Guia de Estratégia para a Maratona SBC

Passo Fundo, 22 de agosto de 2023

- O que levar para a Maratona?
- Estratégias para a competição
- Aquecimento (warm-up)
- Início da prova
- Lendo a prova
- Implementando
- Testando
- Submissão
- Discussão de problemas
- Dinâmica
- Minutos finais
- Pós-prova
- Dicas para ser mais competitivo

O que levar para a Maratona?

- Você pode levar todo e qualquer material **impresso**
  - Livros, cadernos, folhas impressas, trechos de código, ...
  - Evite livros didáticos - você deve estudar **antes** da prova
- ***Notebook*** (caderno, não *laptop*)
  - Conjunto de implementações de algoritmos conhecidos
  - Facilita na hora de lembrar como o algoritmo funciona
  - Usar implementações que já foram testadas e funcionam
  - Veja previamente quais algoritmos existem nele e entenda para que servem. Assim a equipe sabe quais armas tem em mãos na hora de pensar em uma solução

O que levar para a Maratona?

- ***Cheat sheets* (colinhas)**
  - Fórmulas matemáticas, funções, ou algoritmos prontos
- Leve material para planejar seu código **no papel** enquanto outros usam o computador
  - Lápis, caneta, borracha, régua, …
  - Bloco de notas, papel timbrado, papel quadriculado, …
- **Nada de material digital** como celulares, relógios, calculadoras, ou *pendrives* durante a competição (resulta em **desclassificação**)
  - Contudo, pode-se levar um *pendrive* **apenas** para salvar os códigos-fonte da equipe, **após o fim da prova**

O que levar para a Maratona?

- **Lanche para a competição**
  - Ajuda a pensar melhor e acalmar os nervos durante a prova. Isso vale para Maratona, OBI, e concursos em geral
- Procure levar lanches leves e práticos
  - Barras de cereal, balas, chocolates, frutas (ex.: banana)
  - Água mineral, com gás, ou suco natural
  - Evite refrigerantes, a não ser talvez para ajudar na digestão
- Algumas sedes permitem que os competidores comam no laboratório, porém **nem todas**
  - Saia para o lanche e volte assim que terminar, com calma

Estratégias para a competição

- Cada equipe possui apenas **um computador** para ser utilizado
  - “5 horas de máquina, 15 horas de humano”
- O tempo de submissão de cada problema é contado a partir do início da competição, e é um dos critérios de desempate
  - Cada submissão errada (ou correta) faz a diferença
- O número de problemas geralmente é elevado (aprox. 10 a 15)
  - **15** problemas na Primeira Fase de 2020
  - **14** problemas na Primeira Fase de 2021 e na de 2022
- Portanto, é preciso uma **estratégia** para maximizar a pontuação
  - **Como fazer isso?**

Aquecimento (warm-up)

- Procure tirar todas as suas dúvidas quanto ao ambiente computacional e às regras da competição, para aproveitar melhor o tempo quando a competição começar
- Use o editor de texto ou IDE de sua preferência. Caso não encontre, procure as ferramentas mais adequadas à equipe
- Teste o sistema de submissão eletrônica, as impressões de código-fonte, e principalmente, aproveite para errar à vontade
- **Depois do aquecimento**:
  - Alimente-se o suficiente, mas não a ponto de passar mal
  - Aproveite e revise o material de consulta antes da prova (inclusive este guia de estratégia)
  - Ouça o discurso motivacional do técnico

Início da prova

- Acomodem-se com calma no ambiente da competição
  - Deixem os materiais à mão e preparem-se para começar
- Abram o caderno de prova e procurem as questões mais fáceis
  - Um procura no começo, outro no meio, outro no fim
- Quem achar a questão mais fácil assume a máquina para codar
- Leia atentamente o enunciado e os exemplos de casos de teste
  - A interpretação de texto também faz parte da prova
- **Teste bem** o seu código antes de submeter
  - É preferível demorar uns minutos do que tomar penalidade
  - No entanto, tomar a dianteira do placar é sempre bom

Início da prova

- Invariavelmente, cerca de 20 a 40% da prova pode ser resolvida na primeira hora por competidores experientes
- **Problema**: como identificar os problemas fáceis o mais rápido possível, para diminuir a penalidade de tempo?
  - Solução prática: manter uma **tabela** de quem já leu cada problema, e sempre ler um problema que ninguém ainda leu (processo guloso)
- Ao identificar um problema fácil, quem o leu assume o computador para implementar a solução, enquanto o resto do time procura os outros problemas mais fáceis da prova

Anexo: Tabela de problemas (**exemplo**)

AC	Ordem	Escrito	Leitores	Complexidade	Resumo do problema
A					
B					
C					
D					
E					
F					
G					
H					
I					
J					

Como usar a tabela de problemas

- **AC**: risque (ou pinte) a letra de cada problema já resolvido
- **Ordem**: anote a ordem de solução dos problemas (a lápis)
  - Dê prioridade aos problemas mais fáceis da prova
- **Escrito**: descreve se tem código escrito para aquele problema
- **Leitores**: quem já leu o problema escreve seu nome ali
- **Complexidade**: qual a complexidade da solução pensada?
  - Notação **Big-O**: complexidades de **tempo** e de espaço
  - Imprimir PDF e levar: bigocheatsheet.com
- **Resumo do problema** - qual o assunto da questão? É **ad-hoc**?
  - Estruturas, PD, grafos, matemática, strings, geometria, ...

