how hard can it be?

- algorithms:
  - what can they do?
  - what can't they do?

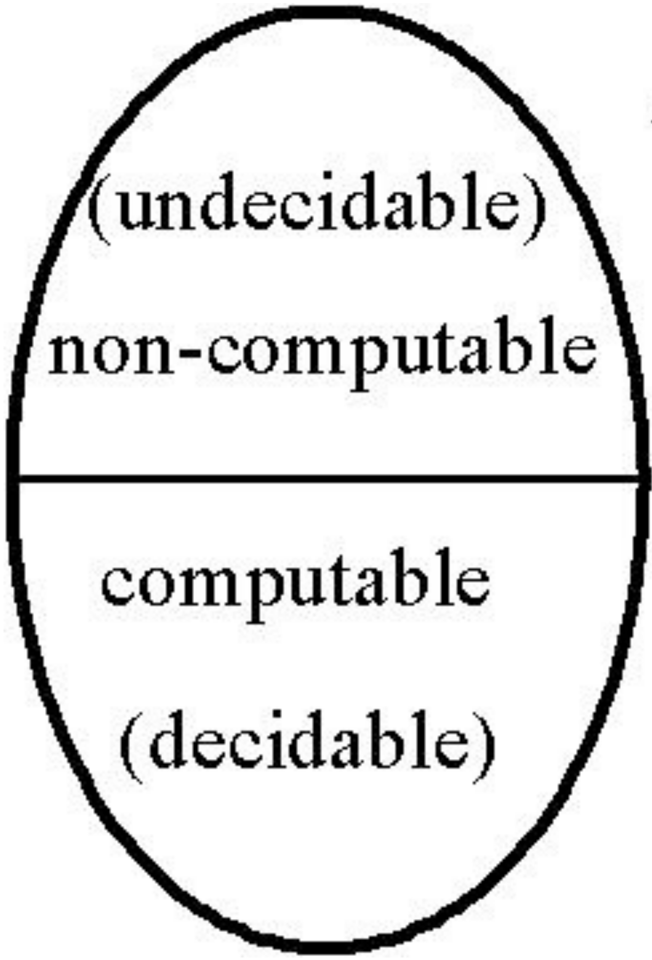
in three easy parts

- giving instructions
  - sequence, parameters
  - conditionals, iteration, recursion
- needles in haystacks
  - hard to find; easy to recognise
- some things are impossible

Alice laughed: "There's no use trying," she said; "one can't believe impossible things."

"I daresay you haven't had much practice," said the Queen. "When I was younger, I always did it for half an hour a day. Why, sometimes I've believed as many as six impossible things before breakfast."

Alice in Wonderland.



Algorithmic problems

- is 3 prime ?
- is 3719 prime ?
- is 1024 prime ?
- is n prime ?
- does $3719^2 = 13{,}830{,}961$?
- does $x \times y = z$?

PRIME NUMBERS
BETWEEN 1 AND 1,000

2	79	191	311	439	577	709	857
3	83	193	313	443	587	719	859
5	89	197	317	449	593	727	863
7	97	199	331	457	599	733	877
11	101	211	337	461	601	739	881
13	103	223	347	463	607	743	883
17	107	227	349	467	613	751	887
19	109	229	353	479	617	757	907
23	113	233	359	487	619	761	911
29	127	239	367	491	631	769	919
31	131	241	373	499	641	773	929
37	137	251	379	503	643	787	937
41	139	257	383	509	647	797	941
43	149	263	389	521	653	809	947
47	151	269	397	523	659	811	953
53	157	271	401	541	661	821	967
59	163	277	409	547	673	823	971
61	167	281	419	557	677	827	977
67	173	283	421	563	683	829	983
71	179	293	431	569	691	839	991
73	181	307	433	571	701	853	997

computable questions

Alan Turing (1912 – 1954)

Alonzo Church (1903 – 1995)

- program A - a sequence of instructions
- input D - any text string of data
- A (D) - result of program A with input D
  - result: true T or false F
  - or the program never stops

computable questions

we say "N is prime?" is **computable** because there is a program A such that

- A(N) = T if N is a prime number
- A(N) = F if N is not a prime number

a True/False question is **computable** if there is a program that computes the answer and halts for every input

