Amino_KasaRomana

Maciej Nasinski

6 października 2016

Poniższa analiza została przeprowadzona w celach edukacyjnych przy najwyższej staranności. Wykorzystując baze danych zawierającą wartości 1002 kuponów zbadano ich charakterystyki oraz szanse na wygraną. W procesie znajdowania par kuponów potzrebnych do wygrania wykożystano algorytm knapsack. Wszystkie obliczenia zostały wykonane w programie R.

Regulamin Promocji

Źródło bazy danych:

Warotści z 1000 zupek otwartych na kanale Youtube “Bez Kanału”

Wgranie wszystkich obserwacji.

```
kupony = as.numeric(readLines("./knapsack.txt"))

kupony
```

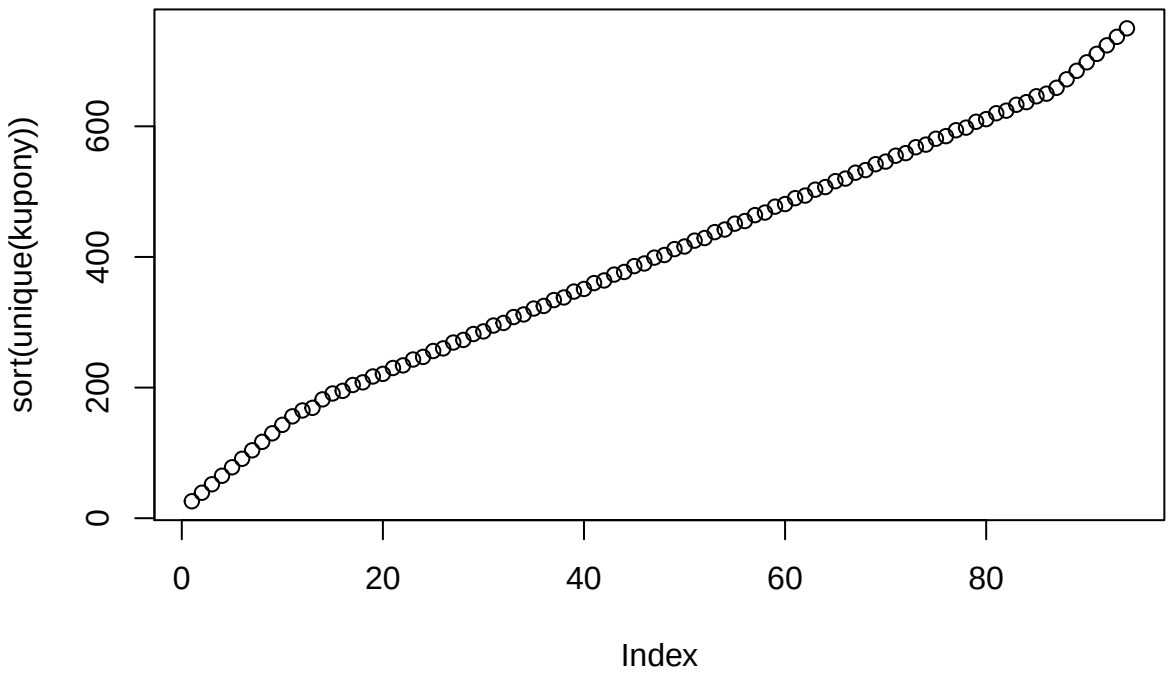
```
##    [1] 455 650  26 464  26 750 490 520 399 390 750 455 104 260  26 438 633
##   [18] 455 195 516 598 481 299 724 243  26 533 455 572 208 520 130 455  26
##   [35] 650 581 412 559 260 455 637 750 130 338 321  26 555  26 338 611  65
##   [52]  26 490 195 260 390 260 455 750 399 455 429 750 416 533 533 750 273
##   [69] 572 273 750 130 455 325 750 455 611 750 416 269 455 542  26 130  52
##   [86] 260 191 438 650 737 217 308 390 529 260 546 351  78 624 243  26 321
##  [103]  65 412 338 130  65 650 412 130 351 130 455 117 156 520 650  78 737
##  [120] 351 442 750 650 321 650 282 260 468 312 117 750 451  26 412 412 750
##  [137] 217 260 672 750 286 750 724 750 438 533 438 260 533 750  26 260  52
##  [154] 455 130 585 295  26 750 516  39 260 104 373  26 572  26 438  78 390
##  [171] 750 390 204 650 650 572 130 260  26 130 364 611 594 260 750 503  91
##  [188] 672 559 455 455 260 169 750 390 299 650 516 468 598 221  26 494 273
##  [205] 390 594  26 390 260 399 750 195 425  26  26 351 308 230  26 607 650
##  [222] 637 568 390 572  65  26 273  26 165 386 750 650 130 529 182 130 130
##  [239] 390 234 390 650 624 438 390 373  26  26 260 750 455 546 390  26 750
##  [256] 412  26  26 711 429 711 195 390 611 542 390 750 412  26 490 568 481
##  [273] 169 260  78 711 520 195 659  26 260 442 455 247 542  26 650 750 217
##  [290] 672 269 750 260  39 425  26  26  39 130 520 282 299 269 750 750 750
##  [307] 299 685 390  39 624 533 260 455  26 208 390 338  65 130 338 750 260
##  [324] 477 390  52 546 455 351 546 217 221 243 390 117 260 256 390  78 737
##  [341] 104 143 412 750 581 585 750 130 750 390 464 650 191  26 494 425 737
##  [358] 477 312  26 130 221  26 529 650 611 546 750 165 334 399 650 169 312
##  [375] 750 325 130  52  91  26 260 750  26 455 338 273 117 650 377 542 750
##  [392] 390 659  26 165 672  39 390  26 390 455 568 529 312 650 455 451 403
##  [409] 455  26 260  26  26 156 750 750 455 555 299 260 607 455 620 390 750
##  [426] 260 243 295 390 455 