa entra em loop infinito.
- ***Restricted Function (RF)***: O programa tentou utilizar uma chamada de sistema ilegal, como fork() ou fopen(). Comporte-se!!
- ***Submission Error (SE)***: Um ou mais campos de submissão não foram corretamente informados.

Respostas dos julgamentos

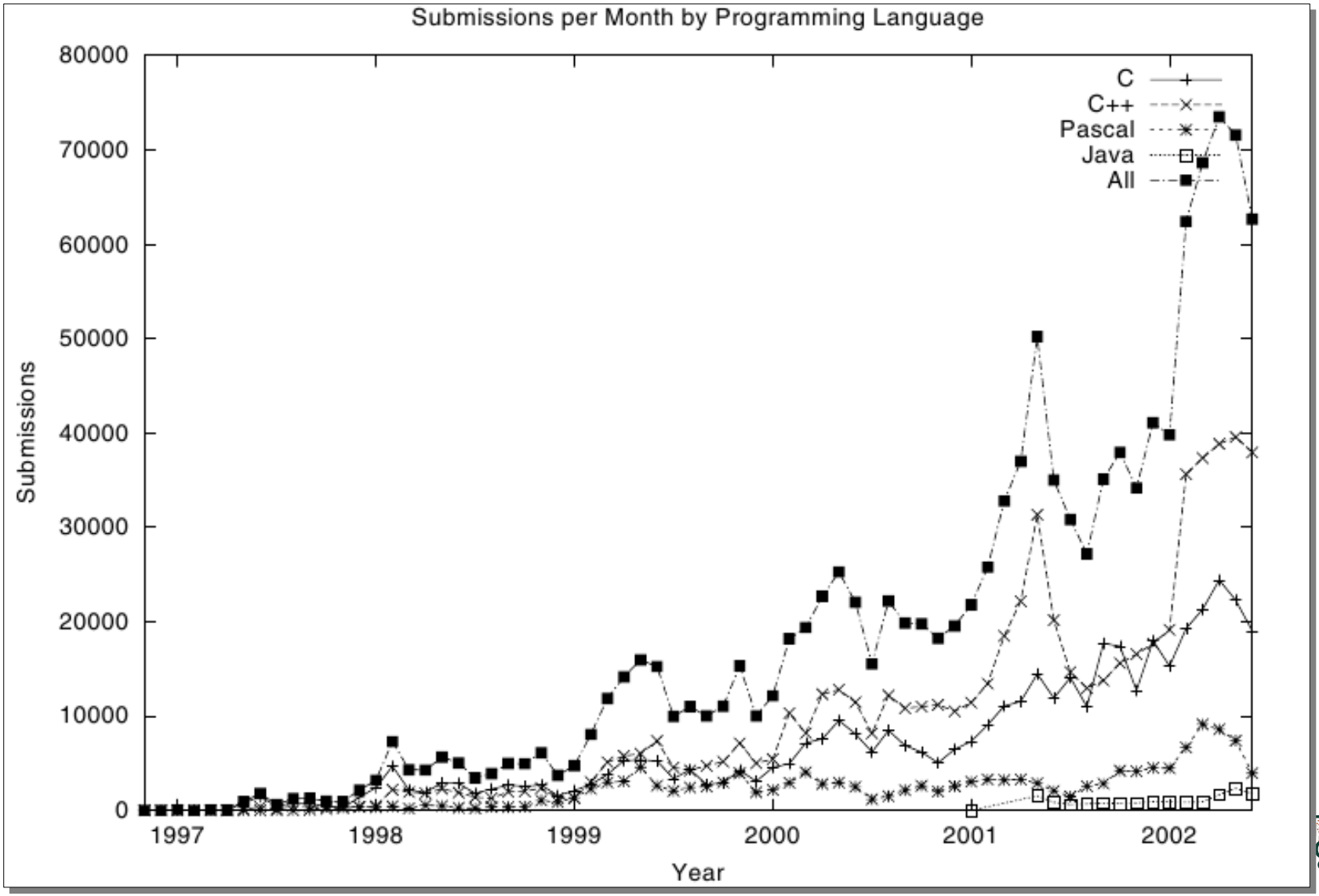
- No enunciado do problema,
  - Somente um exemplo será apresentado;
  - Serão apresentados todos os limites possíveis;
  - Será informado como deverá ser a entrada e como deverá ser escrita a saída.
- Porém, no julgamento serão realizados todos os testes, ou seja, utilizando todos os limites e possibilidades de erros.
- Atenção:
  - Se o programa falhar para um único teste realizado no julgamento, esse erro específico não será informado para você.
  - Somente será enviada uma mensagem falando que seu programa está errado.

