Webquines, The Collatz Graph and a Weightless Website

Ian Stark

Lab Lunch

Tuesday 1 February 2011

http://homepages.inf.ed.ac.uk/stark

A modest URL

This talk is about the construction of the following URL, which contains a website of unknown extent:

```
data:text/html,<script>e=encodeURI(d='<title>Weightless_Website_Page_Q</title><h1>Weightless_website_page_Q</h1>Explore_the_Collatz_graph:_from_page_<em>n</em>_you_can_go_to_page_2<em>n</em>,_and_to_page_(2<em>n</em>-1)/3_if_that_is_a_whole_number.<p>V<p>This_is_page_Q._From_here_you_can_go@Z</a>~There_is_a_pirate_on_page_42.~<img_src="">~_or@Z</a>.~_to_<a_href="data:text/html,<script>e=encodeURI(d=!27X!27);String.prototype.r=d.replace;s=d.r(/!21/g,!27%!27).r(/!5F/g,!27_!27).split(!27!7E!27);document.write(s[0].r(/Q/g,K).r(/Z/,2*K).r(/V/,s[K==42?2:1]).r(/@/,s[4].r(/!4B/g,2*K).r(/X/,e))+(K!253==2?(s[3].r(/Z/,(2*K-1)/3).r(/@/,s[4].r(/!4B/g,(2*K-1)/3).r(/X/,e))):!27.!27));!3C/script>">page_');String.prototype.r=d.replace;s=d.r(/!/g,'%').r(/_/g,'%20').split('~');document.write(s[0].r(/Q/g,1).r(/Z/,2).r(/V/,s[1]).r(/@/,s[4].r(/K/g,2).r(/X/,e))+'.');</script>+'.');</script>
```

Quines

A *quine* is a program which outputs its own source code.

To write a quine we need a language with:

- Some means of computation
- Some kind of output
- A notion of source code

It's not necessary that the computation be Turing complete, nor that the language have any kind of input.

The name "quine" was coined by Hofstadter in *Gödel, Escher, Bach* after the 20th century philosopher W. V. Quine

Quines

A *quine* is a program which outputs its own source code.

The empty program is a quine, but quite a dull one.

Prize for *Worst Abuse of the Rules*, 1994 International Obfuscated C Code Contest, awarded to Szymon Rusinkiewicz, MIT, for the world's smallest self-replicating program.

Quines

A *quine* is a program which outputs its own source code.

Use of introspection can give quines, but unilluminating ones.

```
10 REM Not a very good Quine in BASIC
20 LIST

#!/bin/sh
cat ${0}
# Not a very good Quine in sh
```

Quines

A *quine* is a program which outputs its own source code.

Quines become interesting when they really compute the output.

```
char* s = "printf(s);";
printf(s);
```

Quines

A *quine* is a program which outputs its own source code.

Quines become interesting when they really compute the output.

```
char* s = "char* s =\"...\";\nprintf(s);";
printf(s);
```

Quines

A *quine* is a program which outputs its own source code.

Quines become interesting when they really compute the output.

```
char* s = "char* s =\"char* s =...\";\nprintf(s);";
printf(s);
```

Quines

A *quine* is a program which outputs its own source code.

Quines become interesting when they really compute the output.

```
char* s = "char* s = XX;\nprintf(s);";
printf(s);
```

Quines

A *quine* is a program which outputs its own source code.

Quines become interesting when they really compute the output.

```
char* s = "char* s = XX;\nprintf(quotedselfinsert(s));";
printf(quotedselfinsert(s));
```

Quines

A *quine* is a program which outputs its own source code.

Quines become interesting when they really compute the output.

```
char* s =
"char* s =\nXX;\nprintf(quotedselfinsert(s));{payload}";
printf(quotedselfinsert(s));{payload}
```

Quines

A *quine* is a program which outputs its own source code.

