Dia 1

Introdução

Março de 2015

Roteiro

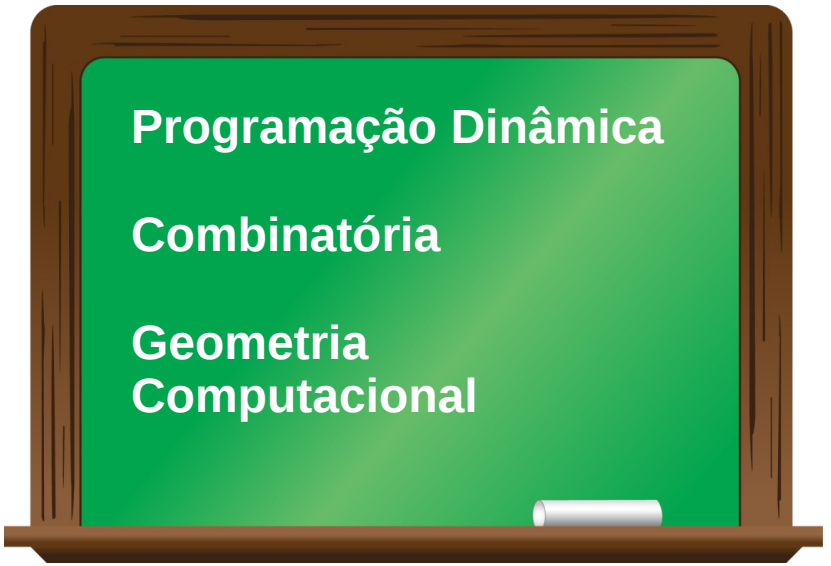
- O grupo de estudos;
- A maratona de programação da SBC;
- Linguagens de programação;
- Tratamento de entrada e saída;
- Lidando com erros;
- Dicas interessantes.
- Primeiros passos;

O grupo de estudos

- Horário do treinamento: XXX-feira às XX:Xxh
- Todos os alunos da universidade podem participar.
- Tem o objetivo de estimular a participação dos alunos em competições de programação.
- É um ótimo ambiente para trocar ideias.
- Aprender assuntos que não aprendemos na graduação.

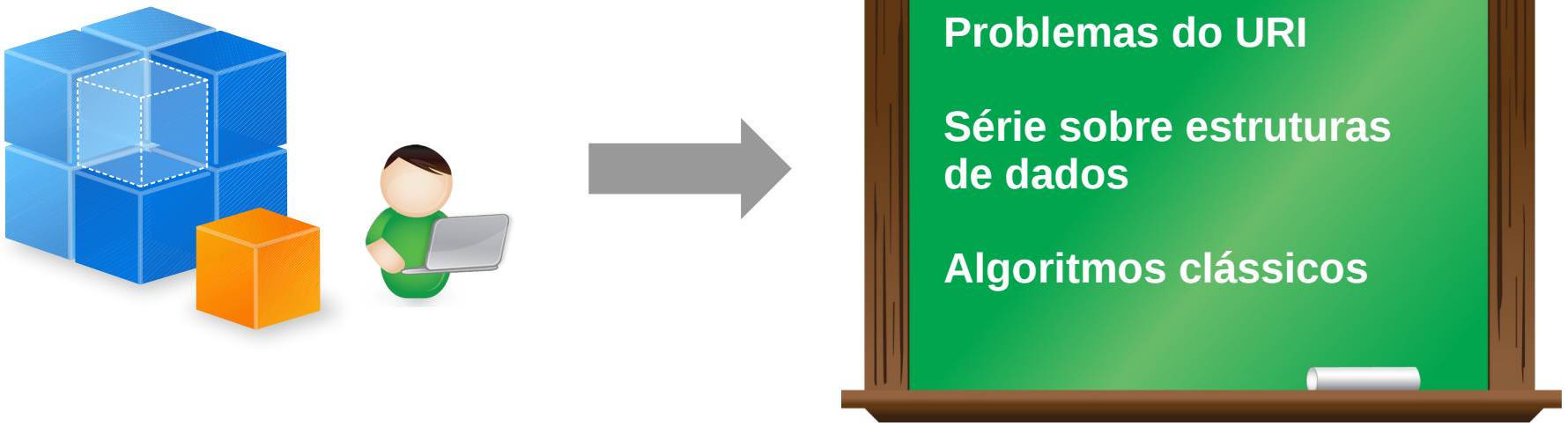
O grupo de estudos

- **Modo de funcionamento:**
  - Treinamentos ministrados por professores:



