1 **/\***

2 **\* Ovo je prijepis mog hibridnog algoritma razvrstavanja da se može**

3 **\* pokrenuti na JavaScriptinoj virtualnoj mašini. Kako bi se AEC-om mogla**

4 **\* ciljati JavaScriptina virtualna mašina, napravio sam novi compiler.**

5 **\* Ovaj puta je pisan u C++-u, radio sam i novi parser i novi tokenizer.**

6 **\* Također sam malo promijenio sintaksu, da omogućim pisanje čistijeg koda**

7 **\* te da podržavam različite vrste podataka (prije je AEC podržavao samo**

8 **\* 32-bitne decimalne brojeve). Novi compiler proizvodi WebAssembly,**

9 **\* standardizirani oblik JavaScriptinog bytecodea. Izvorno je WebAssembly**

10 **\* bio Mozillin standard, ali danas ga podržavaju gotovo sve JavaScriptine**

11 **\* virtualne mašine. Ovdje ciljamo primarno na NodeJS, JavaScriptinu**

12 **\* virtualnu mašinu koju razvija Google i primarno je namijenjena da se**

13 **\* vrti na serverima (no može se pokrenuti i na veoma slabim računalima).**

14 **\*/**

15

16 **// Uvezimo prvo neke funkcije iz JavaScripta, koje će nam trebati...**

17 **Function** daj\_velicinu\_niza**()** **Which** **Returns** **Integer32** **Is** **External;**

18 **Function** kopiraj\_niz\_na\_adresu**(**

19 **Integer32Pointer** adresa**)** **Which** **Returns** **Nothing** **Is** **External;**

20 **Function** printString**(CharacterPointer** str**)** **Which** **Returns** **Nothing** **Is** **External;**

21 **Function** printInteger**(Integer32** n**)** **Which** **Returns** **Nothing** **Is** **External;**

22 **Function** printFloat**(Decimal32** x**)** **Which** **Returns** **Nothing** **Is** **External;**

23 **Function** pocni\_mjerenje\_vremena**()** **Which** **Returns** **Nothing** **Is** **External;**

24 **Function** zavrsi\_mjerenje\_vremena**()** **Which** **Returns** **Nothing** **Is** **External;**

25 **Function** izvijesti\_o\_obrnuto\_poredanim\_nizovima**(**

26 **Integer32** n**)** **Which** **Returns** **Nothing** **Is** **External;**

27 **Function**

28 izvijesti\_o\_poredanim\_nizovima**(Integer32** n**)** **Which** **Returns** **Nothing** **Is** **External;**

29 **Function** izvijesti\_o\_pokretanju\_QuickSorta**(**

30 **Integer32** n**)** **Which** **Returns** **Nothing** **Is** **External;**

31 **Function** izvijesti\_o\_pokretanju\_MergeSorta**(**

32 **Integer32** n**)** **Which** **Returns** **Nothing** **Is** **External;**

33 **Function** izvijesti\_o\_pokretanju\_SelectSorta**(**

34 **Integer32** n**)** **Which** **Returns** **Nothing** **Is** **External;**

35 **Function**

36 izvijesti\_JavaScript\_o\_nedostatku\_memorije**()** **Which** **Returns** **Nothing** **Is** **External;**

37

38 **Integer32** DEBUG **:** **=** **0,** broj\_mjerenja **:** **=** **0;** **// broj\_mjerenja bit će koristan za**

39 **// pokretanje raznih algoritama**

40 **// ovisno o rednom broju mjerenja,**

41 **//što dobro dođe za**

42 **// eksperimentiranje s algoritmima.**

43

44 **// Napravimo sada omotnicu oko WebAssemblerske naredbe "memory.grow"...**

45 **Function** zauzmi\_memorijske\_stranice**(Integer32** broj\_stranica**)** **Which** **Returns**

46 **CharacterPointer**

47 **Does** **{** **// Vitičasta zagrada ovdje je opcionalna, compiler ju ignorira. No,**

48 **// dobro dođe kada pišemo AEC-ovski program u nekom IDE-u koji je**

49 **// primarno namijenjen za C-olike jezike, tada on može razumijeti da**

50 **// je ono između vitičastih zagrada blok naredbi.**

51 **Integer32** nova\_adresa\_u\_stranicama

52 **:** **=** **asm\_i32** **//"asm\_i32" kaže compileru da umetne asemblerski kod, i da**

53 **// pretpostavi da će se nakon njega na sistemskom stogu**

54 **// nalaziti vrijednost tipa "i32". To očito nije točno ako**

55 **// netko prebaci JavaScript virtualnu mašinu u 64-bitni**

56 **// način rada, ali nadam se da to nitko neće napraviti.**

57 **// Vjerojatnost da će JavaScript virtualnoj mašini trebati**

58 **// više nego 4GB RAM-a je zanemariva, a vjerojatnost da će**

59 **// se neki korisni programi srušiti zbog prebacivanja u**

60 **// 64-bitni način rada nije baš zanemariva.**

61 **(**R"(

62 (memory.grow

63 (i32.load %broj\_stranica) ;; Compiler će zamijeniti `%broj\_stranica` s

64 ;; asemblerskim kodom koji predstavlja pokazivač

65 ;; na varijablu `broj\_stranica`.