Resumo de Notação Big-O

Notação	**Complexidade**	**Exemplo**
O(1)	constante	Verificar se um número é par
O(log n)	logarítmica	Busca binária
O(n)	linear	Percorrer um vetor inteiro
O(n log n)	linearítmica	*Quicksort* (caso médio)
$O(n^2)$	quadrática	Dois **for**s aninhados; *bubble sort*, *insertion sort*, ...
O(n^k)	polinomial	Algoritmos com complexidade $O(n^2)$, $O(n^3)$, ...
O(k^n)	exponencial	Resolver o problema do caixeiro viajante com PD
O(n!)	fatorial	Gerar todas as permutações de um vetor ou string

Lendo a prova

- Adote uma estratégia para que todas as questões tenham sido lidas por pelo menos um integrante do time nos primeiros 40 minutos da competição
- Ao ler um problema, sublinhe ou destaque as partes mais importantes, detalhes que podem ser *trickies* (pegadinhas)
  - A interpretação de texto também faz parte da prova
- Se tiver incerteza em relação a algo, leia novamente. Se a dúvida permanecer, envie uma *Clarification* **imediatamente**.

Lendo a prova

- Veja a página de *Clarifications* periodicamente, principalmente no início da competição. Erros no enunciado são comunicados ali, e outras pessoas podem ter tido a mesma dúvida que você tem.
- Ao final das primeiras duas horas, é recomendável que todos os integrantes tenham lido todos os problemas, para facilitar a troca de ideias.

Implementando

- Apenas uma pessoa no computador por vez
- Tente trabalhar em equipe, principalmente no estágio de criação do algoritmo. Depois de ter o algoritmo pronto, a parte de implementação normalmente fica mais “individual”
  - Se o algoritmo pensado estiver correto, a implementação normalmente é rápida e não tão complicada
- O resto do time continua lendo outros problemas
- Determine um tempo máximo que uma pessoa pode ficar continuamente no computador (ex.: algo entre 10 e 20 minutos)

Implementando

- Caso você ache uma solução e o computador esteja ocupado, planeje melhor o código no papel, para implementar mais rapidamente quando for sua vez
- Não se preocupe em fazer um código elegante e limpo, reutilizável e todo comentado
  - O código em si não vai ser avaliado, somente a sua saída
- Cuidado ao utilizar algoritmos de força bruta, pois os testes poderão ser grandes, e o tempo limite pode estourar

Testando

- Gaste algum tempo testando o seu programa
- Uma submissão precoce pode fazer com que algum erro ou diferença na saída passe despercebido, fazendo com que a equipe seja **penalizada**.
- Além dos testes de exemplo, faça outros testes, principalmente com os "*corner cases*". Ou seja, casos que envolvem os limites estabelecidos no enunciado do problema.
- Testar mais vai fazer você perder 2 ou 3 minutos, mas enviar uma solução errada faz você perder 20 minutos de pontuação.

Testando

- Os exemplos de entrada mostrados nos problemas servem somente para ajudar a visualização. Os dados de entrada utilizados na verificação do programa serão bem diferentes.
- Também procure por **exceções** (ex.: o que aconteceria se entrasse com um número negativo?), descontando os casos que estão **explicitamente excluídos** pela definição do problema.
- Procure ler com atenção o que o problema pede. Se você esquecer um espaço ou quebra de linha na saída, o seu programa vai ser rejeitado (*presentation error*, ou *WA* mesmo).

Submissão

- Ao terminar de implementar e testar uma solução, faça a submissão
  - O resultado pode levar de alguns segundos até alguns minutos para chegar
- **Sempre** faça a impressão de seu código-fonte imediatamente após a submissão
  - Alguém vai ter que usar o computador depois de você
  - A solução pode estar errada, e você poderá analisar o seu código-fonte no papel

Submissão: vereditos dos juízes

Uma vez submetido, seu código recebe um dos seguintes pareceres:

- **YES** (ou seja, Accepted)
- `NO - Compilation error`
- `NO - Runtime error`
- `NO - Time limit exceeded`
- `NO - Wrong answer`
- `NO - Contact staff`
- `NO - Class name mismatch`
- `NO - Wrong language`
- `NO - Problem mismatch`

Submissão

- Caso a submissão esteja errada: **Não entre em pânico**
  - Analise o seu código impresso
  - Leia novamente o enunciado. Preste atenção nos limites do problema e *overflows*.
  - Caso esteja obtendo *RUNTIME ERROR*, gere várias entradas aleatórias (usando Python, talvez) e use o **gdb** (ou depurador equivalente) para encontrar rapidamente o erro.
  - Ao achar o erro, se for uma modificação rápida (2 a 3 minutos), pode interromper quem está no computador no momento

Submissão

- Veja na tabela quem já leu o problema e descreva o algoritmo para essa pessoa; **nunca** faça isso antes da pessoa ler o problema e pensar numa solução
- Não fique no problema por muito tempo
- Desconfie dos limites. Use **assert**s (`#include<assert.h>`)
- Faça ***printf*** ou ***cout*** de variáveis ao longo do código-fonte
  - ***fprintf****(stderr, ...)* e ***cerr*** imprimem na saída padrão de erros. Isso facilita a distinção da saída padrão do programa.
- Resolvido o problema, não esqueça de **remover** os *printf*s, *cout*s e afins utilizados para debugar!