Quines become interesting when they really compute the output.

```
char* s=
"char* s=%c%c%s%c;%cprintf(s,10,34,s,34,10);{payload}";
printf(s,10,34,s,34,10);{payload}
```

Quines

Quines have been written for every language you might imagine, and more. For example:

- English (Quine's original)
- SQL, querying an empty database
- sed, as replacement of regexps is enough computation
- Postscript, generating a page bitmap of its own source
- LZ-decompression, giving a zipfile which unzips to itself
- Multi-stage quines: P_1 outputs P_2 outputs…P_n outputs P_1
- Multi-language quines: P_i all in different scripts

Wikipedia claims that the "first known" self-reproducing programme was written in the 1960s at Edinburgh by Hamish Dewar in Atlas Autocode, but lacks citations…

Quines

Reflections on Trusting Trust

Ken Thompson, 1983 Turing Award lecture

http://cm.bell-labs.com/who/ken/trust.html

Compile-a-Virus: W32/Induc-A

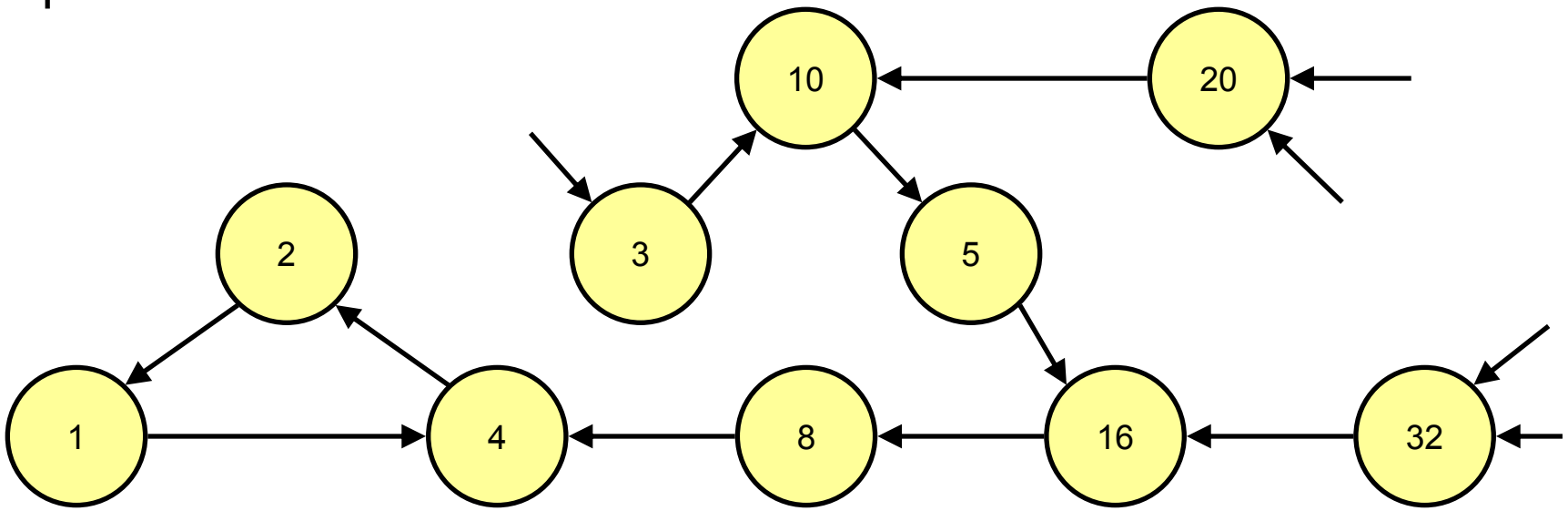
http://nakedsecurity.sophos.com/2009/08/18/compileavirus/