66 )

67 )"**);**

68 **If** nova\_adresa\_u\_stranicama **=** **-1** **Then** **{** **// Ako nema više**

69 **// slobodne memorije...**

70 **Return(-1);** **// Ne stavim li zagrade oko "-1", ClangFormat će misliti da je**

71 **// "Return" ime neke varijable.**

72 **}**

73 **EndIf;** **// Ni točka-zarez ovdje nije potreban, ali pomaže IDE-ovima da razumiju**

74 **// programski kôd.**

75 **Return** nova\_adresa\_u\_stranicama **\*** **64** **\*** **1024;** **// Na JavaScript Virtualnoj**

76 **// Mašini, jedna stranica**

77 **//(page) iznosi 64 KB.**

78 **}**

79 **EndFunction;**

80

81 **Integer32** velicina\_niza**;**

82 **Integer32Pointer** originalni\_niz**,** pomocni\_niz**;** **// To su globalne varijable,**

83 **// po defaultu su u nuli,**

84 **// dakle "originalni\_niz" i**

85 **//"pomocni\_niz" su na početku**

86 **// programa nulti pokazivači.**

87

88 **Integer32** broj\_obrnuto\_poredanih\_podniza**,** broj\_vec\_poredanih\_podniza**,**

89 broj\_pokretanja\_QuickSorta**,** broj\_pokretanja\_MergeSorta**,**

90 broj\_pokretanja\_SelectSorta**;**

91

92 **// Sad ćemo implementirati neke matematičke funkcije koje će nam trebati.**

93 **// Ne možemo pozvati JavaScriptine matematičke funkcije, jer one su metode**

94 **// singletona "Math", a ne postoji standardizirani način da se zovu**

95 **// metode JavaScriptinih objekata iz WebAssemblyja.**

96 **Decimal32** PRECISION **:** **=** **128;** **// Ovdje možemo balansirati između brzine i**

97 **// preciznosti. Ako smo previše precizni, bit**

98 **//ćemo spori. Ako smo previše neprecizni, lako**

99 **// se može dogoditi da precijenimo koliko se**

100 **// duboko rekurzija smije granati i izazovemo**

101 **// stack overflow.**

102

103 **Decimal32**

104 memoizacija\_prirodnog\_logaritma**[8192];** **// Memoizacija je dio memorije u koju**

105 **// se spremaju već izračunati podaci,**

106 **// da se ne računaju više puta.**

107

108 **Function** Eulerov\_broj\_na\_potenciju**(Decimal32** x**)**

109 **Which** **Returns** **Decimal32** **Is** **Declared;** **// Vidi dolje za pojašnjenje ovoga.**

110

111 **Function** prirodni\_logaritam**(Decimal32** x**)** **Which** **Returns** **Decimal32** **Does** **{**

112 **If** x **<** **1** **Then** **{** **Return(-**prirodni\_logaritam**(1** **/** x**));** **}**

113 **EndIf;**

114 **If(**x **<** **8192** **and**

115 **not(**memoizacija\_prirodnog\_logaritma**[asm\_f32(** **// Vidi iduću funkciju za**

116 **// pojašnjenje ovoga...**

117 "(f32.nearest (f32.load %x))"**)]** **=** **0))** **Then** **{**

118 **Return**

119 memoizacija\_prirodnog\_logaritma**[asm\_f32(**"(f32.nearest (f32.load %x))"**)];**

120 **}**

121 **EndIf;**

122 **If** x **>** **8192** **Then** **{**

123 **Return** **1** **+** prirodni\_logaritam**(**x **/** Eulerov\_broj\_na\_potenciju**(1));**

124 **}**

125 **EndIf;**

126 **// Prirodni logaritam je integral od 1/x u intervalu od 1 do x,**

127 **// srednjoškolska matematika.**

128 **Decimal32** zbroj **:** **=** **0,** epsilon **:** **=** **(**x **-** **1)** **/** **(5** **\*** PRECISION**),** i **:** **=** **1;**

129 **While(**epsilon **>** **0** **and** i **<** x**)** **or** **(**epsilon **<** **0** **and** i **>** x**)** **Loop** **{**

130 zbroj **+=** epsilon **/** i**;**

131 i **+=** epsilon**;** **// Preuzeo sam naredbe "+=", "-=", "\*=" i "/=" iz C-olikih**

132 **// jezika, smatram da znatno skraćuju neke kodove, a da ih ne**

133 **// čine nečitkima.**

134 **}**

135 **EndWhile;**

136 **If** x **<** **8192** **Then** **{**

137 **Return**

138 memoizacija\_prirodnog\_logaritma**[asm\_f32(**"(f32.nearest (f32.load %x))"**)]**

139 **:** **=** zbroj**;**

140 **}**

141 **EndIf;**

142 **Return** zbroj**;**

143 **}**

144 **EndFunction;**

145

146 **Decimal32**

147 memoizacija\_Eulerovog\_algoritma**[512];** **// Možda malo ubrza program, gotovo**

148 **// sigurno ga neće usporiti...**

149

150 **Function** abs**(Decimal32** x**)** **Which** **Returns** **Decimal32** **Is**

