

Last week

- computability: what cannot be computed at all
- complexity: what cannot be computed efficiently
- universal model of computation: Turing machine
- enumerable sets: sets whose members can be listed
- diagonalisation: not all sets are enumerable
- uncomputability: mechanical computation is limited

the halting problem

- proof (by contradiction) that the halting function cannot be computed:
- let's assume we can: there is a halting function H .
- $H(\text{program}, \text{data})$ true if program halts on data, else false.
- Now let's make a function Q :
- $Q(x) = \text{if } H(x, x) \text{ then loop; else stop;}$

- Now if we run Q on itself, there is a contradiction:
- $Q(Q,Q) = \text{if } H(Q,Q) \text{ then loop; else stop; but } H(Q,Q) \text{ means } Q \text{ halts on } Q \text{ (which it doesn't). And not } H(Q,Q) \text{ means } Q \text{ doesn't halt on } Q \text{ (which it does)}$
- \rightarrow contradiction, hence hypothesis must be wrong: we cannot construct the halting function H .

Complexity

algorithm quality difficulty grows with size of problem

time complexity steps needed to solve instance of problem (in order of the problem size)

- constant (hashtable)
- polynomial (n^2, n^3, n^c) (sorting)
- exponential ($2^n, 3^n, c^n$)

space complexity space required based on the size of the input.

Does it matter?

- problem has not been “well-solved” until polynomial time algorithm is found
- exponential time algorithms are not “good”
- polynomial time algorithms are “good”
- (keep in mind: this is a worst-case approximation).

Time complexity function	Size n					
	10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second
n^4	.1 second	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	2×10^4 centuries	1.3×10^{13} centuries

Figure 1.2 Comparison of several polynomial and exponential time complexity functions.

P complexity

- P: class of (decision) problems that can be answered in polynomial time (in terms of the input) by a Turing machine.
- comparison based sorting: bubblesort or mergesort.

Other problems?

- Travelling salesman problem:

A salesman has to visit n cities, and wants to minimise his travelling distances.

- Checking solution is easy: traverse found path, check if each city visited once, sum the weights encountered, compare to k .

- Generating solution is hard:

greedy algorithm fails: cannot use local knowledge.

if you try all possible paths: $n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$.

NP complexity

- NP: class of problems that can be
 - solved in polyn. time by a nondeterministic TM.
 - verified in polynomial time by a Turing machine.
- NP complete: a NP problem to which every NP problem can be reduced.
- 3SAT, first NP-complete problem (1971): *is there an assignment that makes the expression true?*

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

Hierarchy

- $P \stackrel{?}{=} NP$
- all known algorithms for NP problems exponential.
- have to attack NP problems using heuristics.