Maratona de Programação

Introdução e Estratégias Básicas

Por Felipe “Malkava” Sodré

Maratona x OBI

- Competição em Equipe de 3 pessoas
- Resultados em “Tempo Real”
- Material de Apoio
- Tempo é levado em conta
- C, C++ ou JAVA
- Balões :)

Maratona

- 3 Fases
  - Sub-Regional (20 de Setembro – Algum lugar do interior de SP)
  - Regional (Final Nacional – 14 e 15 de Novembro – Vila Velha-ES)
    - Máximo de 2 times por universidade
  - Final Mundial (2009 - Estocolmo – Suécia)
    - Máximo de 1 time por universidade

Maratona

- Equipes são formadas por 3 estudantes da mesma universidade.
- Um competidor não pode participar se:
  - Já participou de 2 finais mundiais
  - Já participou de 5 regionais
  - Começou seus estudos em 2003 ou antes e nasceu em 1984 ou antes

Competição em Equipe

- Cada equipe possui apenas um computador para ser utilizado
- O tempo de submissão de cada problema é contado a partir do inicio da competição, e é levado em conta para o desempate
- O número de problemas geralmente é elevado
- Conclusão: Uma estratégia se faz necessária para maximizar a pontuação

Início da prova

- Invariavelmente, cerca de 30 a 40% da prova pode ser resolvida na primeira hora por competidores com alguma experiência.
- Problema: Como identificar tais problemas o mais rápido possível, para diminuir penalidade de tempo?

Início da prova

- Solução prática: Manter uma tabela de quem já leu qual problema, e sempre ler um problema que ninguém ainda leu (processo guloso).
- Ao identificar um problema facilmente resolvível, quem o leu assume o computador para implementar a solução, enquanto o resto da equipe continua procurando problemas fáceis.

Início da prova

- Problema: Como saber se um problema é fácil?
- Solução prática: Após ler e entender o problema, pense nele por 2 ou 3 minutos. Se não conseguir pensar em nada, provavelmente não é fácil.
- Solução 2: Balões

Lendo a prova

- Adote uma estratégia para que todas as questões tenham sido lidas por pelo menos um integrante do time na primeira meia-hora da competição
- Ao ler um problema, sublinhe as partes mais importantes, detalhes que podem ser *trickies.*
- Se ficar incerto com relação a algo, leia novamente. Se a dúvida permanecer, envie um Clarification imediatamente.
- Veja a página de clarifications periodicamente, principalmente no início da competição. Erros no enunciado são comunicados alí, e outras pessoas podem ter tido a mesma dúvida que você tem.
- Ao final das duas primeiras horas, é recomendável que todos os integrantes tenham lido todos os problemas, para facilitar a troca de idéias.

Implementando

- Apenas uma pessoa no computador por vez
- O resto do time continua lendo outros problemas
- Caso você ache uma solução e o computador estiver ocupado, planeje melhor o código no papel, para implementar mais rapidamente quando for sua vez.
- Determine um tempo máximo que uma pessoa pode ficar continuamente no computador (10 a 15 minutos)

Testando

- Além dos testes de exemplo, faça outros testes, principalmente com os “corner-cases”.
- Testar mais vai fazer você perder 2 ou 3 minutos, mas enviar uma solução errada faz você perder 20 minutos de pontuação.

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 sua solução logo após a submissão
  - Alguém vai ter que usar o computador depois de você
  - A solução pode estar errada, e você pode analisar o código no papel

Submissão

- Caso a submissão esteja errada:
  - DON'T PANIC
  - Analise o seu código impresso
  - Leia novamente o enunciado (preste atenção aos limites e overflows)
  - Caso esteja obtendo Runtime Error, gere várias entradas aleatórias e use o gdb para rapidamente encontrar o erro.
  - Ao achar o erro, se for uma modificação rápida (algo em torno de 2 a 3 minutos), pode interromper quem está no computador no momento.
  - Veja na tabela quem já leu o problema e descreva o algoritmo para ele (**NUNCA** faça isso antes da pessoa ler o problema e pensar numa solução antes)
  - Não fique no problema por muito tempo
  - Desconfie dos limites. Use asserts.

Discussão

- 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.
- Se encontrar solução que pareça errada ou ineficiente e não saír disso, tente essa solução.

Dinâmica

- Manter sempre atenção no placar e nos clarifications
- O placar mostra quais problemas estão sendo feitos, o que geralmente indica seu grau de dificuldade
- A pagina de clarifications mostra erros de enunciado e dúvidas de outras pessoas, que podem facilitar sua vida.
- 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 por um tempo para beber e comer algo.
- Pessoas comemoram fazendo barulho, não se abale :)

Final de Jogo

- Na última hora de competição, o placar não é mais atualizado (os balões, no entanto, continuam chegando)
- Nos últimos 10 minutos, é jogo às cegas. Não é possível saber se a submissão está certa, até a hora da premiação.
- 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 (um programa, e os outros vão vendo se a implementação está certa).

Material de Apoio

- Você pode levar todo e qualquer material impresso (livros, folhas impressas da internet, trechos de código, pornografia (exceto infantil))
- Notebook
  - Trata-se de um conjunto de implementações de algoritmos conhecidos. Isso diminui o tempo que você precisa pra lembrar de como o algoritmo funciona, e também o tempo de depuração (geralmente tais implementações já foram testadas em outros problemas e funcionaram)
  - Antes de usar, veja quais algoritmos existem nele e entendam para que servem. Assim você sabe quais armas tem em mãos ao pensar numa solução

Post-Mortem

- Durante a competição, você vai cometer erros estúpidos. Além de ficar revoltado, é muito importante discutí-los, principalmente erros de estratégia de equipes.
- Esta apresentação é resultado de anos de Post-Mortems
- Façam os problemas que deixaram de fazer na prova

Maratona é legal !

- Muitas pessoas participam da Maratona uma vez, vão mal, e desistem. Quem não faz isso, corre o risco de:
  - **Aprimorar enormemente a capacidade de problem solving (isto é, ter ferramentas e saber utiliza-las pra resolver um problema, de modo geral)**
  - **Aprimorar enormemente conhecimentos de matemática (algebra, geometria, cálculo, métodos numéricos, combinatória)**
  - **Conhecer muita gente talentosa que gosta das mesmas coisas que você.**
  - **Aprimorar a velocidade e *acurracy* na hora de escrever código.**
  - **Aprimorar a capacidade de encontrar bugs**
  - **Construir um networking acadêmico e profissional muito valioso.**
  - **Conseguir colocações em grandes empresas de tecnologia muito mais facilmente (como Google, Microsoft, Yahoo, IBM, entre outras).**
  - **Sofrer tietagem. Num futuro próximo, competidores serão sex-symbols.**
  - **E o mais importante: Viajar e comer de graça :)**