# Playing Regex Golf with Genetic Programming

MACHINE LEARNING LAB

A. Bartoli, A. De Lorenzo, E. Medvet, F. Tarlao
University of Trieste, Italy

# The problem (“Regex Golf”)


- Specific kind of code golf
- Writing the shortest regular expression which:
  - matches all the strings in a given list
  - does not match any strings in another list

# Regex Golf - Naive solution

Positive examples

- The Phantom Menace
- Attack of the Clones
- Revenge of the Sith
- A New Hope
- The Empire Strikes Back
- Return of the Jedi

Negative examples

- The Wrath of Khan
- The Search for Spock
- The Voyage Home
- The Final Frontier
- The Undiscovered Country
- Generations
- First Contact
- Insurrection
- Nemesis

```
The Phantom Menace|Attack of the Clones|Revenge of the Sith|A New Hope|The Empire Strikes Back|Return of the Jedi
```

# Regex Golf - Best solution

Positive examples

- The Phantom Menace
- Attack of the Clones
- Revenge of the Sith
- A New Hope
- The Empire Strikes Back
- Return of the Jedi

Negative examples

- The Wrath of Khan
- The Search for Spock
- The Voyage Home
- The Final Frontier
- The Undiscovered Country
- Generations
- First Contact
- Insurrection
- Nemesis

```
m | [tN]|B
```

# 16 difficult instances

Match all of these…

- ✓ aerate aerate arrest errant serene tanner testes
- ✓ aerate assent assent assert rester retest tenant
- ✓ aerate assert rearer renter resent serene teaser
- ✓ aerate easter easter tenant tester testes tsetse
- ✓ arrest arrest easter entree errant resent senate
- ✓ assent assess assets estate resent staree teaser
- ✓ assert astern renter rerent resent staree street
- ✓ assert enseat entree errata rennet teaser tsetse
- ✓ assert rennet renter reseat rester serene tenant
- ✓ assess easter estate rennet rennet tenant testes
- ✓ assess easter estate rerent resent retest snarer
- ✓ assess renter renter searer seater snarer testes
- ✓ astern enseat entree serene staree tartar tartar
- ✓ astern rennet retest searer snarer tartar tester
- ✓ enseat errata seater senate strata teaser tsetse
- ✓ entree searer staree taster taster tenant testes
- ✓ rerent rester tanner tartar teaser teaser testes

and none of these…

- ✗ aerate astern assess enseat senate street tsetse
- ✗ aerate rennet errant enseat rerent senate testes
- ✗ arrest assess assess assent astern searer testes
- ✗ assert assess errata enseat earner seater serene
- ✗ assert astern staree senate snarer tanner tester
- ✗ assert strata rerent rerent tanner testes tsetse
- ✗ assess easter entree rester reseat seater tartar
- ✗ astern assets rearer rearer assess rearer testes
- ✗ astern easter taster serene reseat taster tester
- ✗ earner entree rerent reseat teaser strata staree
- ✗ earner errant estate taster reseat estate taster
- ✗ enseat astern arrest enseat searer seater tenant
- ✗ errant errant senate renter rearer street tsetse
- ✗ rennet rennet assent errant rester staree tester
- ✗ rennet snarer senate retest tanner tartar tsetse
- ✗ retest astern arrest tsetse strata senate tsetse
- ✗ searer errant teaser staree assess teaser tsetse

Match all of these…

- ✓ x
- ✓ xx
- ✓ xxxx
- ✓ xxxxxxxx
- ✓ xxxxxxxxxxxxxxxx
- ✓ xxxxxxxxxxxxxx... [32 chars]
- ✓ xxxxxxxxxxxxxx... [64 chars]
- ✓ xxxxxxxxxxxxxx... [128 chars]
- ✓ xxxxxxxxxxxxxx... [256 chars]
- ✓ xxxxxxxxxxxxxx... [512 chars]
- ✓ xxxxxxxxxxxxxx... [1024 chars]

and none of these…

- ✗ xxx
- ✗ xxxxx
- ✗ xxxxxxx
- ✗ xxxxxxxxxx
- ✗ xxxxxxxxxxxx
- ✗ xxxxxxxxxxxxxxxxxxxxxxxxxxxx
- ✗ xxxxxxxxxxxxxx... [48 chars]
- ✗ xxxxxxxxxxxxxx... [160 chars]
- ✗ xxxxxxxxxxxxxx... [401 chars]
- ✗ xxxxxxxxxxxxxx... [600 chars]
- ✗ xxxxxxxxxxxxxx... [1025 chars]