O grupo de estudos

- **Modo de funcionamento:**
  - Treinamentos ministrados por alunos:



O grupo de estudos

- **Modo de funcionamento:**
  - Recursos discutidos aqui estarão disponíveis no repositório do github: https://github.com/meitcher/estudos-maratona-2015
  - *Uma competição final entre os participantes utilizando o BOCA (sistema utilizado pela maratona de programação).*
  - Lugar para postagem de informações referentes aos treinamentos:
    - Moodle
    - Academic do URI
    - Fórum

A maratona de programação

- **O que é?**
  - A Maratona de Programação é um evento da Sociedade Brasileira de Computação que existe desde o ano de 1996.
  - Time de 3 alunos – 5 horas de competição.
  - Ganha o time que resolver mais problemas (com menos erros acumulados) em menos tempo!

Etapa regional em Três de Maio - 2015

A maratona de programação

- **Por que participar?**
  - Porque é divertido.
  - Porque você aprende coisas úteis.
    - Projeto e análise de algoritmos;
    - Novas habilidades;
  - Porque é um diferencial no mercado de trabalho.
    - Procura por ex-maratonistas vem crescendo;
    - Carreira acadêmica?

“Não que haja nada de errado em escrever programas de controle de estoque de padaria. Mas, honestamente, não é lá muito interessante.”

A maratona de programação

- **Eu ganho alguma coisa participando da Maratona?**
  - Além do conhecimento adquirido:
  - Viagens;
  - Prêmios;
  - Atividade Complementar de Graduação (ACG).

Etapa regional em Três de Maio - 2015

A maratona de programação

- **Por que treinar e como começar?**
  - Porque ninguém nasce sabendo.
  - Para aprender novos conceitos e algoritmos.
  - Com o grupo de estudos.
  - Resolvendo problemas por níveis.
  - Acompanhamento com algum livro.

Linguagens de programação

- C/C++:
  - É importante conhecer o funcionamento básico da linguagem, e conhecer recursos dela (Map, Set, Limits...).
  - Curso de C/C++ básico/avançado.
  - Curso de C++ para programadores JAVA.
- JAVA:
  - É importante conhecer o funcionamento básico da linguagem, e conhecer recursos dela (BigInteger, List, ArrayList...).

Linguagens de programação

- Um pouco sobre C++:
  - Linguagem rápida.
  - Suporte para orientação a objetos.
  - Standard Template Library (STL).
    - Estruturas de dados;
    - Algoritmos básicos;
    - Utilidades.
    - etc.

Tratamento de entrada e saída

- **Entrada:**
  - Entradas com End Of File (EOF):

```
while(cin >> n){ … }
```

  - Entradas com *n* casos:

```
cin >> n;
while(n--){ … }
```

  - Outras:

```
scanf("%d\n", &n);          # lê um inteiro e pula o \n
scanf("%*s %s", str);       # pula um nome e lê a próxima str
scanf("%[^\n]\n", str);     # lê uma linha e pula o \n
scanf("%[a-z] ", str);      # lê uma str de [a-z] e pula um char
scanf("%s%n", str, &n); # lê uma str e a qtd de char lidos
```

Tratamento de entrada e saída

- **Saída:**
  - Imprimir um espaço ' ' entre os números pares.

```
printf(“0”);
for(i=2; i<=N; i+=2) printf(“ %d”, i);
```

  - Limitar casas decimais.

```
printf(“%.4lf”, 3.1415926);  # 3.1416
```

  - Outras:

```
printf(“%4s”, str);        # margem a esquerda de 4 casas - fixa
printf(“%*s”, 4, str);     # margem a esquerda de 4 casas - dinâmica
cout << str << endl;       # imprimir uma string no c++
printf(“%s”, str);         # imprimir uma string(char*) no c
```

Tratamento de entrada e saída

- **Especificadores de formato:**

“%d”	→ int	→	16 bits	→	$(2^{-15}, 2^{15})$	
“%ld”	→ long int	→	32 bits	→	$(2^{-31}, 2^{31})$	
“%lld”	→ long long int	→	64 bits	→	$(2^{-63}, 2^{63})$	
“%u”	→ unsigned	→	16 bits	→	$[0, 2^{16})$	
“%lu”	→ long unsigned	→	32 bits	→	$[0, 2^{32})$	
“%llu”	→ long long unsigned	→	64 bits	→	$[0, 2^{64})$	
“%c”	→ char	→	08 bits	→	$[0, 2^{8})$	
“%s”	→ char*	→	XX bits	→	$n * 2^{8}$	
“%f”	→ float	→	32 bits	→	$(2^{-31}, 2^{31})$	* 23 significativos.
“%lf”	→ double	→	64 bits	→	$(2^{-64}, 2^{64})$	* 52 significativos.
“%Lf”	→ long double	→	128 bits	→	$(2^{-128}, 2^{128})$	* 112 significativos. *