594 546 130 260 390 260 568 650 286 243 598 607
##  [443] 438 308 650 455 455 130 390 260 477 481 165 130 659 650  26  91 295
##  [460]  52 555 650 546 390 386 737  26 390 685 130  91 477  26 455  52 685
##  [477]  26 130 390 390 520 143  26 455 585 455 455 646 390 750 455 390 260
##  [494] 542 650  91  39 390 312  26 750 260 455 442 494 416 598 260 572 455
##  [511]  39  65 351 611 455 230  52 750  65 273 650 724 598 568  26  26  26
##  [528] 455 195 117 750 750 698 390 130 130  91 182 269 750 260  26 455 750
```

```
##  [545] 234 650 256 325 260  26 169 542  26 455 750 442 256 611 412 130 455
##  [562] 412 130 165 399 260 750 477  26 130 650 425 455 455 581 464 698 425
##  [579] 624  26 117 598 130 529 438 559 390 130 217 468 750 750 347 308 516
##  [596] 260 312  52 390 230 373 390 260 191 620  52 650 130 750 750 347 585
##  [613] 243 490 282 282 750 325  26 273 455 377 750 507 455 455 724 390 650
##  [630] 221 585 442 637 295 438 520 750 425 347 247  26 750 295 698 169 130
##  [647] 260 520 559 455 750  91 750 347 312 260 130 321 455  26  26 442 143
##  [664] 750 633  26 360 607 390 737 156  26 750 750 516 191  78 455 455 156
##  [681] 455 360 191 455 464 130 650 507 559 546  52  26 425 390 533 260 559
##  [698] 650 468 230 282 750  26 555  26 568 130 533 260 455 581 750  26 260
##  [715] 455 659 750 191 750 650 624 451  26  26 399 390 117 503 455  78 750
##  [732] 659 503 494 321 256 637  26 520 520 455 347 650 750 182 455 247 659
##  [749] 130 386 503 260  26 572 386 529 533 750 650 195 750 477 455 650 425
##  [766] 130 455 243 455  26 260 585 624 546 650 416 711 555 455 455 750 594
##  [783] 542 282 234 230 750 750 494  26 390 455 650 221 455 390 325 650 750
##  [800] 143 416  26 299 130 377  26 260 750 542 130 455 750 737 390 182 347
##  [817] 477  78 555 442 598 351  26 503 750  39 243 455 750  26  26 130 429
##  [834] 455 568 130 455  26 650 650 555 438 650 416 650 243 455 260 390 650
##  [851] 455 130 650 555 416 724 260 130 685 650 455 542 260 130 637 425 412
##  [868] 390 724  26 130 529 165  26 555 321 204 390 130 195 390  26  26 260
##  [885] 156 260 312 130 130 338  39 390 750 451 650 130 412 390 464 455 260
##  [902] 130 373 230 650 286 130 442 243  26 308 390 750  26 243  26 182 685
##  [919] 750  26 221 321 295 585 650 364 750 659 650 711 633  26  26 347 260
##  [936] 750 260 624 286 260 230 260 295 442  26 130 390 243 386  26 598 325
##  [953] 503 130 260 364 672 429 247 273  39 117 637 260 598 750 169 130 260
##  [970] 286 260 260 750 750 273 182 130 260 130 412 451 191 390 438 516 130
##  [987] 650 750 750  26 295 390  26  26 750 390 347 377 156  78 516 594
```