# A lot of vibrant discussions...

GitHub Gist | Search... | All Gists | Sign up for a

jpsim / **answers.md**

Last active on 20 Dec 2013

Regex Golf Answers (http://regex.alf.nu/) HN: https://news.ycombinator.com/item?id=6941231

answers.md — Raw

1. Plain Strings (207): `foo`
2. Anchors (208): `k$`
3. Ranges (202): `^[a-f]*$`
4. Backrefs (201): `(...).*\1`
5. Abba (169): `^(.(?!(ll|ss|mm|rr|tt|ff|cc|bb)))*$|^n|ef`
6. A man, a plan (177): `^(.)[^p].*\1$`
7. Prime (286): `^(?!(..+)\1+$)`
8. Four (199): `(.)(.\1){3}`
9. Order (198): `^[^o].....?$`
10. Triples (507): `(^39|^44)|(^([0369]|([147][0369]*[258])|(([258]|[147][0369]*[147])([0369]*|[258][0369]*[147])([147]|[258][0369]*[258])))*$)`
11. Glob (364): `^((\*(er|f|i|p|t|v))|(b|c(h|o|r)|do|l|mi|p(a|r|u)|re|w))`
12. Balance (286): `^(<(<(<(<(<(<>)*>)*>)*>)*>)*>)*$`
13. Powers (56): `^((((((((((x)\10?)\9?)\8?)\7?)\6?)\5?)\4?)\3?)\2?)\1?$`

Total Score: 3060

**nadinengland** commented on 20 Dec 2013

Abba (190): ^((?!(.)(.)\3\2).)*$

**jonathanmorley** commented on 20 Dec 2013

Abba (193): ^(?!.*(.)(.)\2\1)

**alkino** commented on 20 Dec 2013

Power (58): `^(x|(.*)\2)$`

Glob (378): `^(\*(er|f|i|p|t|v)|b|c[^a]|do|l|mi|p|re|w)`

**pochmann** commented on 20 Dec 2013

Triples (588): `(00[039]|12|015|50)$|1..?4|4.2|1.7|6.0|006`

Powers (72): `^(xx?|xxxx|x{8}|x{16}|x{32}|(x{64})*)$`

# ...really a lot...

**bbarry** commented on 28 Dec 2013

@fugyk: ~~I don't think it is possible to create a regular expression that matches any list of words in alphabetical order (one of 6 letter combinations only would be incredibly long). I guess the hint is a reference back to #9 which is possible without cheating (even though it says to cheat).~~ I stand corrected, ty @berndjendrissek.

You could create the brute force expression of every possible 6 letter word in order containing the letters `aenrst` (which is still terribly long), or the one containing all the words in this list (-69 points, goes: `^(aerate?)*(arrest ?)*...$` and is 409 characters long). Anything beyond that I feel is cheating so badly that it doesn't matter how much more you do.

I want to know what the meaning of the hitchhikers guide reference in long count v2 is all about. My solution there cheats a lot (it matches far more than the intended binary nibble count sequence, just happens to not match any of his counter examples because none of them swap more than 1 nibble out).

**berndjendrissek** commented on 28 Dec 2013

@bbarry: You don't need to list every possible word for a non-cheating solution. It suffices to exclude any adjacent pair with a common prefix (which may have zero length) followed by even a single out-of-order letter:

```
^(?!.*\b(([^ ]*)t[^ ]+ \2[^t][^ ]*|([^ ]*)s[^ ]+ \3[^st][^ ]*|([^ ]*)r[^ ]+ \4[aen][^ ]*|([^ ]*)n[^ ]+ \5[ae][^ ]*|([^ ]*)e[^ ]+ \6[a][^ ]*).*$)
```

[−] **Overv** 8 punti 6 mesi fa

You can do Abba with 193 points:

```
^(?!.*(.)(.)\2\1)
```

permalink genitore

[−] **ProRustler** 2 punti 6 mesi fa

GAH! I was missing the first .*; this makes sense now, thanks for posting.

permalink genitore

[−] **[deleted]** 3 punti 6 mesi fa

The solution for prime is amazing, good job.