Discussão de problemas

- Discutir problemas pode ser benéfico ou destrutivo; depende de como se faz
- Apresentar uma solução para uma pessoa que não pensou numa outra solução “mata” a capacidade dela de pensar em outra abordagem
- Discutir com alguém que tem outra solução faz com que um aponte defeitos na solução do outro, e venha a surgir uma solução que funciona. Erros de interpretação são consertados aqui.
    - Daí a importância de todos terem lido todos os problemas nas primeiras duas horas.

Discussão de problemas

- **Jamais** brigue com seus colegas de equipe durante a competição (e nem com os seus adversários)
  - Isso pode desconcentrar e afundar o seu time, não só moralmente, como também no placar
- Se encontrar uma solução que pareça errada ou ineficiente e não sair disso, tente essa solução.
  - Às vezes a solução do problema é força bruta mesmo. Novamente, preste atenção no enunciado e nos limites.

Dinâmica

- Manter **sempre** atenção no placar e nas *clarifications*.
- Durante a competição, não é recomendável que duas pessoas trabalhem no mesmo problema ao mesmo tempo.
- Não fique de cabeça quente. Se precisar, saia da mesa ou da sala por um tempo para beber ou comer algo.
  - Não esqueça de chamar um “melancia” para isso, pois sair sozinho da sala implica em **desistência**
- Pessoas comemoram fazendo barulho, não se abale :-)

Minutos finais

- Na última hora de competição, o placar é congelado. Os balões, no entanto, continuam chegando.
- Nos últimos 30 minutos, não chegam mais balões, mas você ainda pode conferir o veredito das suas submissões.
- Na última hora, recomenda-se que o time se concentre somente em um problema de cada vez. Se organizem para ver qual é o problema mais promissor e façam *pair programming*
- Nos últimos 5 minutos, não é hora de economizar submissões. Façam leves modificações e submetam (mas sem *spam*)

Pós-prova

- Aguardem a revelação do placar e torçam (muito) pelo melhor
  - Em caso de classificação ou título, comemorem muito \o/
- Durante a competição, você pode acabar cometendo erros estúpidos. Além de ficar revoltado(a), é muito importante discuti-los, principalmente erros de estratégia de equipes.
- Procurem entender o que estavam fazendo de errado durante a competição
- Façam em casa os problemas que deixaram de fazer na prova

Pós-prova

Mantenham a cabeça erguida, **sempre**! Muita gente participa da Maratona de Programação uma vez, vai mal, e desiste. Quem não se desanima e segue treinando, corre os riscos de:

- Aprimorar muito a capacidade de *problem solving*;
- Aprimorar enormemente conhecimentos de matemática (álgebra, geometria, cálculos numéricos, combinatória, ...);
- Aprender a escrever código com rapidez e precisão;
- Aprimorar a capacidade de encontrar e resolver bugs;

Pós-prova

- Manter o ritmo atual para chegar ainda mais forte no próximo ano, se preparando antes de todas as equipes adversárias;
- Conhecer pessoas muito talentosas que gostam das mesmas coisas que você;
- Construir um *networking* profissional e acadêmico valioso;
- Conseguir colocações em grandes empresas de tecnologia facilmente;
- E não menos importante: **viajar e comer de graça \o/**

Dicas para ser mais competitivo

- Busque digitar código mais rapidamente (!)
  - Quando você consegue resolver o mesmo número de problemas que seu adversário, a diferença se dará pela suas habilidades de programação (produzir código conciso e robusto)... e em último caso, pela velocidade de digitação
- Pratique quando estiver programando, e também com os links:
  - https://www.typingtest.com
  - https://typing.io
  - https://typeracer.com
- **Lembre-se**: você vai ter de usar o teclado da sede, e não o seu

Dicas para ser mais competitivo

- Domine pelo menos a sua principal linguagem de programação
  - C/C++ ainda é a preferida para a Maratona e para o ICPC
  - Python é ótima para iniciantes e para alguns problemas
  - Java é lenta e verbosa, mas possui bibliotecas poderosas
  - Kotlin é uma alternativa menos verbosa à linguagem Java
- Resolva muitos exercícios de programação em diversos portais
  - BeeCrowd, Codeforces, OnlineJudge.org, SPOJ, Kattis, ...
- E sempre procure trocar ideias com competidores experientes, tanto da sua universidade, como de outras instituições :-)
  - Grupo da Maratona no Telegram: t.me/MaratonaBrasil

Bons treinos e boa sorte!
Que venha a Maratona :-)