Wykres rosnących wartości unikalnych:

Z wszystkich wartości wyrzucono te powtarzające się a następnie posortowano do najmniejszej do najwiekszej.

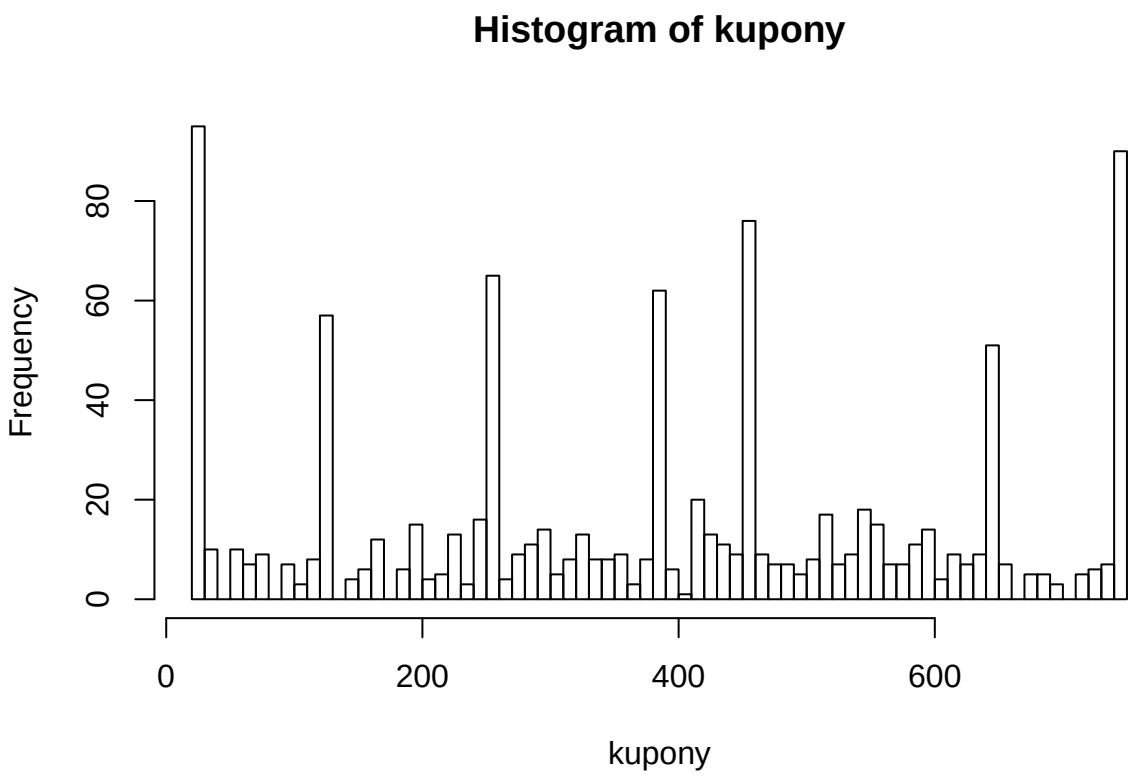
```
plot(sort(unique(kupony)))
```



Łatwo zaobserwować że występuje pewna zależność.

Histrogram:

```
hist(kupony,breaks=100)
```



Histogram ukazuje że pewne wartości powtarzają się dużo częściej niż inne.