This is a perfect match (but lower score) solution for powers:

```
^((((((((((x)\10?)\9?)\8?)\7?)\6?)\5?)\4?)\3?)\2?)\1?$
```

Add.: part of me wants perfect matches to get significant bonus point, heh.

permalink genitore

[−] **Bisqwit** 2 punti 6 mesi fa

Well, there's this one which ties the false-positives one. Use it if you are pedantic :-)

```
^(x|(xx){1,4}|((((((x{16})\8?)\7?)\6?)\5?)\4?)\3?)$
```

Even though it falsely approves "xxxxxx", not included in the fail-testcases.

permalink genitore

[−] **[deleted]** 2 punti 6 mesi fa

I fiddled a bit more, and I think I'll take

```
^(x|xx|(x{4}){1,6}|(x{32}){1,4}|(x{32}){6,})$
```

for 65 points with no false positives. :)

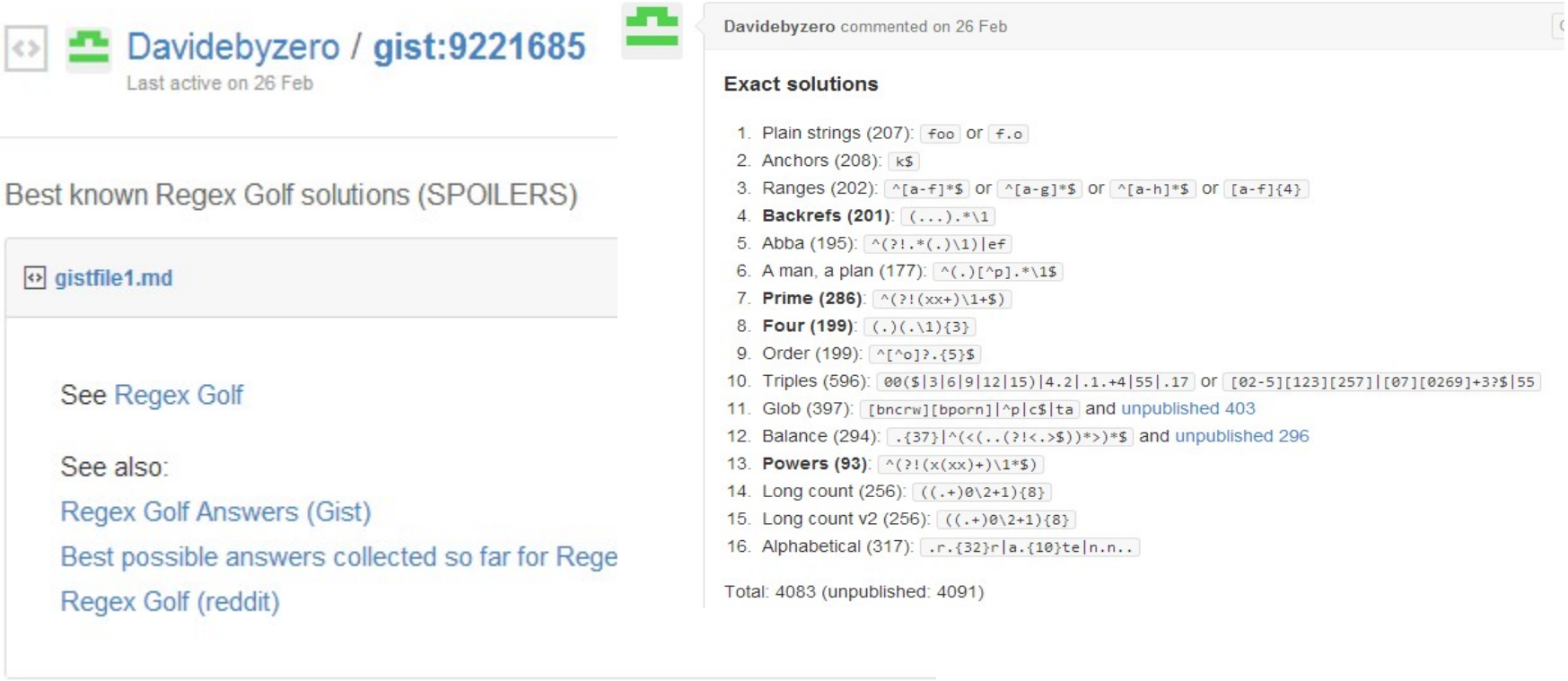
Add.: scratch that,

```
^(x|(xx){1,10}|(x{32}){1,4}|(x{32}){6,})$
```