  - Sempre usem *double* ao invés de *float*. Verificar *<limits.h>.*

  - Mais informaçoes: https://wpollock.com/CPlus/PrintfRef.htm

Lidando com erros

- **Accepted (ACC)** → Significa que seu código está correto, congratulations!
- **Wrong Answer (WA)** → Significa que seu código está errado. Tente reler o problema ou pensar em casos em que seu algoritmo falha.
- **Compilation Error (CE)** → Significa que teve erros durante a compilação do seu código. Tente analisar a saída do compilador.
- **Time Limit Exceeded (TLE)** → Significa que seu código demora mais do que deveria. Tente pensar uma melhor maneira de resolver o problema (usando um algorítmo de menor complexidade).
- **Presentation Error (PE)** → Significa que a saída do seu código teve algo de inesperado. Tente reler a descrição do problema e verificar *prints* esquecidos no código.
- **Runtime Error (RE)** → Significa que alguma exceção foi lançada e não tratada. Tente revisar os limites do problema.

Dicas interessantes

- **1) Tente codificar rápido!**
  - Muitas vezes acontece de perder uma posição por minutos ou até mesmo segundos.
  - Teste sua habilidade aqui: http://www.typingtest.com/ (~55-66 wpm)

- **2) Identificar rapidamente o tipo do problema:**
  - Verificar em qual categoria ele se encaixa.
  - Análisar se já resolveu algo parecido antes.
  - Resolver o mais fáceis primeiro.

Dicas interessantes

- **3) Faça a análise de algoritmos:**
  - Uma vez que pensou num algoritmo para resolver o problema, analise qual será seu comportamento quando a entrada for limite máximo.
  - Por exemplo, o algoritmo Insertion Sort tem complexidade no pior caso de $O(n^2)$, então quando $n >= 10000$, o algoritmo vai levar cerca de 10s para executar!
  - Para fazer essa análise, basta levar em conta que um computador de hoje em dia executa cerca de 10^7 instruções em 1s, com isso podemos fazer:

$O(n^2)$ → $10000^2 / 10^7 = 10^8 / 10^7 =$ **10s**

$O(nlgn)$ → $10000 * lg(10000) / 10^7 = 10^8 / 10^7 =$ **0.01s**

Dicas interessantes

- **4) Domine diferentes linguagens de programação:**
  - C/C++ é uma linguagem rápida e eficaz e tem a STL, mas mesmo assim devemos dominar Java. Pois Java tem APIs interessantes como: BigInteger, GregorianCalendar, Regex, etc.

- **5) Teste seu código:**
  - Analise a saída do seu código com a saída do exemplo.

```
./a.out < input > my_output
diff my_output example_output
```

  - Teste casos de testes chaves.

- **6) Pratique!**

Primeiros passos

- **Registro no URI Online Judge:**
  - Se registrar no site.
  - Configurar seu perfil (colocar a UNIPAMPA como universidade).
  - Começar a explorar os problemas. (comece pelos de iniciante para aprender como funciona).

- **Recomendações:**
  - Livro Introdução à Algoritmos – 3 edição – CLRS.
  - Livro Competitive Programming – 3 edição – Halim, S. Halim, F.
  - Curso de C e C++ avançado – Lima, A. Disponível em: https://allanlima.wordpress.com/category/curso-de-c-e-c-avancado/

Referências

[1] CORMEN, T.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Introdução à Algoritmos (Terceira Edição). MIT Press and McGraw-Hill. 2009.

[2] HALIM, S.; HALIM, F. Competitive Programming 3 (Terceira Edição). Handbook for ACM ICPC and IOI contestants. 2013.

Fim

Próximo dia:

Abstração dos problemas;

Estruturas de dados básicas;

Breve introdução sobre análise de complexidade;

Resolver problemas.

Dúvidas?

marcosvtreviso@gmail.com
facebook.com/marcos.treviso

Obrigado pela atenção!