Liczby wydają się być generowane z dwóch scieżek:

- jako kolejne iloczyny liczby 13 zaczynając od 26
- liczba 165 oraz sumowanie jej z kolejnymi iloczynami liczby 13

```
rep13 = sapply(1:52,function(x) 13*x)

rep2_13 = sapply(1:45,function(x) 165+(13*x))

sum(kupony %in% rep13 & kupony %in% rep2_13)
```

```
## [1] 0
```

Wszystkie kupony z naszej bazy danych znajdują się pośród wszystkich wygenerownych liczb z 2 podanych ścieżek.

Porównanie kuponów z bazy danych oraz wygnerowanych.

Do kuponów z bazy danych dodano kupon o kwocie 178.

```
a = sort(c(unique(kupony),178))
b = sort(c(sapply(0:45,function(x) 165+(13*x)),sapply(2:50,function(x) (13*x))))

both = data.frame(realne = a, wygenerowane = b)

both
```

```
##    realne wygenerowane
## 1      26           26
## 2      39           39
## 3      52           52
## 4      65           65
## 5      78           78
## 6      91           91
## 7     104          104
## 8     117          117
## 9     130          130
## 10    143          143
## 11    156          156
## 12    165          165
## 13    169          169
## 14    178          178
## 15    182          182
## 16    191          191
## 17    195          195
## 18    204          204
## 19    208          208
## 20    217          217
## 21    221          221
## 22    230          230
## 23    234          234
## 24    243          243
## 25    247          247
## 26    256          256
## 27    260          260
## 28    269          269
## 29    273          273
## 30    282          282
## 31    286          286
## 32    295          295
## 33    299          299
## 34    308          308
## 35    312          312
## 36    321          321
## 37    325          325
## 38    334          334
## 39    338          338
## 40    347          347
## 41    351          351
## 42    360          360
## 43    364          364
## 44    373          373
## 45    377          377
## 46    386          386
## 47    390          390
## 48    399          399
## 49    403          403
## 50    412          412
## 51    416          416
## 52    425          425
## 53    429          429
```

```
## 54    438          438
## 55    442          442
## 56    451          451
## 57    455          455
## 58    464          464
## 59    468          468
## 60    477          477
## 61    481          481
## 62    490          490
## 63    494          494
## 64    503          503
## 65    507          507
## 66    516          516
## 67    520          520
## 68    529          529
## 69    533          533
## 70    542          542
## 71    546          546
## 72    555          555
## 73    559          559
## 74    568          568
## 75    572          572
## 76    581          581
## 77    585          585
## 78    594          594
## 79    598          598
## 80    607          607
## 81    611          611
## 82    620          620
## 83    624          624
## 84    633          633
## 85    637          637
## 86    646          646
## 87    650          650
## 88    659          659
## 89    672          672
## 90    685          685
## 91    698          698
## 92    711          711
## 93    724          724
## 94    737          737
## 95    750          750
```

Single knapsack problem:

Knapsack to metoda optymalizacyjna, linear programming dla liczb całkowitych.

WYkorzystując po 100 wartości z każdego unikalnego kuponu przerowadzono wnioskowanie. Nalezy zanzaczyć iż wykorzystując po jednej wartości każdego kuponu wyniki sa identyczne. Dla przypomnienia poszukujemy takiej kombiacji kuponów, która będzie się sumować do kolejnych iloczynów setki aż do tysiąca.

```
library(adagio)

cupon_we_have = rep(both$wygenerowane,100)
```

```
cupon_we_want = seq(100,1000,100)

for(i in cupon_we_want){

solution <- try(knapsack(p=cupon_we_have[cupon_we_have<i], w=cupon_we_have[cupon_we_have<i], cap=i))

print(solution$profit)
}
```

```
## [1] 91
## [1] 195
## [1] 299
## [1] 399
## [1] 499
## [1] 599
## [1] 699
## [1] 798
## [1] 898
## [1] 998
```

Wykorzystując kupony z bazy danych nie jest mozliwe osiągnięcie jakiegokolwiek zysku.