The Collatz Graph

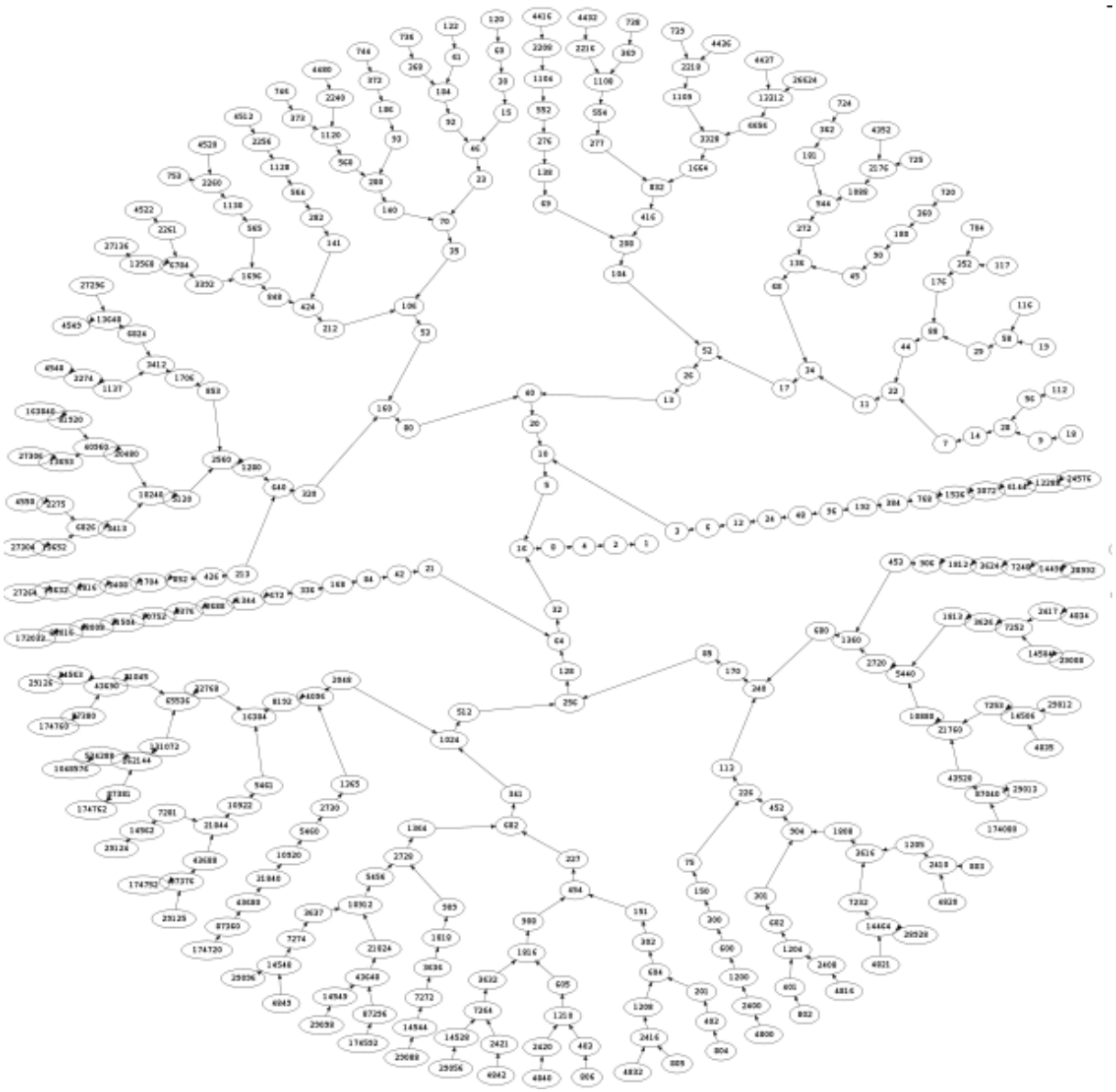
The Collatz Conjecture

1. Take a positive integer *n*.
2. If *n* is even, halve it; if odd, multiply by 3 and add 1.
3. Go to step 2.

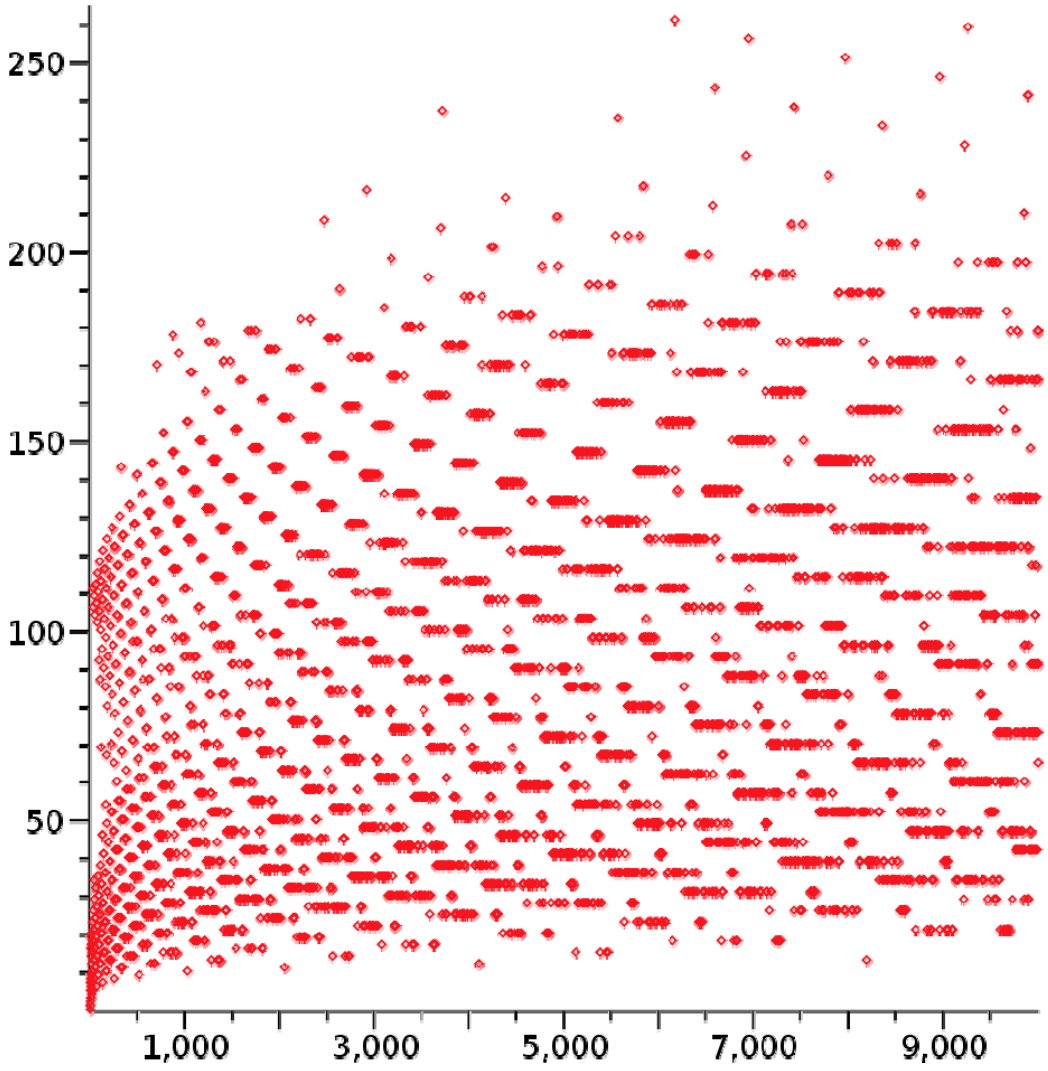
The conjecture is that this procedure always ends up with *n* in the loop 1-4-2-1-…



The Collatz Graph



The Collatz Graph



The Collatz Graph

Miscellaneous facts about the Collatz Graph.

- From 27 to 1 takes 111 steps, going via 9232.
- Distributed computing projects like 3x+1@home have so far checked every n up to 20×2^{58} (around 6×10^{18}).
- Apart from 1-4-2-1, there are no other loops with fewer than 338,466,909 elements.

There are some mathematical results, such as the one involving loops, but no great progress on proof or refutation yet

"Mathematics is not yet ready for such problems"
Paul Erdős

Data URI Scheme

Under the original http: protocol web pages with many small images, icons, bullets etc. require several round-trip messages to the server, each carrying small amounts of data.

This can be slow, and is a particularly bad fit with the next protocol down, TCP, which works best on long-lived connections.

