Maratona SBC de Programação 2026

Ambiente de Testes e Sistema de Submissão

1 Informações sobre o Ambiente de Testes

1.1 Ambiente

O sistema de correção das submissões será executado na distribuição `Ubuntu GNU/Linux 24.04 LTS amd64`, com os seguintes compiladores e interpretadores configurados:

```
C: gcc (Ubuntu 13.3.0-6ubuntu2~24.04.1) 13.3.0
C++20: g++ (Ubuntu 13.3.0-6ubuntu2~24.04.1) 13.3.0
Python 3: PyPy3, Python 3.9.18 (7.3.15+dfsg-1build3, Apr 01 2024, 03:12:48)
Java: javac 21.0.11
Kotlin: kotlinc-jvm 2.1.20 (JRE 21.0.11+10-1-24.04.2-Ubuntu)
```

1.2 Linguagens aceitas

As linguagens aceitas pelo juiz e as extensões de arquivo correspondentes são:

Linguagem aceita	Extensão do arquivo
C	`.c`
C++20	`.cpp`
Java	`.java`
Kotlin	`.kt`
Python 3 (PyPy3)	`.py`

A linguagem da submissão é identificada pela extensão do arquivo que você envia, portanto envie o seu código-fonte com a extensão correta. Um arquivo cuja extensão não apareça na tabela acima será recusado, e renomear um arquivo não muda a linguagem com que ele é julgado: o servidor confere a linguagem novamente quando a submissão chega.

1.3 Limites de memória

Todas as submissões, em todas as linguagens, estão limitadas a um total de 1GB de RAM e a uma pilha (*stack*) de no máximo 128MB. O uso de memória de uma submissão é medido pelo seu pico de memória residente (a memória de fato utilizada), e não pelo tamanho do espaço de endereçamento que ela reserva.

Observação: o limite de 1GB de memória especificado inclui a memória usada pelo ambiente de execução, e isso precisa ser levado em conta. Por exemplo, em Java, Python e Kotlin é preciso considerar que o limite de memória inclui a memória usada pelo ambiente de execução, que pode ser de vários megabytes. Em C e C++ a memória usada pelo runtime da linguagem é relativamente pequena, mas ainda assim deve ser considerada.

1.4 Limites de tempo

Antes da competição, os juízes terão resolvido todos os problemas em linguagens de pelo menos dois dos três grupos de linguagens distintos (C/C++, Java/Kotlin e Python3). Os limites de tempo de cada problema serão calculados com base no tempo de execução dessas soluções.

Os limites de tempo de cada problema também estarão disponíveis na interface do MOJ.

1.5 Outros limites

```
Tamanho do arquivo-fonte: 100KB
Tamanho do binário gerado / da saída do programa: 128MB
Tempo de compilação: 10 segundos
```

1.6 Compilação e execução

É fortemente recomendado que você compile e execute suas soluções usando os comandos especificados nas seções por linguagem a seguir. Embora esses comandos não imponham limites de memória ou de tempo como o juiz faz, eles correspondem ao ambiente que os juízes usarão para compilar e executar as submissões.

1.6.1 C/C++20

- O arquivo executável gerado pelo compilador será executado para gerar a saída da submissão.
- Seu programa deve retornar zero, executando, como último comando, `return 0` ou `exit(0)`.
- Sabe-se que, em problemas com entradas grandes, os objetos de `iostream` podem ser lentos porque, por padrão, usam um buffer sincronizado com a biblioteca stdio. Se você quiser usar `cin` e `cout`, é aconselhável desabilitar essa sincronização. Isso pode ser feito chamando `std::ios::sync_with_stdio(false);` no início da sua função main. Note que, nesse caso, deve-se evitar usar `scanf` e `printf` no mesmo programa, pois, com buffers separados, podem ocorrer comportamentos inesperados.
- Compile soluções em C com:

  ```
  gcc -std=gnu11 -O2 -lm -static {submitted_file} -o main
  ```
- Compile soluções em C++ com:

  ```
  g++ -std=c++20 -O2 -lm -static {submitted_file} -o main
  ```
- Execute soluções em C/C++ com:

  ```
  ./main
  ```

1.6.2 Java

- A classe principal compilada será executada para gerar a saída da submissão.
- NÃO declare um `package` no seu programa Java; se você declarar um package o programa não será executado no MOJ.
- O nome do arquivo submetido deve coincidir com o nome da sua classe pública principal. Se o seu arquivo se chama `Klass.java`, a classe pública que contém o método main deve ser `Klass`. Se você não tiver uma classe pública contendo o método main, ou não seguir essa convenção, poderá receber um Compilation Error.
- Sabe-se que, em problemas com entradas grandes, o `Scanner` pode ser lento demais. É aconselhável usar E/S com buffer (por exemplo, `BufferedReader` e `PrintWriter`).
- Compile soluções em Java com:

  ```
  javac {submitted_file}
  ```
- Execute soluções em Java com:

  ```
  java -Xms10m -Xmx1024m -Xss128m {class_name}
  ```
- Note que os tamanhos de heap e de pilha acima já são definidos para você pelo juiz, de acordo com os limites de memória da Seção 1.