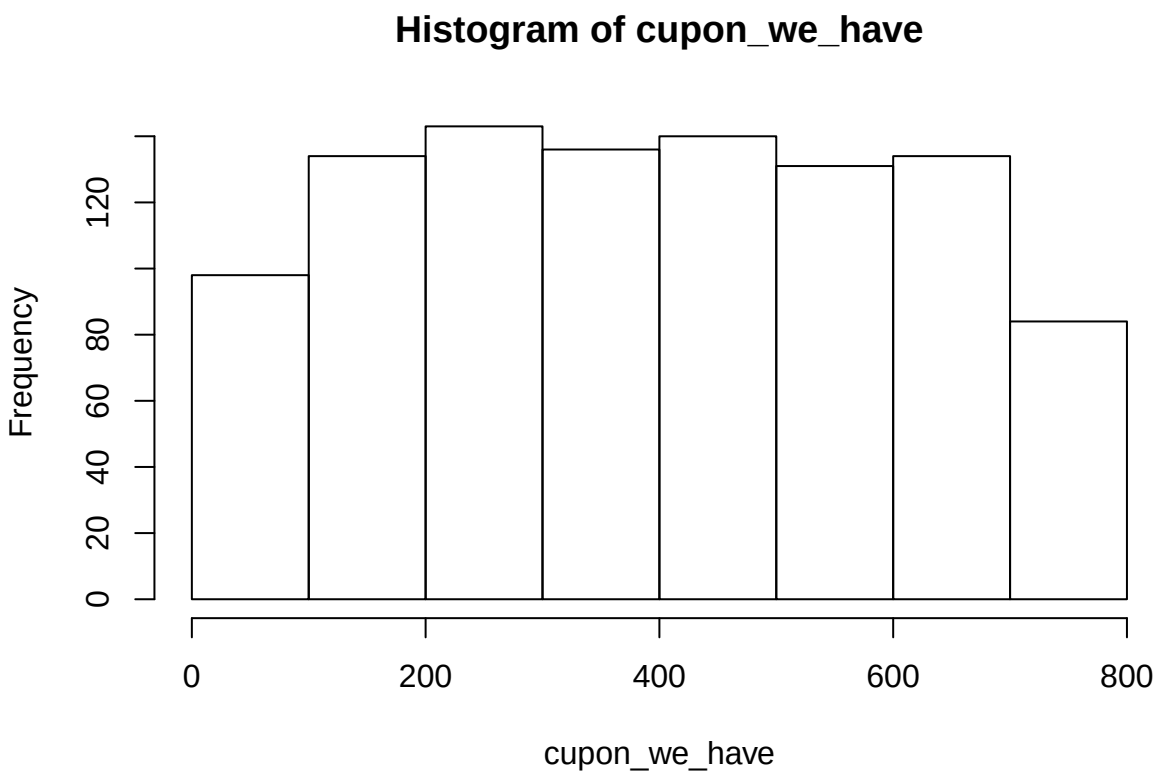
Dodatek ukazujący co gdyby na kuponach były liczby z rozkładu ciągłego (zaokrąglone do całkowitych):

