Setup JVM dan MEMSTACK di DOMjudge

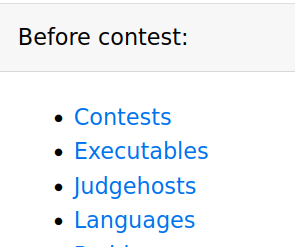
# Pendahuluan

Di bahasa pemrograman java, untuk eksekusi nya dibutuhkan sesuatu yang namanya Java Virtual Machine (JVM). Ini berfungsi sebagai runtime/tempat bagi code java untuk berjalan. Di lomba pemrograman, pada soalnya terdapat *memory limit* untuk setiap eksekusi solusi peserta. Masalahnya, untuk eksekusi solusi dengan java, JVM nya akan memakan sebagian memori sehingga *memory limit* ini bisa terhabis banyak untuk JVM nya saja. Akibatnya, hanya tersisa sedikit untuk program peserta nya, yang akan memberikan kerugian bagi pengguna java dibandingkan pengguna bahasa lain.

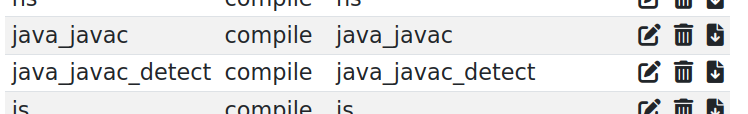
# Solusi

Solusinya, kita bisa memodifikasi batas memori yang digunakan untuk JVM dan stack pada script kompilasi java di website DOMjudge nya itu sendiri. Untuk itu, berikut tahap melakukannya:

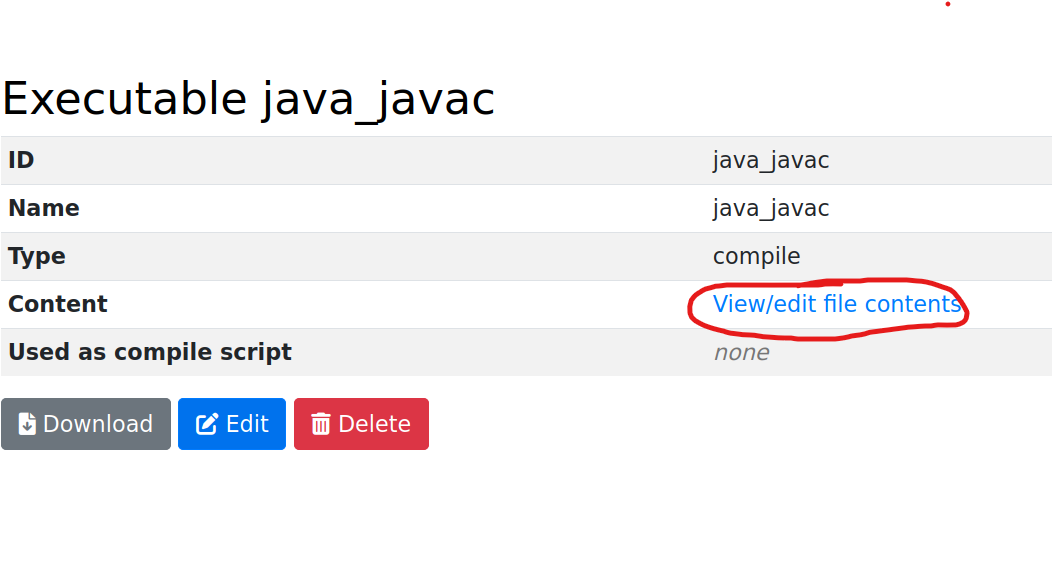
1. Pergi ke halaman “Executables” untuk mengakses script compile tersebut.