151 **Declared;** **// Deklarirat ćemo funkciju "abs" da se može koristiti prije no**

152 **// što se definira.**

153

154 **Function** Eulerov\_broj\_na\_potenciju**(Decimal32** x**)** **Which** **Returns** **Decimal32** **Does** **{**

155 **If(**abs**(**x**)** **<** **64** **and**

156 **not(**memoizacija\_Eulerovog\_algoritma

157 **[asm\_f32(**"(f32.nearest (f32.mul (f32.const 4) (f32.load %x)))"**)** **+**

158 **256]** **=** **0))**

159 **Then** **{** **// Koristio sam ovdje umetnuti asembler zato što ništa drugo što**

160 **// napišem u AEC-u neće učiniti da compiler koji sam napravio**

161 **// pozove asemblersku instrukciju "f32.nearest" (poznatu kao**

162 **// "round" u drugim programskim jezicima), a mnogo je lakše**

163 **// napisati malo umetnutog asemblera nego mijenjati i recompilirati**

164 **// compiler.**

165 **Return** memoizacija\_Eulerovog\_algoritma

166 **[asm\_f32(**"(f32.nearest (f32.mul (f32.const 4) (f32.load %x)))"**)** **+** **256];**

167 **}**

168 **EndIf;**

169 **// Eulerov Algoritam iz Matematike 2...**

170 **Decimal32** i **:** **=** **0,** y **:** **=** **1,** epsilon **:** **=** x **/** PRECISION**;**

171 **While(**epsilon **>** **0** **and** i **<** x**)** **or** **(**epsilon **<** **0** **and** i **>** x**)** **Loop** **{**

172 y **+=** epsilon **\*** y**;**

173 i **+=** epsilon**;**

174 **}**

175 **EndWhile;**

176 **If** abs**(**x**)** **<** **64** **Then** **{**

177 **Return** memoizacija\_Eulerovog\_algoritma

178 **[asm\_f32(**"(f32.nearest (f32.mul (f32.const 4) (f32.load %x)))"**)** **+** **256]**

179 **:** **=** y**;**

180 **}**

181 **EndIf;**

182 **Return** y**;**

183 **}**

184 **EndFunction;**

185

186 **Function** abs**(Decimal32** x**)** **Which** **Returns** **Decimal32** **Does** **{**

187 **// U svoj sam programski jezik ugradio uvijetni "?:" operator kakav**

188 **// postoji u C-u, C++-u i JavaScriptu. Izgleda malo ružno, ali nekad zna**

189 **// znatno skratiti programske kodove. Odlučio sam implementirati desno**

190 **// asocijativan uvijetni operator, kakav je u C-u, C++-u i JavaScriptu,**

191 **// a ne lijevo asocijativan kakav je u PHP-u i srodnim jezicima.**

192 **// Jednostavno mi ima više smisla da uvijetni operator bude asocijativan**

193 **// na desno nego na lijevo.**

194 **Return(**x **<** **0)** **?** **// Ako je x manji od 0...**

195 **-**x **//...vrati (proglasi rezultatom) -x...**

196 **:** x**;** **// inače, proglasi x rezultatom.**

197 **}**

198 **EndFunction;**

199

200 **Function** ostatak\_pri\_dijeljenju**(Decimal32** x**,**

201 **Decimal32** y**)** **Which** **Returns** **Decimal32** **Does** **{**

202 **If** DEBUG **=** **1** **Then** **{**

203 printString**(**"Zatrazen je ostatak pri dijeljenju od brojeva: "**);**

204 **// Neću upotrebljavati hrvatske znakove u stringovima, jer ću**

205 **// naletjeti na probleme pri pretvorbi u JavaScriptin string.**

206 printFloat**(**x**);**

207 printFloat**(**y**);**

208 printString**(**"Sada ce se program mozda srusiti..."**);**

209 **}**

210 **EndIf;**

211 **If** abs**(**x **/** y**)** **>** Eulerov\_broj\_na\_potenciju**(**prirodni\_logaritam**(2)** **\*** **63)** **Then** **{**

212 **Return** **0;** **// Imate bolju ideju što da se radi u slučaju da količnik**

213 **// ne stane niti u Integer64 (C-ovski "long long")?**

214 **}**

215 **EndIf;**

216 **Return** x **-** y **\*Integer64(**x **/** y**);** **// Ako napišem "Integer32",**

217 **// riskiram da će JavaScript**

218 **// virtualna mašina prekinuti**

219 **// izvođenje programa jer je**

220 **// broj "x/y" izvan intervala**

221 **// koji 32-bitni cijeli brojevi**

222 **// mogu prikazati (od oko dvije**

223 **// milijarde u pozitivno i**

224 **// negativno).**

225 **}**

226 **EndFunction;**

227

228 **Function** pow**(Decimal32** x**,** **Decimal32** y**)** **Which** **Returns** **Decimal32** **Does** **{**

229 **Decimal32** result

230 **:** **=** Eulerov\_broj\_na\_potenciju**(**prirodni\_logaritam**(**abs**(**x**))** **\*** y**);**