Symulacja z kuponami z rozkładu ciągłego z zaokraglonymi wartościami. Postaramy się odnaleźć jak najwięcej 1000 żłotych.

```
set.seed(1234)

cupon_we_have = round(c(runif(1000,26,750)))

hist(cupon_we_have,10)
```




```
cupon_we_want=c(1000)

prof = 0

repeat{

solution <- try(mknapsack(p=cupon_we_have, w=cupon_we_have, k=cupon_we_want,bck=0))

if(class(solution)=="try-error") break

diff = NULL

for(i in 1:length(cupon_we_want)) diff[i] = cupon_we_want[i] - sum(cupon_we_have[solution$ksack==i])

if(all(diff==0)) cupon_we_want = c(cupon_we_want,1000) else break

prof = prof + 1
}

c = cupon_we_have
names(c) = solution$ksack
c = c[sort(as.numeric(names(c)),index.return=TRUE)$ix]
c = c[!names(c)==0]
c
```

```
##   1   1   1   1   1   1   2   2   2   3   3   3   3   3   4   4   4   4
```

```
## 108 477  33 194 161  27 467 477  56 649 231  55  36  29 490 398  59  26
##   4   5   5   5   5   6   6   6   6   7   7   7   7   8   8   8   8   8
##  27 508 421  45  26 528 238 194  40 695 233  37  35 632 219  79  42  28
##   9   9   9   9   9   9  10  10  10  10  10  11  11  11  11  11  12  12
## 255 245 141 184 137  38 613 218  63  52  54 407 356 157  47  33 688 247
##  12  12  13  13  13  13  14  14  14  14  14  15  15  15  15  15  16  16
##  28  37 628 172 152  48 393 213 252 114  28 576 265  91  33  35 744 256
##  17  17  18  18  18  19  19  19  19  20  20  20  20  20  21  21  21  21
## 611 389 427 494  79 476 377  77  70 516 203 199  45  37 580 250 142  28
##  22  22  22  22  23  23  23  23  24  24  24  25  25  25  26  26  26  26
## 545 249 129  77 391 391 168  50 384 570  46 640 258 102 652 251  51  46
##  27  27  27  27  28  28  28  28  29  29  29  29  30  30  30  30  31  31
## 538 304  99  59 394 435 121  50 672 146 123  59 593 308  49  50 402 510
##  31  32  32  32  33  33  33  33  33  34  34  34  35  35  35  35  36  36
##  88 697 229  74 368 420 123  30  59 677 250  73 675 174 106  45 678 243
##  36  37  37  37  37  38  38  38  38  39  39  39  39  39  39  40  40  40
##  79 396 435 114  55 563 262 120  55 138 341 185 228  62  46 542 115 231
##  40  40  41  41  41  42  42  42  42  43  43  43  44  44  44  44  45  45
##  67  45 714 208  78 687 118  85 110 711 192  97 603 257  65  75 565 332
##  45  46  46  46  47  47  47  47  48  48  48  49  49  49  49  50  50  50
## 103 689 189 122 746  81 119  54 708 160 132 378 390 123 109 386 485  62
##  50  51  51  51  52  52  52  53  53  53  53  54  54  54  55  55  55  56
##  67 723 196  81 688 245  67 365 459 121  55 684 262  54 483 261 256 655
##  56  56  57  57  57  57  58  58  58  59  59  59  59  60  60  60  61  61
## 276  69 390 375 113 122 738 174  88 284 480  99 137 563 344  93 436 462
##  61  62  62  62  63  63  63  63  64  64  64  65  65  65  65  66  66  67
## 102 736 180  84 444 298 168  90 642 275  83 741 110  87  62 749 251 428
##  67  67  67  68  68  68  68  69  69  69  70  70  70  70  71  71  71  71
## 337 161  74 443 339 164  54 487 338 175 607 249  90  54 574 229  99  98
##  72  72  72  73  73  73  74  74  74  74  75  75  75  76  76  76  77  77
## 449 335 216 539 333 128 576 213 111 100 432 373 195 461 407 132 718 183
##  77  78  78  78  79  79  79  79  80  80  81  81  81  82  82  82  83  83
##  99 635 203 162 484 319 113  84 564 436 489 409 102 745 104 151 665 219
##  83  84  84  84  85  85  85  86  86  86  87  87  87  87  88  88  88  89
## 116 613 268 119 621 270 109 630 217 153 556 248  98  98 738 163  99 489
##  89  89  90  90  90  91  91  91  92  92  92  92  93  93  93  94  94  94
## 332 179 504 333 163 582 228 190 556 220 114 110 740 125 135 575 285 140
##  95  95  95  95  96  96  96  97  97  97  97  98  98  98  99  99  99 100
## 436 373 129  62 701 167 132 488 298 152  62 533 276 191 642 202 156 388
## 100 100 101 101 102 102 102 103 103 103 104 104 