Various additions to the protocols were proposed:

- Persistent connections in HTTP 1.1
- Caching of files, in various places
- The data: URI scheme

The first two are now universally employed to fix the problem. The third, less so. (Failure of Internet Explorers 1–7 to implement data: URI didn't help.)

Data URI Scheme

```
data:[<MIME-type>][;base64],<data>
```

```
<img src="
AABAAAAAQCAMAAAAoLQ9TAAAAUVBMVEWAWYCAWYCAWYBmZplmZplmZp
lmZplmZplmZplmZplmZplmZplmZpnMMzPMMzPMMzPMMzPMMzPMM
zPMMzPMMzPMMzPMMzPMMzPMMzNmZpndV6eGAAAAGXRSTlMADhQA8WsG
c/huMMVsQACWOkB3gElOk/r7ebcYNwAAAFRJREFUGJVj4AMDfklJfgi
LAUwKSAKBIJKAEEhAGElABCQgiiTAJyYuATMDChgZGZggLGYYYGEFUz
ABNnYOTiQBLm4WKSkeJAFeKSksApwEVZAtAAAueAp68HV63AAAAABJR
U5ErkJggg==">
```

Data URI Scheme

```
data:[<MIME-type>][;base64],<data>
```

```
<a href="data:text/plain,A%20brief%20note">Footnote</a>

<a href="data:text/html,<p>Hello World!</p>">Go here</a>

<a href="data:text/html,<a href=%22data:...>">Quine?</a>
```

Back to the Weightless Website

Combining all these, we obtain a self-supporting website of unbounded extent, contained within a single URL.

- It's a quine, starting with a data: URL…
- …which codes for a Javascript program…
- …which writes an HTML page…
- …which contains modified versions of the same URL.

These modifications give a page for every number reachable by working backwards from 1 in the Collatz graph.

This may include all positive integers.

A modest URL

```
data:text/html,<script>e=encodeURI(d='<title>Weightless_Website_Page_Q</ti
tle><h1>Weightless_website_page_Q</h1>Explore_the_Collatz_graph:_from_page
_<em>n</em>_you_can_go_to_page_2<em>n</em>,_and_to_page_(2<em>n</em>-1)/3_
if_that_is_a_whole_number.<p>V<p>This_is_page_Q._From_here_you_can_go@Z</a
>~There_is_a_pirate_on_page_42.~<img_src="
oAAAANSUhEUgAAAEAAAABgAgMAAADQq5x7AAAADFBMVEUAAQD/AAD/upn+//xbpuzVAAACDUlE
QVQ4y72UTWorMQzHVZtCMQWvup9lmTMU3CN08XIfk+xyCtPVoFOY5AC9ig8w4CfJHzPjtqsHz4
sk/kWW/vqwIUNZNpcFawXgRlBNIMGR7ADon0HcAAiAfwPuP4OXPz1f/n48vcMevJzoA30HD6cP
MIhLB2QwIy2bkwAyULxfaPclFmQQllYvBh9qlzzX9B0GIBv1pC4bQAbzpXduFoANZEAG5tzAnV
R4UMr7FlaAUT78BkwB5w7mTwaz72CaC8AGXqkWATDM/QgXx+zBFaUcpoMz+hmvqI7gHFRL33yi
MhTnqQ4qCzUE1DJVQC0wUnUC3AdICslomTC4xGWEfEOqiJ+Mz9FmssqRogSYqA1l6AgEduT1qi
ugmnLvrHSygNLK5CowBbhYwQ0EBBdrG25ez1DUBgK7vF42sPAZpTeA3tBA2R1YFGlzuUUhD1Rj
kl11ULYYjIe3XJVGVUYScs0FIouAUorIILEuGcLI4w+wsi4Om1MBTga53kjLILFSLyGSLVBhHd
NVJwZrvyxrfRMS2Cgmax3pqPmnZpDl6nFbIx8Sp2BFM88858cgNX9OpIOueRO4l2taS0W29x6n
g9ivOP+l7+v2XgmIxeCZTZK4rwZRF9AeNU2K83p4ruIIYHhaHIfZP4pvtmbX1vMIQA9Aw+BDR3
uMIjMLo7JvIB6AHoEddLhvYDzitih/AQ5xnQ8oxvDCAAAAAElFTkSuQmCC">~_or@Z</a>.~_t
o_<a_href="data:text/html,<script>e=encodeURI(d=!27X!27);String.prototype.
r=d.replace;s=d.r(/!21/g,!27%!27).r(/!5F/g,!27_!27).split(!27!7E!27);docum
ent.write(s[0].r(/Q/g,K).r(/Z/,2*K).r(/V/,s[K==42?2:1]).r(/@/,s[4].r(/!4B/
g,2*K).r(/X/,e))+(K!253==2?(s[3].r(/Z/,(2*K-1)/3).r(/@/,s[4].r(/!4B/g,(2*K
-1)/3).r(/X/,e))):!27.!27));!3C/script>">page_');String.prototype.r=d.repl
ace;s=d.r(/!/g,'%').r(/_/g,'%20').split('~');document.write(s[0].r(/Q/g,1)
.r(/Z/,2).r(/V/,s[1]).r(/@/,s[4].r(/K/g,2).r(/X/,e))+'.');</script>
```