231 **Return** x **=**

232 **0** **?** **0** **:** ostatak\_pri\_dijeljenju**(**x**,** **2)** **=** **1** **and** x **<** **0** **?** **-**result **:** result**;**

233 **}**

234 **EndFunction;**

235

236 **// I sada krećemo pisati taj hibridni algoritam razvrstavanja...**

237 **Function** hybrid\_sort**(Integer32** donja\_granica**,** **Integer32** gornja\_granica**,**

238 **Integer32** dubina\_rekurzije**)** **Which** **Returns** **Nothing** **Does** **{**

239 **If** gornja\_granica **-** donja\_granica **<** **2** **Then** **{** **// Ako je niz duljine manje od**

240 **// 2 (0 ili 1), znači da je već**

241 **// poredan, pa prekidamo**

242 **// izvođenje ovog potprograma.**

243 **Return;**

244 **}**

245 **ElseIf** gornja\_granica **-** donja\_granica **=** **2** **Then** **{** **// Najčesći slučaj,**

246 **// vrijedi ga posebno**

247 **// obraditi jer time**

248 **// možemo znatno ubrzati**

249 **// program.**

250 **If** **ValueAt(**originalni\_niz **+** donja\_granica**)** **>**

251 **ValueAt(**originalni\_niz **+** donja\_granica **+** **1)** **Then** **{**

252 **Integer32** pomocna\_varijabla\_za\_zamijenu

253 **:** **=** **ValueAt(**originalni\_niz **+** donja\_granica**);**

254 **ValueAt(**originalni\_niz **+** donja\_granica**)**

255 **:** **=** **ValueAt(**originalni\_niz **+** donja\_granica **+** **1);**

256 **ValueAt(**originalni\_niz **+** donja\_granica **+** **1)**

257 **:** **=** pomocna\_varijabla\_za\_zamijenu**;**

258 **}**

259 **EndIf;**

260 **Return;**

261 **}**

262 **ElseIf** gornja\_granica **-**

263 donja\_granica**<8** **or** **asm\_i32(**"(global.get $stack\_pointer)"**)>** **4** **\*** **1024** **-**

264 **73** **Then** **{**

265 **// Za male je nizove SelectionSort brži i od MergeSorta i QuickSorta. Također,**

266 **// kako nije rekurzivan, može se koristiti i kad posve potrošimo memoriju na**

267 **// sistemskom stogu(na JavaScript Virtualnoj Mašini to jest ne više nego 4 KB,**

268 **// kako bijaše u doba Netscapea 2, godine 1996, tako i danas).**

269 broj\_pokretanja\_SelectSorta**:**

270 **=** broj\_pokretanja\_SelectSorta **+** **1;**

271 **Integer32** i **:** **=** donja\_granica**;**

272 **While** i **<** gornja\_granica **Loop** **{**

273 **Integer32** gdje\_je\_minimum **:** **=** i**;**

274 **Integer32** j **:** **=** i **+** **1;**

275 **While** j **<** gornja\_granica **Loop** **{**

276 **If** **ValueAt(**originalni\_niz **+** gdje\_je\_minimum**)** **>**

277 **ValueAt(**originalni\_niz **+** j**)** **Then** **{**

278 gdje\_je\_minimum

279 **/\***

280 **\* ClangFormat (koji koristim za formatiranje AEC programa, a**

281 **\* primarno je namijenjen za C-olike jezike) pogrešno tumači**

282 **\* AEC-ov operator pridruživanja ":=" kao C-ovu oznaku za labele**

283 **\* ':' plus C-ov operator pridruživanja '='. Ne vidim nekakvo**

284 **\* jednostavno rješenje tog problema. Na sreću, AEC-ov tokenizer,**

285 **\* još od najranije verzije, trpi ako se stavi whitespace znak**

286 **\* (razmak, tabulator ili znak za novi red) između ':' i '=' u**

287 **\* operatoru ":=", ali svejedno to ne izgleda dobro.**

288 **\*/**

289 **:**

290 **=** j**;**

291 **}**

292 **EndIf;**

293 j **+=** **1;**

294 **}**

295 **EndWhile;**

296 **Integer32** pomocna\_varijabla\_za\_zamijenu **:** **=** **ValueAt(**originalni\_niz **+** i**);**

297 **ValueAt(**originalni\_niz **+** i**)** **:** **=** **ValueAt(**originalni\_niz **+** gdje\_je\_minimum**);**

298 **ValueAt(**originalni\_niz **+** gdje\_je\_minimum**)**

299 **:** **=** pomocna\_varijabla\_za\_zamijenu**;**

300 i **+=** **1;**

301 **}**

302 **EndWhile;**

303 **Return;**

304 **}**

305 **EndIf;**

306 **Decimal32** razvrstanost **:** **=** **0;**

307 **Integer32** i **:** **=** donja\_granica**,** je\_li\_niz\_vec\_poredan **:** **=** **1;**

308 **While** i **<** gornja\_granica **-** **1** **Loop** **{**

309 razvrstanost **+=**

310 **(ValueAt(**originalni\_niz **+** i**)** **<** **ValueAt(**originalni\_niz **+** i **+** **1)** **or**