104 105 105 105 106 106
## 394 218 607 393 384 380 236 603 293 104 436 371 193 611 250 139 437 442
## 106 107 107 107 108 108 108 109 109 109 110 110 110 111 111 111 112 112
## 121 569 234 197 743 146 111 525 321 154 736 166  98 586 165 249 725 275
## 113 113 113 114 114 114 114 115 115 115 116 116 116 117 117 117 118 118
## 603 229 168 411 337 154  98 458 354 188 434 403 163 453 282 265 638 170
## 118 119 119 119 120 120 120 121 121 121 122 122 122 123 123 123 124 124
## 192 460 357 183 568 223 209 567 314 119 575 278 147 605 172 223 621 379
## 125 125 125 125 126 126 126 127 127 127 128 128 128 129 129 129 130 130
## 420 325 149 106 691 162 147 594 292 114 560 182 258 530 321 149 620 171
## 130 131 131 131 132 132 133 133 133 134 134 134 135 135 135 136 136 136
## 209 500 335 165 715 285 467 322 211 575 251 174 528 240 232 486 347 167
## 137 137 137 138 138 138 139 139 139 140 140 140 141 141 141 142 142 142
## 416 423 161 583 247 170 520 272 208 630 182 188 650 244 106 502 297 201
## 143 143 144 144 144 145 145 145 146 146 147 147 147 147 148 148 149 149
```

```
## 692 308 557 271 172 665 229 106 716 284 368 306 164 162 699 301 323 294
## 149 149 150 150 150 151 151 152 152 152 153 153 154 154 154 155 155 155
## 212 171 400 359 241 734 266 509 226 265 693 307 564 213 223 539 210 251
## 156 156 157 157 157 158 158 158 159 159 159 160 160 160 161 161 162 162
## 709 291 455 312 233 555 205 240 378 435 187 582 235 183 430 570 504 283
## 162 163 163 163 164 164 164 165 165 165 166 166 166 167 167 167 168 168
## 213 619 205 176 551 177 272 576 177 247 331 498 171 502 199 299 524 182
## 168 169 169 170 170 170 171 171 171 172 172 172 173 173 174 174 174 175
## 294 712 288 527 296 177 562 220 218 556 245 199 694 306 414 311 275 523
## 175 176 176 176 177 177 177 178 178 178 179 179 180 180 181 181 182 182
## 477 328 327 345 574 213 213 588 206 206 441 559 575 425 588 412 611 389
## 183 183 184 184 185 185 185 186 186 186 187 187 187 188 188 188 189 189
## 726 274 654 346 328 428 244 397 296 307 324 453 223 505 271 224 683 317
## 190 190 191 191 191 192 192 193 193 194 194 194 195 195 196 196 196 197
## 701 299 334 344 322 506 494 700 300 503 263 234 566 434 482 289 229 470
## 197 197 198 198 198 199 199 200 200 200 201 201 202 202 202 203 203 203
## 339 191 519 248 233 465 535 442 310 248 674 326 400 330 270 422 315 263
## 204 204 204 205 205 205 206 206 206 207 207 207 208 208 208 209 209 210
## 479 309 212 440 312 248 397 311 292 364 398 238 405 396 199 432 568 504
## 210 211 211 212 212 213 213 214 214 214 215 215 215 216 216 216 217 217
## 496 579 421 460 540 579 421 463 378 159 457 270 273 510 245 245 606 394
## 218 218 219 219 220 220 221 221 222 222 223 223 223 224 224 225 225 226
## 627 373 699 301 455 545 695 305 484 516 320 336 344 469 531 633 367 386
## 226 226 227 227 228 228 229 229 230 230 230 231 231 232 232 233 233 234
## 318 296 743 257 651 349 608 392 441 315 244 472 528 458 542 488 512 637
## 234 235 235 236 236 237 237 238 238 239 239 240 240 241 241 242 242 243
## 363 395 605 607 393 507 493 589 411 377 623 624 376 466 534 431 569 637
## 243 244 244 244 245 245 246 246 247 247 248 248 249 249 250 250 251 251
## 363 284 372 344 397 603 523 477 417 583 562 438 459 541 418 582 475 525
## 252 252 253 253 254 254 255 255 255 256 256 257 257 258 258 259 259
## 426 574 495 505 536 464 366 263 371 309 691 418 582 655 345 550 449
```

Wygrana przy tym konkretnym losowaniu mogłaby osiągnąć 257 tysięcy złotych. Dla innych losowań wartości kuponów z zaprezentowanego rozkładu, wygrane są bardzo zbliżone. Dlatego z oczywistych względów w konkursie Amino zaimplementowano szczególny algorytm generowania liczb losowych. Jednak pytaniem pozostaje jak czesto i czy wogole występują kupony pozwalające na wygraną.