computable questions

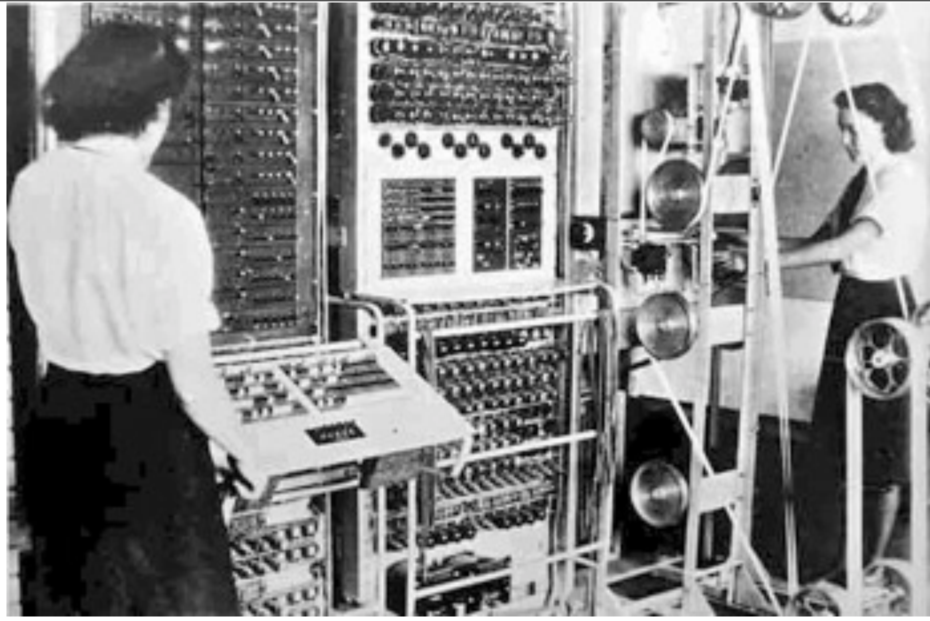
a program A is just a string of text, so can be used as data

is this question computable:

- "Does program A halt with input D?"

if it is computable, there is a program P such that

- P (A, D) = T if A halts with input D
- P (A, D) = F if A does not halt with input D



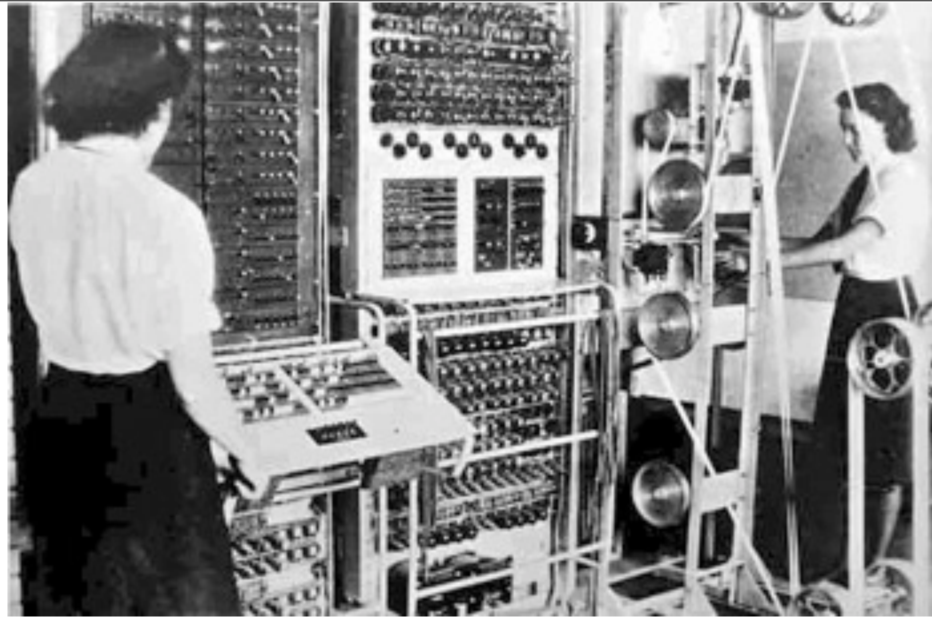
(not) halting

- P (A, D) = Good if A halts given input D
- P (A, D) = Bad if A does not halt on input D

let Q be a new program.

Q takes an input A - then runs P (A,A)