A much more remarkable URL

```
data:text/html,<body><script>for(B=i=y=u=b=i=5-5,x=10,z=15,I=[],l=[];l[B]="
ustvrtsuqqqqqqqqyyyyyyyyy}{|~z|{}@G@TSb~?A6J57IKJT576,+-48HLSUmgukgg%20OJNML
K%20%20IDHGFE".charCodeAt(B)-64,B++<120;I[B-1]=B%x?B/x%x<2|B%x<2?7:B/x&4?0:
l[u++]:7);function%20X(c,h,e,S,s){c^=8;for(var%20T,o,L,E,D,O=20,G,N=-1e8,n,
g,d=S&&X(c,0)>1e4,C,R,A,K=78-h<<9,a=c?x:-x;++O<99;)if((o=I[T=O])&&(G=o&z^c)
<7){A=G--&2?8:4;C=9-o&z?l[61+G]:49;do{R=I[T+=l[C]];g=D=G|T+a-e?0:e;if(!R&&(
G||A<3||g)||(1+R&z^c)>9&&G|A>2){if(!(2-R&7))return%20K;for(E=n=G|I[T-a]-7?o
&z:6^c;E;E=!E&&!d&&!(g=T,D=T<O?g-3:g+2,I[D]<z|I[D+O-T]|I[T+=T-O])){L=(R&&l[
R&7|32]*2-h-G)+(G?0:n-o&z?110:(D&&14)+(A<2)+1);if(S>h||1<S&S==h&&L>2|d){I[T
]=n,I[g]=I[D],I[O]=D?I[D]=0:0;L-=X(c,h+1,E=G|A>1?0:T,S,L-N);if(!(h||S-1|B-O
|T-b|L<-1e4))return%20W(y=E),c&&setTimeout("X(8,0,y,2),X(8,0,y,1)",75);E=1-
G|A<7|D|!S|R|o<z||X(c,0)>1e4;I[O]=o;I[T]=R;I[D]=I[g];D?I[g]=G?0:9^c:0}if(L>
N||!h&L==N&&Math.random()<.5)if(N=L,S>1)if(h?s-L<0:(B=O,b=T,0))return%20N}}
}while(!R&G>2||(T=O,G|A>2|z<o&!R&&++C*--A))}return-K+768<N|d&&N}function%20
W(){i="<table>";for(u=18;u<98;document.body.innerHTML=i+=++u%x-9?"<th%20wid
th=60%20height=60%20onclick='I[b="+u+"]&8?W():X(0,0,y,1)'style='font-size:5
0px'bgcolor=#"+(u-B?u*.9&1||9:"d")+"0f0e0>&#"+(I[u]&15?9808+l[67+(I[u]&15)]
:160)+";":u++&&"<tr>")B=b}W()</script></body>
```

Based on the 1k Javascript Chess program of Óscar Toledo Gutiérrez