for 69 looks better.

permalink genitore

# February 26-th, 2014



Davidebyzero / gist:9221685

Last active on 26 Feb

Best known Regex Golf solutions (SPOILERS)

gistfile1.md

See Regex Golf

See also:

Regex Golf Answers (Gist)

Best possible answers collected so far for Rege

Regex Golf (reddit)

**Davidebyzero** commented on 26 Feb

**Exact solutions**

1. Plain strings (207): `foo` or `f.o`
2. Anchors (208): `k$`
3. Ranges (202): `^[a-f]*$` or `^[a-g]*$` or `^[a-h]*$` or `[a-f]{4}`
4. **Backrefs (201)**: `(...).*\1`
5. Abba (195): `^(?!.*(.)\1)|ef`
6. A man, a plan (177): `^(.)[^p].*\1$`
7. **Prime (286)**: `^(?!(xx+)\1+$)`
8. **Four (199)**: `(.)(.\1){3}`
9. Order (199): `^[^o]?.{5}$`
10. Triples (596): `00($|3|6|9|12|15)|4.2|.1.+4|55|.17` or `[02-5][123][257]|[07][0269]+3?$|55`
11. Glob (397): `[bncrw][bporn]|^p|c$|ta` and unpublished 403
12. Balance (294): `.{37}|^(<(..(?!<.>$))*>)*$` and unpublished 296
13. **Powers (93)**: `^(?!(x(xx)+)\1*$)`
14. Long count (256): `((.+)0\2+1){8}`
15. Long count v2 (256): `((.+)0\2+1){8}`
16. Alphabetical (317): `.r.{32}r|a.{10}te|n.n..`

Total: 4083 (unpublished: 4091)

# Automatic Regex Golf


bbarry commented on 7 Jan

using finite state machines

## Regex Golf with Arbitrary Lists

We can define a convenience function to do this finding and verifying, and we might as well do it in both directions (e.g. separating winners from losers and losers from winners). We will also report the number of characters in the solution and the *competitive ratio* of the solution: the ratio between the length of a trivial solution and the solution found (a trivial solution for the set of winners {'one', 'two', 'three'} is the disjunction '^(one|two|three)$').

```
def findboth(A, B):
    "Find a regex to match A but not B, and vice-versa.  Print s
ummary."
    for (W, L, legend) in [(A, B, 'A-B'), (B, A, 'B-A')]:
        solution = findregex(W, L)
        assert verify(solution, W, L)
        ratio = len('^(' + OR(W) + ')$') / float(len(solution))
        print '%3d chars, %4.1f ratio, %2d winners %s: %r' % (
            len(solution), ratio , len(W), legend, solution)
```

# Our previous GP-based tool

**Automatic** regex generation from **examples**

For **data extraction**

| Input string | String to match |
|---|---|
| this is a valid ip 127.0.0.1 | 127.0.0.1 |
| 12.3 is just a number | |
| ping from 192.168.0.1 | 192.168.0.1 |
| today is 7/11/2012 | |
| msg to 66.231.55.67 sent | 66.231.55.67 |
| telnet 17.23.133.22:8080 | 17.23.133.22 |
| this is old plain text | |
| germany-italy 1-2 | |
| 172.30.40.254 is a server ip | 172.30.40.254 |
| It's nine o'clock on a Saturday | |