1.6.3 Python

- O arquivo-fonte principal será executado pelo interpretador PyPy3 do Python3 para gerar a saída da submissão.
- Programas Python passarão por uma "verificação de sintaxe" ao serem submetidos; programas que falharem nessa verificação receberão o veredito "Compilation Error".
- Note que as submissões em Python serão executadas com o interpretador **PyPy3**, e não com o CPython tradicional. Isso geralmente resulta em melhor desempenho, mas fique atento a eventuais diferenças sutis de comportamento.

- "Compile" (verifique a sintaxe de) soluções em Python com:
  `pypy3 -m py_compile {submitted_file}`
- Execute soluções em Python com:
  `pypy3 {submitted_file}`

1.6.4 Kotlin

- A classe principal compilada será executada para gerar a saída da submissão.
- NÃO declare um `package` no seu programa Kotlin; se você declarar um package o programa não será executado no MOJ.
- Não há convenção de nomes para o arquivo submetido: o ponto de entrada é a `fun main()` de nível superior do arquivo que você submeter.
- Compile soluções em Kotlin com:
  `kotlinc -d .  {submitted_file}`
- Execute soluções em Kotlin com:
  `kotlin -J-Xms10m -J-Xmx1024m -J-Xss128m {file_name}Kt`
- Note que os tamanhos de heap e de pilha acima já são definidos para você pelo juiz, de acordo com os limites de memória da Seção 1.

2 Instruções para o Uso do sistema MOJ

Os vereditos que você pode receber dos juízes são:

- Not Answered Yet
- Accepted
- Wrong Answer
- Time Limit Exceeded
- Runtime Error
- Compilation Error

Numa prova, o veredito é tudo o que você recebe: sem o teste que falhou, sem diff, sem nota parcial. Isso é de propósito, pois os testes são material de prova.

Se um veredito parecer resultado de circunstâncias imprevisíveis, use a aba "Clarification" e identifique a submissão (problema e horário da submissão) para obter mais esclarecimentos.

Seu programa pode ser executado sobre múltiplos arquivos de entrada. Note que isso significa que, se o seu programa tiver mais de um erro (digamos, "Time Limit Exceeded" e "Wrong Answer"), você pode receber qualquer um desses erros como veredito.

Tenha em mente as seguintes observações sobre o julgamento:

- Se a sua solução demorar demais para compilar ou produzir um executável/bytecode grande demais, você receberá o veredito "Compilation Error".
- Não existe "Presentation Error" nem "Format Error". Se você escrever errado a palavra "impossible", por exemplo, e o problema exigir essa palavra como saída, então sua submissão será julgada como "Wrong Answer".
- A formatação da saída deve seguir a saída de exemplo do enunciado do problema, embora espaços em branco extras, dentro do razoável, sejam aceitáveis. Por exemplo, se você imprimir milhares de espaços em branco dentro do limite de tempo do problema, isso seria julgado como Wrong Answer; porém, um espaço extra no fim de uma linha ou entre tokens, ou uma linha em branco extra, é aceitável.

- Em problemas com saída em ponto flutuante, os juízes aceitam qualquer resposta que satisfaça as restrições descritas no enunciado. Salvo indicação em contrário, essas restrições são dadas como uma tolerância absoluta e/ou relativa, de modo que sua resposta não precisa coincidir exatamente com os dígitos do exemplo, desde que fique dentro da tolerância. Se o valor correto é x e você imprime y, o juiz aceitará sempre que a condição do enunciado for satisfeita (por exemplo, $|x - y| \leq \varepsilon$ para uma tolerância absoluta ε, ou $\frac{|x-y|}{|x|} \leq \varepsilon$ para uma tolerância relativa). Por exemplo, se a tolerância é 10^{-9} e a resposta correta é 1, saídas como 1, 1.000000 ou 0.999999999 são todas aceitáveis.
- Quando o enunciado especifica que múltiplas saídas são aceitáveis (por exemplo, qualquer caminho mínimo ou qualquer ordenação que satisfaça as restrições dadas), os juízes verificarão que a sua saída obedece ao formato e aos requisitos pedidos; não se espera que ela coincida exatamente com a saída de exemplo.
- Se o seu programa gerar saída em excesso, sua submissão receberá o veredito "Runtime Error". Note que todo conteúdo escrito na saída de erro padrão (*stderr*) também conta para esse limite. Portanto, um excesso de mensagens de depuração pode causar erros de execução. Se você encontrar um Runtime Error inesperado, considere reduzir os logs no *stderr*.

Penalidades

Cada submissão a um problema que receber um veredito diferente de "`Accepted`" antes do primeiro veredito "`Accepted`" para o mesmo problema acarretará uma penalidade de tempo de 20 minutos, que é somada ao tempo total do time. Não há tempo de penalidade para problemas não resolvidos.
A exceção a essa regra é "`Compilation Error`", que não tem penalidade de tempo associada.

Tempos de Resposta

Note que o tempo necessário para produzir um veredito varia dependendo do problema, do veredito e do momento da competição em que a submissão é recebida. Como as soluções são julgadas simultaneamente, isso significa que você pode receber os vereditos das suas submissões fora de ordem. Além disso, em alguns casos, é necessária uma verificação manual dos juízes. Dependendo do que precisa ser verificado e do número de submissões que exigem verificação manual, atrasos até mesmo da ordem de minutos são esperados.

Se você tiver mais dúvidas sobre o sistema de correção, acesse a documentação detalhada do competidor disponível na sua interface web.