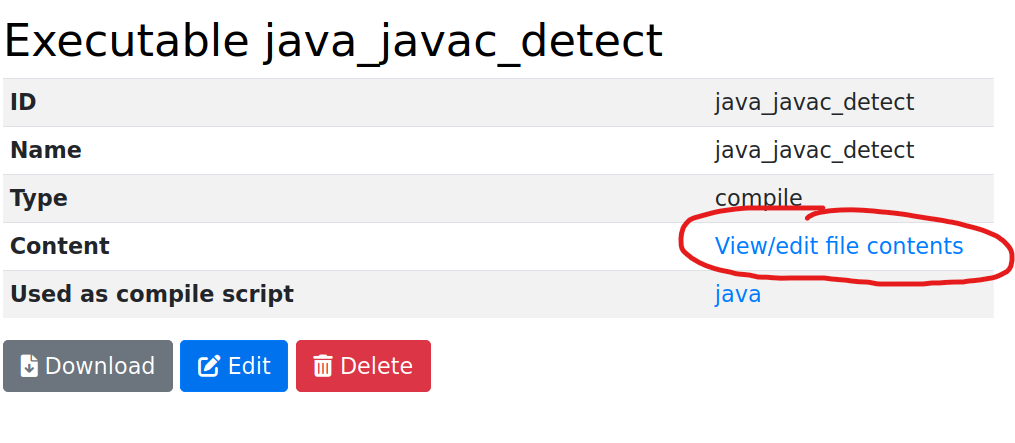


1. Di dalamnya, terdapat script-script khusus yang digunakan oleh DOMjudge, salah satunya “javac”. Batas memori untuk JVM dan stack dapat kita atur pada 2 file yaitu “java\_javac” dan “java\_javac\_detect”.



1. Klik “View/edit file contents” pada keduanya.



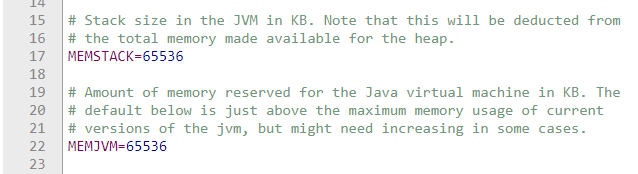


1. Bagian yang menyimpan informasi memori JVM dan stack tersebut dapat ditemukan di file run. Inilah file yang akan kita edit.





1. Di dalam file run tersebut, terdapat variabel MEMJVM dan MEMSTACK. Nilai default yang diberikan DOMjudge yaitu 65536 untuk keduanya.



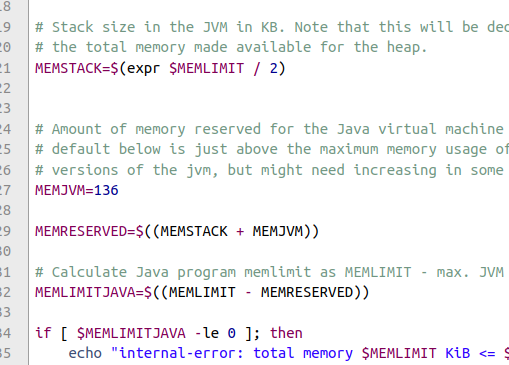
1. Kita akan mengubah nilai tersebut, dan tekan “Save files” atau “Submit” di bagian paling bawah setelahnya. Lakukan ini untuk kedua file run yaitu di script “java\_javac” dan “java\_javac\_detect”. Ubah nilainya ke nilai apa? Lihat bagian selanjutnya.

# Nilai MEMSTACK dan MEMJVM Optimal

Singkatnya, setelah dilakukannya ekplorasi dan eksperimen, nilai yang optimal yaitu:

* + MEMJVM: 136
  + MEMSTACK: (memory\_limit\_soal)/2

Ubah seperti berikut, untuk kedua file run:



Catatan Eksplorasi/Eksperimen

Beberapa hal yang saya temukan yaitu:

* Nilai minimum memory yang dibutuhkan MEMJVM adalah 136 (kB).
* Nilai MEMSTACK sebaiknya disesuaikan berdasarkan memory limit soal (MEMLIMIT). Hal ini karena DOMjudge akan memberikan error jika:

memory limit soal <= MEMSTACK + MEMJVM

Apabila MEMSTACK/MEMJVM di-set secara tetap dengan angka yang besar, maka soal dengan memory limit kecil akan menyebabkan error.