*IEEE Computer*, *GECCO Hot Off the Press*

http://regex.inginf.units.it

# Key observations

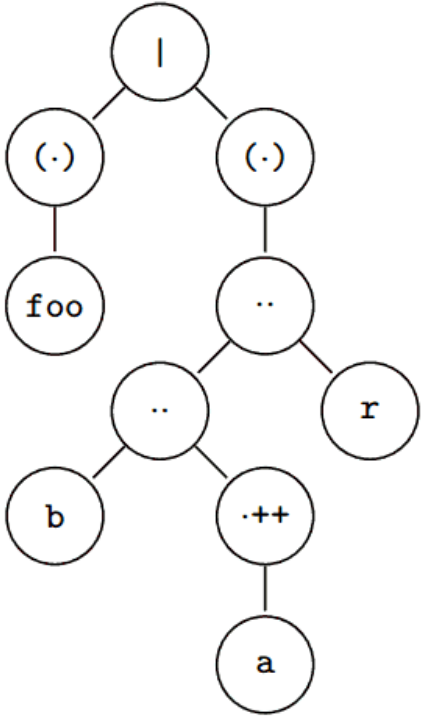
- We need a **classifier**---not an **extractor**
  - No need to identify boundaries

- No need to infer a **general pattern**
- No need to process streams with unknown items
  - We need to “overfit”

# Our approach

- Candidate regex = tree

- Internal nodes: usual regex operators
  - No greedy/lazy quantifiers
    (execution time too long to be practical)
- Leave nodes:
  - a-z, A-Z, ^, $, ..
  - **problem-dependent elements** (*next slide*)



# Problem-dependent elements

- All characters in matches
- All partial ranges including those characters
- All “most useful” n-grams (n=2,3,4)
  - Build all n-grams
  - Score each n-gram based on its frequency:
    +1 for each match, -1 for each unmatch
  - Rank n-grams
  - Select the smallest set totalling M points
    (M being the number of matches)

# Evolutionary search

- 500 individuals, 1000 generations
- 32 independent searches

- Multiobjective fitness (NSGA-II)
- Minimize
  - Number of misclassifications
  - Length

# Problem in detail

- 16 instances, each with:
  - Set of matches M
  - Set of unmatches U
  - Weight w (a “difficulty” coefficient)

- Score of regex r on a given instance:
  - w * #misclassifications - length(r)

# Problem in detail

| | Problem name | $\|M\|$ | $\|U\|$ | $w_{\mathcal{I}}$ | Ideal score | Best human score | Best human solution |
|---|---|---|---|---|---|---|---|
| 1 | Plain strings | 21 | 21 | 10 | 210 | 207 | `foo` |
| 2 | Anchors | 21 | 21 | 10 | 210 | 208 | `k$` |
| 3 | Ranges | 21 | 21 | 10 | 210 | 202 | `^[a-f]*$` |
| 4 | Backrefs | 21 | 21 | 10 | 210 | 201 | `(...).*\1` |
| 5 | Abba | 21 | 22 | 10 | 210 | 193 | `^(?!.*(.)(.)\2\1)` |
| 6 | A man, a plan | 19 | 21 | 10 | 190 | 177 | `^(.)[^p].*$` |
| 7 | Prime | 20 | 20 | 15 | 300 | 286 | `^(?!(..+)\1$` |
| 8 | Four | 21 | 21 | 10 | 210 | 199 | `(.)(.\1){3}` |
| 9 | Order | 21 | 21 | 10 | 210 | 199 | `^.5[^e]?$` |
| 10 | Triples | 21 | 21 | 30 | 630 | 596 | `00($\|3\|6\|9\|12\|15)\|4.2\|.1.+4\|55\|.17` |
| 11 | Glob | 21 | 21 | 20 | 420 | 397 | `ai\|c$\|^p\|[bcnrw][bnopr]` |
| 12 | Balance | 32 | 32 | 10 | 320 | 289 | `^(<(<(<(<<?>?>\|.9)>)>)>)>)$` |
| 13 | Powers | 11 | 11 | 10 | 110 | 93 | `^(?!(.(..)+)\1*$)` |
| 14 | Long count | 1 | 20 | 270 | 270 | 254 | `((.+)0 \2?1 ){7}` |
| 15 | Long count v2 | 1 | 21 | 270 | 270 | 254 | `((.+)0 \2?1 ){7}` |
| 16 | Alphabetical | 17 | 17 | 20 | 340 | 317 | `.r.{32}r\|a.{10}te\|n.n..` |
| | Total | | | | 4320 | 4072 | |