- if P (A,A) = Bad then Q (A) outputs Bad
- if P (A,A) = Good then Q (A) loops



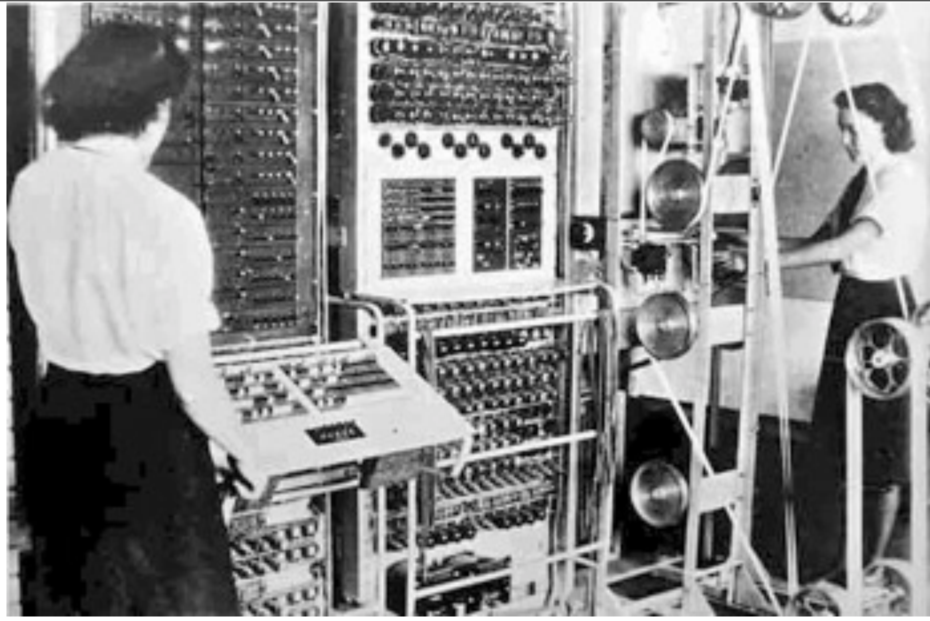
(not) halting

- P (A, A) = Good if A halts given input A
- P (A, A) = Bad if A does not halt given input A

let Q be a new program.

Q takes an input A - then runs P (A, A)