Linguagens de Programação

- Qual linguagem utilizar?

  R: a que você mais conhece e domina.

- Porém:



Padrão de Entrada e Saída

- As maratonas de programação não aceitam a leitura ou escrita de arquivos.
- Então, deve-se ler e escrever da saída padrão.
- Exemplos:

```c
#include <stdio.h>

int main() {
   long p,q,r;
   while ( scanf("%ld %ld",&p,&q) != EOF)
    {
        if (q > p) r = q - p;
        else       r = p - q;

        printf("%ld\n",r);
    }
}
```

```cpp
#include <iostream>
using namespace std;

int main() {
    long p,q,r;
    while (cin >> p >> q)
    {
        if (q > p)  r=q-p;
        else        r=p-q;

        cout << r << endl;
    }
}
```

Compilando e testando

- Crie um arquivo texto com as entradas.

  Ex.: `entradas_prog1.txt`

- Escreva e compile seu código-fonte:

```
gcc prog1.c -o prog1
```

ou

```
g++ prog1.cpp -o prog1
```

- Teste-o com as entradas:

```
./prog1 < entradas_prog1.txt
```

Dicas

- Escreva os comentários primeiro:
  - Antes de escrever as sentenças que você supõe fazer, escreva primeiro seus comentários.
  - Isso é importante, pois se você não conseguir escrever seu comentário de forma fácil, você provavelmente não entende corretamente o que seu programa/função fará/faz.
- Documente suas variáveis:
  - Escreva um comentário de uma linha para cada variável.
  - Se você não souber descrevê-la facilmente, você provavelmente não sabe porquê ela está ali.
- Utilize constantes com cuidado:
  - Declare-as no topo do seu programa.
  - Utilize macros: Ex.: `#define PI 3.141592653589793`
- Utilize as flags de compilação para Warnings e não envie nenhum código com warnings, pois eles sempre podem ser evitados.

Dicas

- Utilize `enum` e novos tipos somente se necessário.
- Escreva funções para evitar redundâncias e erros de código.
- Faça a correção de erros mais fácil:
  - Aprenda a utilizar o ambiente de *debug* do seu sistema.
  - Imprima as variáveis com seu nome.
  - Utilize `"\n\n"` no fim de `printf`'s utilizados para debug, pois isso garante a impressão da linguagem.
  - Também pode-se utilizar o `fflush(stdout)`.
- Aprenda a utilizar o `man` do GNU/Linux.
- Sempre teste a leitura de suas variáveis para ver se estão sendo feitas corretamente.
- Teste se o tipo de dado escolhido consegue abrigar os valores máximos dos seus problemas.

Dicas

- Sempre escreva as entradas dos testes primeiro, ou seja, nunca perca tempo digitando as entradas em cada teste que realizar do programa.
- Escreva várias entradas utilizando sempre todos os limites possíveis, isso ajuda a encontrar erros.
- Sempre declare os vetores com uma posição a mais, isso pode evitar problemas na implementação.
- Leia todos os problemas antes de fazê-los. Classifique-os então quanto ao seus níveis de dificuldade.

  Após isso, deixe que uma pessoa da equipe se concentre em fazer os problemas mais simples. Messe momento, os outros dois podem fazer o algoritmo de um dos problemas mais difíceis.
- Procure identificar qual o melhor método que se encaixa na solução do problema. Após isso, copie seu código e adapte-o ao seu problema.

Tipos elementares de dados

- Tipos básicos de dados, como *arrays*, tem uma importante vantagem sobre estruturas mais sofisticadas como listas encadeadas:
  - Simplesmente por serem mais simples;
  - Erros de ponteiros não podem ocorrer com *arrays* estáticos.
- Mesmo que balanceamento de árvores binárias, manipulação de exceções, processamento paralelo e outros modelos de manipulação de dados sejam importantes,
  eles não são necessários para fazer um programa correto capaz de resolver um problema simples.

Criação das Equipes

e mão na massa....