311 **ValueAt(**originalni\_niz **+** i**)** **=** **ValueAt(**originalni\_niz **+** i **+** **1));**

312 **If** **ValueAt(**originalni\_niz **+** i**)** **>** **ValueAt(**originalni\_niz **+** i **+** **1)** **Then** **{**

313 je\_li\_niz\_vec\_poredan**:**

314 **=** **0;**

315 **}**

316 **EndIf;**

317 i **+=** **1;**

318 **}**

319 **EndWhile;**

320 razvrstanost**:**

321 **=** razvrstanost **/** **((**gornja\_granica **-** donja\_granica **-** **1)** **/** **2.)** **-** **1;**

322 **// Provjeri je li sve u redu, i, ako nije, obavijesti.**

323 **If** abs**(**razvrstanost**)** **>** **1** **Then** **{**

324 **// To ne smije biti...**

325 printString**(**"Apsolutna vrijednost razvrstanosti je veca od 1!"**);**

326 printString**(**"Relevantni dio niza iznosi:"**);** **// Da se ne moram baktati s**

327 **// debuggerima za JavaScript**

328 **// virtualnu mašinu ako dođe**

329 **// do problema, lakše mi**

330 **// je ispisati brojeve u**

331 **// programu nego tražiti**

332 **// kako narediti**

333 **// debuggeru da ih**

334 **// ispiše.**

335 i**:**

336 **=** donja\_granica**;**

337 **While** i **<** gornja\_granica **Loop** **{**

338 printInteger**(ValueAt(**originalni\_niz **+** i**));**

339 i **+=** **1;**

340 **}**

341 **EndWhile;**

342 printString**(**"Kraj relevantnog dijela niza!"**);**

343 **}**

344 **EndIf;**

345 **If** je\_li\_niz\_vec\_poredan **and** **not(**razvrstanost **=** **1** **or** razvrstanost **>** **1)** **Then** **{**

346 **// Opet ne smije biti...**

347 printString**(**"Niz je poredan, a razvrstanost nije 1."**);**

348 printString**(**"Relevantni dio niza iznosi:"**);**

349 i**:**

350 **=** donja\_granica**;**

351 **While** i **<** gornja\_granica **Loop** **{**

352 printInteger**(ValueAt(**originalni\_niz **+** i**));**

353 i **+=** **1;**

354 **}**

355 **EndWhile;**

356 printString**(**"Kraj relevantnog dijela niza!"**);**

357 **}**

358 **EndIf;**

359 **If** **not(**je\_li\_niz\_vec\_poredan**)** **and** **(**razvrstanost **=** **1** **or** razvrstanost **>** **1)**

360 **Then** **{**

361 **// Open ne smije biti...**

362 printString**(**"Razvrstanost je 1, a niz nije poredan!"**);**

363 printString**(**"Relevantni dio niza iznosi:"**);**

364 i**:**

365 **=** donja\_granica**;**

366 **While** i **<** gornja\_granica **Loop** **{**

367 printInteger**(ValueAt(**originalni\_niz **+** i**));**

368 i **+=** **1;**

369 **}**

370 **EndWhile;**

371 printString**(**"Kraj relevantnog dijela niza!"**);**

372 **}**

373 **EndIf;**

374 **// Idemo dalje...**

375 **Decimal32** razvrstanost\_na\_potenciju**[8]** **:** **=** **{1};** **// Formula će se brže**

376 **// izračunati ako ne**

377 **// pozivamo "pow" gdje**

378 **// ne treba (kad je**

379 **// eksponent prirodan**

380 **// broj).**

381i**:**

382 **=** **1;**

383 **While** i **<** **8** **Loop** **{**

384 razvrstanost\_na\_potenciju**[**i**]**

385 **:** **=** razvrstanost\_na\_potenciju**[**i **-** **1]** **\*** razvrstanost**;**

386 i **+=** **1;**

387 **}**

388 **EndWhile;**

389 **// Formula koju je ispisao genetski algoritam za predviđanje koliko će**

390 **// usporedbi QuickSort napraviti:**

391 **// https://github.com/FlatAssembler/ArithmeticExpressionCompiler/tree/master/QuickSort/Genetic\_algorithm\_for\_deriving\_the\_formula**

392 **Decimal32** polinom\_pod\_apsolutnom

393 **:** **=** **2.38854** **\*** razvrstanost\_na\_potenciju**[7]** **-**

394 **0.284258** **\*** razvrstanost\_na\_potenciju**[6]** **-**

395 **1.87104** **\*** razvrstanost\_na\_potenciju**[5]** **+**

396 **0.372637** **\*** razvrstanost\_na\_potenciju**[4]** **+**

397 **0.167242** **\*** razvrstanost\_na\_potenciju**[3]** **-**

398 **0.0884977** **\*** razvrstanost\_na\_potenciju**[2]** **+** **0.315119** **\*** razvrstanost**;**

399 **Decimal32** Eulerov\_broj\_na\_koju\_potenciju

400 **:** **=** **(**prirodni\_logaritam**(**gornja\_granica **-** donja\_granica**)** **+**