Kemudian ada beberapa hal yang saya temukan terkait MEMSTACK ini yaitu:

* Stack banyak digunakan dalam algoritma rekursif.
* Pada java, dikenal MEMHEAP (dalam file tidak disebutkan). MEMHEAP ini adalah memory dinamis yang digunakan untuk array, objek, dsb. Nilai dari MEMHEAP ini berdasarkan memori yang tersisa setelah MEMSTACK dan MEMJVM diinisiasi yaitu:

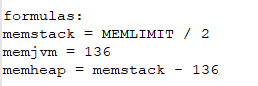
Memory limit soal - MEMSTACK - MEMJVM

* Perlu menjadi catatan apabila MEMHEAP pada saat compiler meng-compile program tidak cukup untuk menjalankan program, maka akan ada message error: RTE

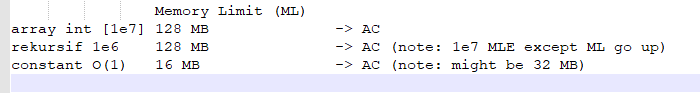
out of memory: java heap space

Jika saat program berjalan dan ternyata memory heap tidak cukup, maka program akan RTE dengan message exit (berbeda seperti yang sebelumnya).

Dengan pertimbangan - pertimbangan di atas, saya memutuskan untuk membuat inisiasi nilai MEM sebagai berikut



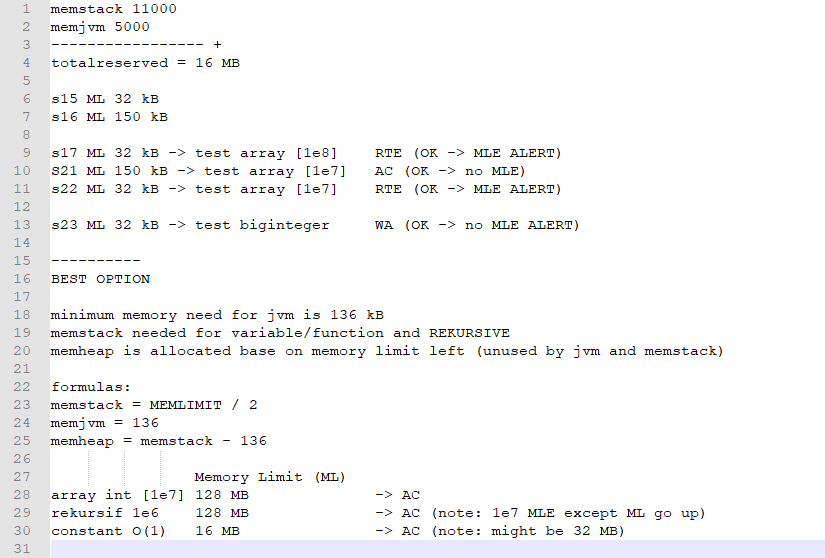
Dengan aturan (rules) di atas, beberapa hal ini possible untuk dilakukan