# Baseline

- GP-RegexExtract (our data extraction tool)
- “Norvig” (January 2014)
  - Deterministic algorithm widely discussed on the web
  - IMPORTANT
    - Not developed for this challenge
    - Designed for completely avoiding misclassifications
    - Comparison not fully fair...but the only algorithm we were aware of

# Results: Warning

- The web site does **not** collect scores/rankings
- Programmers advertised solutions on **forums**
  - GitHub, Reddit, Hacker News
- Sometimes **only scores** without any evidence
- Sometimes slightly **improving earlier results by other programmes**

- No evidence of **time** spent

# Great Results !

- 6-th/8-th worldwide
  - At the time
  - To the best of our knowledge

- **Without any hint** from other programmers
- Much better than the **baseline**

| | Player | Score |
|---|---|---|
| | *Total ideal score* | 4320 |
| | *Best human score* | 4072 |
| 1 | geniusleonid | 4006 |
| 2 | k_hanazuki | 3785 |
| 3 | bisqwit | 3753 |
| 4 | AlanDeSmet | 3736 |
| 5 | adamhiker | 3693 |
| | *GP-RegexGolf* ($n_P = 1500$) | 3412 |
| | *GP-RegexGolf* ($n_P = 1000$) | 3201 |
| 6 | adamschwartz | 3181 |
| 7 | flyingmeteor | 3171 |
| | *GP-RegexGolf* ($n_P = 500$) | 3090 |
| 8 | jpsim | 3060 |
| 9 | ItsIllak | 2939 |
| 10 | bg666 | 2683 |
| | *GP-RegexExtract* | 249 |
| | *Norvig-RegexGolf* | −665 |

# Execution time, Actual regexes

| Problem | GP-RegexGolf |
|---|---|
| 1 | 53 |
| 2 | 52 |
| 3 | 53 |
| 4 | 38 |
| 5 | 34 |
| 6 | 20 |
| 7 | 33 |
| 8 | 19 |
| 9 | 46 |
| 10 | 45 |
| 11 | 44 |
| 12 | 56 |
| 13 | 71 |
| 14 | 94 |
| 15 | 95 |
| 16 | 66 |
| Total | 820 |

| Problem | Regular expression |
|---|---|
| 1 | `foo` |
| 2 | `k$` |
| 3 | `(^..[a-f][a-f])` |
| 4 | `v|[^b][^o][^p]t|ngo|lo|[n]o|rp|rb|ro|ro|rf` |
| 5 | `z|.u|nv|st|ca|it` |
| 6 | `oo|x|^k|ed|^m|ah|^r|v|^t` |
| 7 | `^(?=((?:x[A-Zx])+))\1x` |
| 8 | `ell|j|W|ele|o.o|Ma|si|de|do` |
| 9 | `ch|[l-p]o|ad|fi|ac|ty|os` |
| 10 | `24|55|02|54|00|95|17` |
| 11 | `lo|ro|^p|(?=((c)+))\1r|en|^w|y.|le|^p|rr` |
| 12 | `((?=((?:<<\>\>\>)*))\2(?=((?:<<<(?=(<*))\4\>\><<<<)*))\3(?=((?:<<<<<\>\>\>(?=(<*))\6\>\>\>)*))\5(?=((?:<<<<<<)*))\7^(?=((?:<<\><<)*))\8(?=((?:<<<\>\>\>)*))\9<<)` |
| 13 | `^(?=(((x|^)x)+))\1$` |
| 14 | `0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111` |
| 15 | `0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111` |
| 16 | `tena|[^et][^etren](?=((?:(?:ren|eren.(?=((?:(?:ren|[^ren]))+))\2|eren.(?=((?:(?:ren|[^ren]))+))\3))+))\1|eas` |

# Summary

- Evolutionary computation has reached a level in which **it may successfully compete with human programmers**
- In scenarios explicitly designed to test their **practical skills** and **creativity**
- And, It may do so **without any starting hint** or **external help**.

# Web application

A web-based prototype is public available at http://regex.inginf.units.it/golf