401 prirodni\_logaritam**(**

402 prirodni\_logaritam**(**gornja\_granica **-** donja\_granica**)))** **\***

403 **1.05** **+**

404 **(**prirodni\_logaritam**(**gornja\_granica **-** donja\_granica**)** **-**

405 prirodni\_logaritam**(**

406 prirodni\_logaritam**(**gornja\_granica **-** donja\_granica**))** **-**

407 prirodni\_logaritam**(2))** **\***

408 **0.9163** **\*** abs**(**polinom\_pod\_apsolutnom**);**

409 **Decimal32** koliko\_usporedbi\_ocekujemo\_od\_QuickSorta

410 **:** **=** Eulerov\_broj\_na\_potenciju**(**Eulerov\_broj\_na\_koju\_potenciju**);**

411 **Decimal32** koliko\_usporedbi\_ocekujemo\_od\_MergeSorta

412 **:** **=** **(mod(**broj\_mjerenja**,** **2)** **+** **1)** **\***

413 **(**gornja\_granica **-**

414 donja\_granica**)** **\*** **// Nisam siguran treba li ovdje**

415 **// pisati "2 \* (gornja\_granica...".**

416 **// S jedne strane, MergeSort radi**

417 **// dvije petlje, jedna za spajanje**

418 **// dijelova originalnog niza u**

419 **// pomoćni niz, a druga za kopiranje**

420 **// pomoćnog niza u originalni. S**

421 **// druge strane, iz mjerenja se čini**

422 **// da je cjelokupni algoritam brži**

423 **// ako se ne množi s 2. Zato ćemo**

424 **// nekada množiti s dva, a nekada ne,**

425 **// pa ćemo preciznijim mjerenjima**

426 **// vidjeti što je bolje.**

427 prirodni\_logaritam**(**gornja\_granica **-** donja\_granica**)** **/**

428 prirodni\_logaritam**(2);**

429 **// I sada kreće grananje na temelju izračunatog...**

430 **If** razvrstanost **=** **1** **or** razvrstanost **>** **1** **Then** **{**

431 broj\_vec\_poredanih\_podniza **+=** **1;**

432 **Return;**

433 **}**

434 **ElseIf** razvrstanost **=** **-1** **or** razvrstanost **<** **-1** **Then** **{**

435 broj\_obrnuto\_poredanih\_podniza **+=** **1;**

436 **Integer32** i **:** **=** donja\_granica**;**

437 **Integer32** j **:** **=** gornja\_granica **-** **1;**

438 **While** i **<** gornja\_granica **Loop** **{**

439 **ValueAt(**pomocni\_niz **+** i**)** **:** **=** **ValueAt(**originalni\_niz **+** j**);**

440 j **-=** **1;**

441 i **+=** **1;**

442 **}**

443 **EndWhile;**

444 i**:**

445 **=** donja\_granica**;**

446 **While** i **<** gornja\_granica **Loop** **{**

447 **ValueAt(**originalni\_niz **+** i**)** **:** **=** **ValueAt(**pomocni\_niz **+** i**);**

448 i **+=** **1;**

449 **}**

450 **EndWhile;**

451 **Return;**

452 **}**

453 **ElseIf(**koliko\_usporedbi\_ocekujemo\_od\_MergeSorta **<**

454 koliko\_usporedbi\_ocekujemo\_od\_QuickSorta **or**

455 dubina\_rekurzije **>** pow**(2,** **18** **-** prirodni\_logaritam**(**velicina\_niza**)** **/**

456 prirodni\_logaritam**(2))**

457 **// JavaScriptina virtualna mašina ima**

458 **// 4KB memorije na sistemskom stogu,**

459 **// i alociranje više heap memorije**

460 **// ne mijenja tu nesretnu činjenicu.**

461 **// Ne znam kako Emscripten (modificirana**

462 **// verzija CLANG-a koja compilira**

463 **// C++ u WebAssembly) to rješava.**

464 **)** **and**

465 **not(**gornja\_granica **-** donja\_granica **=**

466 velicina\_niza **and** **not(mod(**broj\_mjerenja**,** **3)))**

467 **// Izgleda da je, iz nekog razloga, program brži ako se QuickSort**

468 **// pokrene barem jednom, no probajmo raditi preciznija mjerenja.**

469 **Then** **{**

470 **// MergeSort algoritam (približno poredani podnizovi,**

471 **// za koje je MergeSort efikasniji od QuickSorta,**

472 **// a moj ga program također koristi kada ima još**

473 **// malo mjesta na sistemskom stogu, pa QuickSort**

474 **// nije opcija)...**

475 broj\_pokretanja\_MergeSorta **+=** **1;**

476 **Integer32** sredina\_niza **:** **=** **(**gornja\_granica **+** donja\_granica**)** **/** **2;**

477 **// Prvo, rastavi niz na koji pokazuje pokazivač "originalni\_niz"**

478 **// na niz od originalni\_niz+donja\_granica do**

479 **// originalni\_niz+sredina\_niza i niz od**

480 **// originalni\_niz+sredina\_niza do**

481 **// originalni\_niz+gornja\_granica,**

482 **// i poredaj ta dva niza.**

483 hybrid\_sort**(**donja\_granica**,** sredina\_niza**,** dubina\_rekurzije **+** **1);**