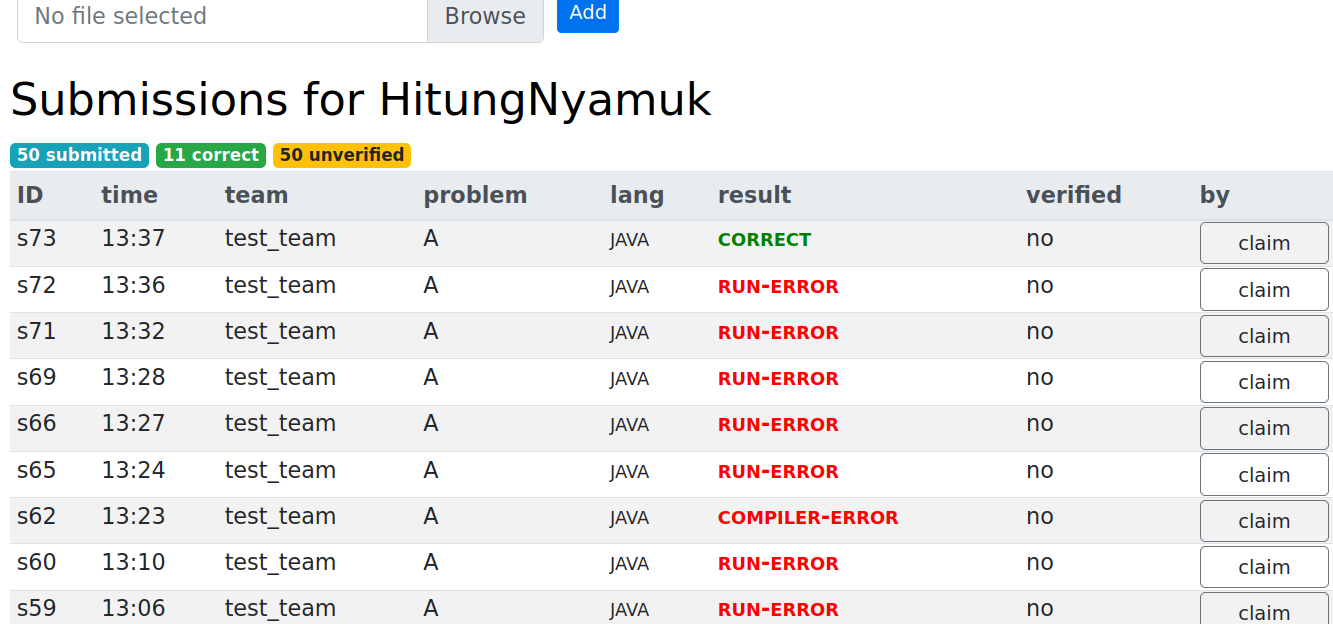


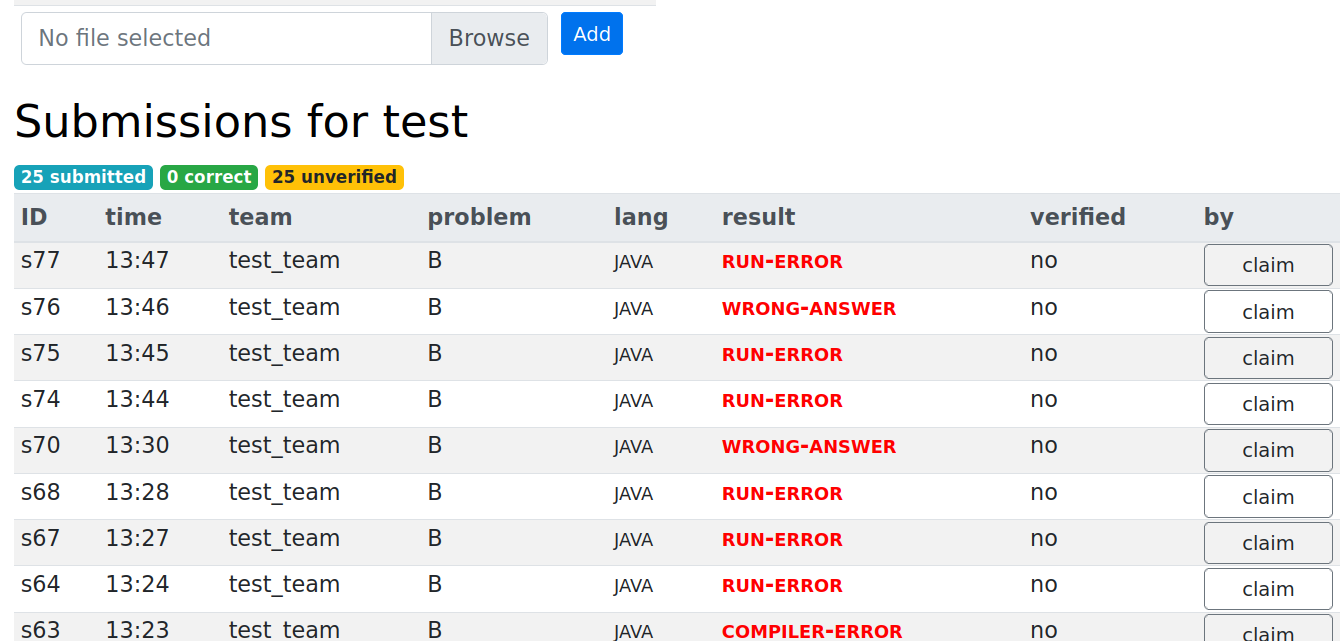
Sehingga sejauh ini, pilihan terbaik untuk solusi JVM adalah menggunakan rumus di atas.