- if P (A, A) = Bad then Q (A) halts
- if P (A, A) = Good then Q (A) loops



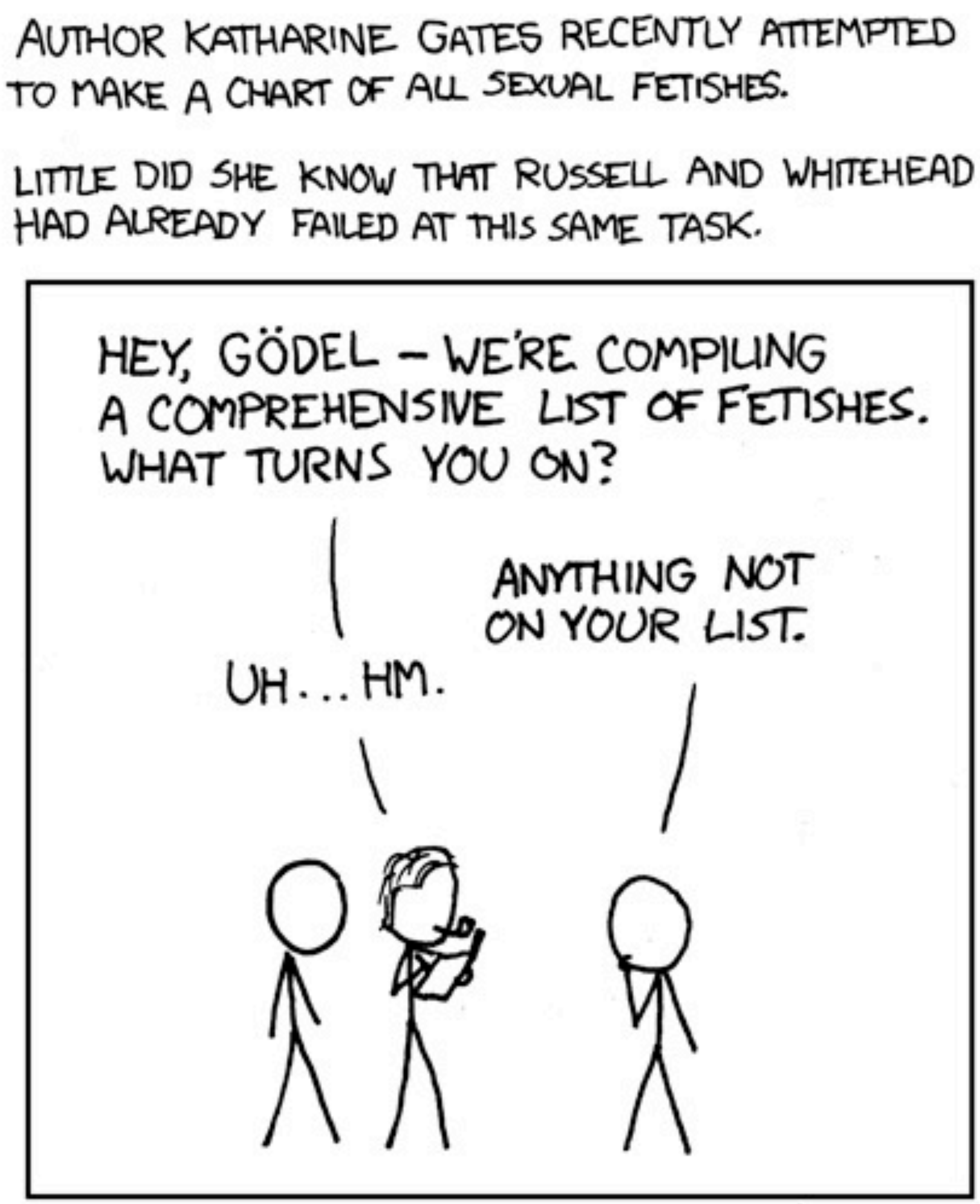
(not) halting

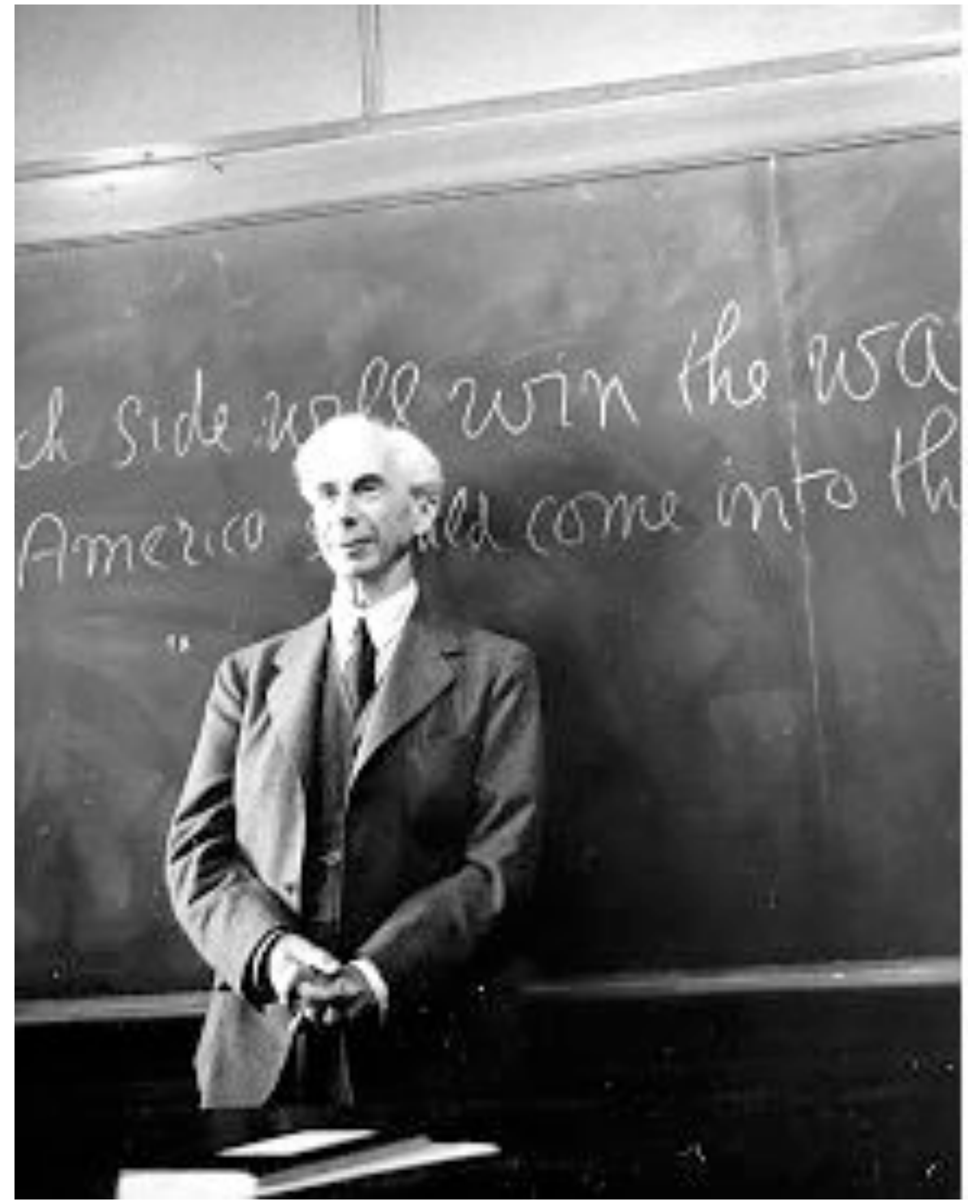
- P (A, A) = Good if A halts given input A
- P (A, A) = Bad if A does not halt given input A

what's the looping behaviour of Q run on Q?

Q run on Q - first runs P (Q, Q)

- if Bad then Q (Q) halts - should be Good
- if Good Q (Q) loops - oh dear! what to do?





Bertrand Russell (1872 – 1970)

provability

- Every provable statement is true.
- Is every true statement provable in arithmetic?
- Kurt Gödel constructed a statement which is true if, and only if, it is not provable in arithmetic. Intuitively it ‘says’
- This statement is not provable in arithmetic - but it is true.