484 hybrid\_sort**(**sredina\_niza**,** gornja\_granica**,** dubina\_rekurzije **+** **1);**

485 **// Spajanje nizova originalni\_niz[donja\_granica..sredina\_niza]**

486 **// i originalni\_niz[sredina\_niza..gornja\_granica] u jedan niz...**

487 **Integer32** i **:** **=** donja\_granica**;**

488 **Integer32** gdje\_smo\_u\_prvom\_nizu **:** **=** donja\_granica**;**

489 **Integer32** gdje\_smo\_u\_drugom\_nizu **:** **=** sredina\_niza**;**

490 **While** i **<** gornja\_granica **Loop** **{**

491 **If(**gdje\_smo\_u\_prvom\_nizu **=**

492 sredina\_niza **or**

493 **ValueAt(**originalni\_niz **+** gdje\_smo\_u\_drugom\_nizu**)** **<**

494 **ValueAt(**originalni\_niz **+** gdje\_smo\_u\_prvom\_nizu**))** **and**

495 gdje\_smo\_u\_drugom\_nizu **<** gornja\_granica **Then** **{**

496 **ValueAt(**pomocni\_niz **+** i**)**

497 **:** **=** **ValueAt(**originalni\_niz **+** gdje\_smo\_u\_drugom\_nizu**);**

498 gdje\_smo\_u\_drugom\_nizu **+=** **1;**

499 **}**

500 **Else** **{**

501 **ValueAt(**pomocni\_niz **+** i**)**

502 **:** **=** **ValueAt(**originalni\_niz **+** gdje\_smo\_u\_prvom\_nizu**);**

503 gdje\_smo\_u\_prvom\_nizu **+=** **1;**

504 **}**

505 **EndIf;**

506 i **+=** **1;**

507 **}**

508 **EndWhile;**

509 i**:**

510 **=** donja\_granica**;**

511 **While** i **<** gornja\_granica **Loop** **{**

512 **ValueAt(**originalni\_niz **+** i**)** **:** **=** **ValueAt(**pomocni\_niz **+** i**);**

513 i **+=** **1;**

514 **}**

515 **EndWhile;**

516 **Return;**

517 **}**

518 **Else** **{** **// QuickSort algoritam (nasumično ispremještani podnizovi)...**

519 broj\_pokretanja\_QuickSorta **+=** **1;**

520 **// Daljnji kod je približno prepisan s**

521 **// https://www.geeksforgeeks.org/quick-sort/**

522 **// Iskreno, ne razumijem ni ja točno kako funkcionira.**

523 **// On navodno preuređuje niz tako da svi elementi koji su manji**

524 **// od onog koji je bio prvi (pivot) dođu prije njega, a ostali**

525 **// poslije njega.**

526 **Integer32** pivot **:** **=** **ValueAt(**originalni\_niz **+** gornja\_granica **-** **1);**

527 **Integer32** i **:** **=** donja\_granica **-** **1;**

528 **Integer32** j **:** **=** donja\_granica**;**

529 **Integer32** pomocna\_varijabla\_za\_zamijenu**;**

530 **While** j **<** gornja\_granica **-** **1** **Loop** **{**

531 **If** **ValueAt(**originalni\_niz **+** j**)** **<** pivot **Then** **{**

532 i **+=** **1;**

533 pomocna\_varijabla\_za\_zamijenu**:**

534 **=** **ValueAt(**originalni\_niz **+** i**);**

535 **ValueAt(**originalni\_niz **+** i**)** **:** **=** **ValueAt(**originalni\_niz **+** j**);**

536 **ValueAt(**originalni\_niz **+** j**)** **:** **=** pomocna\_varijabla\_za\_zamijenu**;**

537 **}**

538 **EndIf;**

539 j **+=** **1;**

540 **}**

541 **EndWhile;**

542 pomocna\_varijabla\_za\_zamijenu**:**

543 **=** **ValueAt(**originalni\_niz **+** i **+** **1);**

544 **ValueAt(**originalni\_niz **+** i **+** **1)**

545 **:** **=** **ValueAt(**originalni\_niz **+** gornja\_granica **-** **1);**

546 **ValueAt(**originalni\_niz **+** gornja\_granica **-** **1)**

547 **:** **=** pomocna\_varijabla\_za\_zamijenu**;**

548 **Integer32** gdje\_je\_pivot **:** **=** i **+** **1;**

549 hybrid\_sort**(**donja\_granica**,** gdje\_je\_pivot**,** dubina\_rekurzije **+** **1);**

550 hybrid\_sort**(**gdje\_je\_pivot**,** gornja\_granica**,** dubina\_rekurzije **+** **1);**

551 **Return;**

552 **}**

553 **EndIf;**

554 **// Ovdje tok programa ne smije doći.**

555 printString**(**"Izgleda da compiler nije ispravno "

556 "preveo kontrolne strukture!"**);**

557 **}**

558 **EndFunction;**

559

560 **// Ovo je funkcija koju će pozvati JavaScript...**

561 **Function** pocetna\_AEC\_funkcija**()** **Which** **Returns** **Nothing** **Does** **{**

562 **If** originalni\_niz **=** **-1** **or** pomocni\_niz **=** **-1** **Then** **{**