Berikut saya lampirkan laporan eksplorasi yang saya sudah lakukan

Gambar di bawah adalah percobaan saya untuk tiap kasusnya (perubahan memory limit, MEMJVM, dan MEMSTACK)

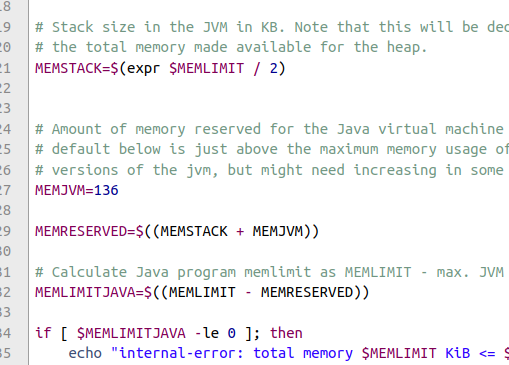




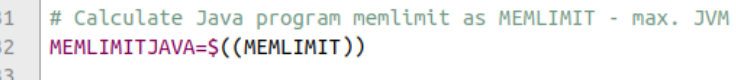


NOTE:

Solusi alternatif mungkin hanya dengan mengubah baris:



menjadi



pada kedua script run di executable java\_javac dan java\_javac\_detect.

Dengan setup tersebut, berikut hasil eksekusi program:

* Solusi rekursif (faktorial) dengan kedalaman memakan memori 11 MB.
* Solusi rekursif (faktorial) dengan kedalaman memakan memori 11 MB.
* Solusi rekursif (faktorial) dengan kedalaman memakan memori 16 MB.
* Solusi rekursif (faktorial) dengan kedalaman memakan memori 37 MB.
* Solusi dengan inisialisasi array int sebesar memakan memori sebesar 54 MB.
* Solusi dengan inisialisasi array int sebesar memakan memori sebesar 54 MB.
* Solusi dengan inisialisasi array int sebesar memakan memori sebesar 55 MB.
* Solusi dengan inisialisasi array int sebesar memakan memori sebesar 60 MB.
* Solusi dengan inisialisasi array int sebesar memakan memori sebesar 94 MB.

Pada DOMjudge yang baru, berikut hasil eksekusi program:

* Solusi dengan arrayList int sebesar diisi dengan nilai :
* memakan memori sebesar 14.10 MB.
* memakan memori sebesar 14.35 MB.
* memakan memori sebesar 17 MB.
* memakan memori sebesar 43 MB.
* memakan memori sebesar 337.53 MB.
* Solusi dengan array int[] sebesar diisi dengan nilai :
* memakan memori sebesar 10.59 MB.
* memakan memori sebesar 10.60 MB.
* memakan memori sebesar 10.97 MB.
* memakan memori sebesar 14.17 MB.
* memakan memori sebesar 49.73 MB.

Perbandingan dengan C++:

* Solusi dengan vector<int> sebesar diisi dengan nilai memakan memori sebesar 64.40 MB.
* Solusi dengan array int sebesar diisi dengan nilai memakan memori sebesar 38.37 MB.