563 **Return;** **// Ako JavaScript nastavlja pokretati ovaj program**

564 **// unatoč nedostatku memorije, neka onda on ne radi ništa.**

565 **}**

566 **EndIf;**

567 **// Testiraj matematičke funkcije...**

568 **If** abs**(**pow**(3,** **3)** **-** **27)** **>** **2** **Then** **{** **// Da, one su jako neprecizne, ali zato**

569 **// jako brze.**

570 printString**(**"Izgleda da matematicke funkcije ne funkcioniraju dobro."**);**

571 printString**(**"pow(3, 3) ="**);**

572 printFloat**(**pow**(3,** **3));**

573 **}**

574 **EndIf;**

575 **// Doznaj veličinu niza iz JavaScripta...**

576 **Integer32** prijasnja\_velicina\_niza **:** **=** velicina\_niza**;**

577 velicina\_niza**:**

578 **=** daj\_velicinu\_niza**();**

579 **// Ako je potrebno, zauzmi još memorije...**

580 **If** velicina\_niza **/** **(64** **\*** **1024** **/** **4)** **+**

581 **not(not(mod(**velicina\_niza**,** **64** **\*** **1024** **/** **4)))** **>**

582 prijasnja\_velicina\_niza **/** **(64** **\*** **1024** **/** **4)** **+**

583 **not(not(mod(**prijasnja\_velicina\_niza**,** **64** **\*** **1024** **/** **4)))** **or**

584 prijasnja\_velicina\_niza **=** **0** **Then** **{**

585 originalni\_niz**:**

586 **=** zauzmi\_memorijske\_stranice**(4** **\*** velicina\_niza **/** **(64** **\*** **1024)** **+**

587 **not(not(mod(**velicina\_niza**,** **64** **\*** **1024** **/** **4))));**

588 pomocni\_niz**:**

589 **=** zauzmi\_memorijske\_stranice**(4** **\*** velicina\_niza **/** **(64** **\*** **1024)** **+**

590 **not(not(mod(**velicina\_niza**,** **64** **\*** **1024** **/** **4))));**

591 **If** originalni\_niz **=** **-1** **or** pomocni\_niz **=** **-1** **Then** **{**

592 printString**(**"Nema dovoljno memorije za nastavak programa!?"**);**

593 izvijesti\_JavaScript\_o\_nedostatku\_memorije**();**

594 **Return;** **// Prekini izvršavanje ovog programa.**

595 **}**

596 **EndIf;**

597 **}**

598 **EndIf;**

599 **// Sada zatraži od JavaScripta da kopira niz koji treba poredati**

600 **// na memorijski prostor koji si (prethodno ili sada) zauzeo.**

601 kopiraj\_niz\_na\_adresu**(**originalni\_niz**);**

602 **// I sada ga kreni razvrstavati i mjeriti koliko ti treba vremena.**

603broj\_obrnuto\_poredanih\_podniza**:**

604 **=** broj\_vec\_poredanih\_podniza **:** **=** broj\_pokretanja\_QuickSorta

605 **:** **=** broj\_pokretanja\_MergeSorta **:** **=** broj\_pokretanja\_SelectSorta

606 **:** **=** **0;** **// Nisam mogao odoljeti da u svoj programski jezik ne dodam**

607 **// ulančano pridruživanje iz C-a, C++-a i JavaScripta (da možemo**

608 **// više varijabli postaviti na neku vrijednost u jednoj naredbi).**

609 broj\_mjerenja **+=** **1;**

610 pocni\_mjerenje\_vremena**();**

611 hybrid\_sort**(0,** velicina\_niza**,** **0);**

612 zavrsi\_mjerenje\_vremena**();**

613 **// Kad završi mjerenje vremena (koje se vrtilo u JavaScriptu),**

614 **// obavijesti JavaScript o onome što si ti izmjerio.**

615 izvijesti\_o\_obrnuto\_poredanim\_nizovima**(**broj\_obrnuto\_poredanih\_podniza**);**

616 izvijesti\_o\_poredanim\_nizovima**(**broj\_vec\_poredanih\_podniza**);**

617 izvijesti\_o\_pokretanju\_QuickSorta**(**broj\_pokretanja\_QuickSorta**);**

618 izvijesti\_o\_pokretanju\_MergeSorta**(**broj\_pokretanja\_MergeSorta**);**

619 izvijesti\_o\_pokretanju\_SelectSorta**(**broj\_pokretanja\_SelectSorta**);**

620 **// Napravi neki osnovni sanity-check, je li niz uistinu poredan?**

621 **Integer32** i **:** **=** **0;**

622 **While** i **<** velicina\_niza **-** **1** **Loop** **{**

623 **If** **ValueAt(**originalni\_niz **+** i**)** **>** **ValueAt(**originalni\_niz **+** i **+** **1)** **Then** **{**

624 printString**(**"Niz nije poredan!"**);**

625 **Return;** **// Nemoj to ispisati više puta, nego prekini program čim**

626 **// si uočio prvu nepodudarnost.**

627 **}**

628 **EndIf;**

629 i **+=** **1;**

630 **}**

631 **EndWhile;**

632 **}**

633 **EndFunction;**

634