A brief discussion of complexity and P vs. NP vs. NP-complete

(not in our book; see Wikipedia page on P versus NP)

As our graphs $G=(V,E)$ get bigger, say with $|V|,|E|$ growing large, most of the YES/NO questions (called decision problems) we have asked about G take longer to answer, e.g.

- Planarity: Is G planar?
- connectivity: Is G connected?
- Eulerian-ness: Does G have an Euler circuit?
- Hamiltonian-ness: Does G have an Hamilton circuit?
- Graph isomorphism: Do $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$ have $G_1 \cong G_2$?
- Degree sequence-ness: Is $\underline{d}=(d_1,\ldots,d_n)=\underline{d}(G)$ for some simple graph G?

As large-scale computation become feasible in the mid 1900's, people started thinking carefully about how much longer it would take, called computational complexity, for various algorithms to answer these and other decision problems

(Non graph theory) EXAMPLES:

- Primality: Is an integer n prime?

- Boolean satisfiability:
  Does some given Boolean formula, like
  $f(x_1, x_2, x_3, x_4, x_5, x_6) =$

  $$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_4) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_5} \vee x_6)$$

  OR:

x	y	$x \vee y$
T	T	T
T	F	T
F	T	T
F	F	F

  AND:

x	y	$x \wedge y$
T	T	T
T	F	F
F	T	F
F	F	F

  NOT:

x	$\overline{x}$
T	F
F	T

  have any assignments of $(x_1, \dots, x_n) \in \{T, F\}^n$ that satisfy it, i.e. that make $f(x_1, \dots, x_n) = T$?

- Halting problem: Given a computer program that takes certain inputs, and a particular instance of that input, decide whether the program will
  run forever versus
  terminating after finitely many steps.

Some complexity HISTORY:

1936: Church shows the Halting problem is not even decidable – there is no algorithm to decide it!

1950s: People like Nash, Von Neumann start formalizing the notion of input size N for various problems,

e.g. $N = |V|$ or $|E|$ or $\max\{|V|, |E|\}$
for various graph decision problems,

$N = \log_2(n)$ = number of (binary) digits
for number-theoretic problems
like primality of n

N = length of Boolean expression $f(x_1, \ldots, x_n)$
for satisfiability

And then they start asking about complexity bounds for # of steps to decide them,

e.g. exponential: $\leq C \cdot b^N$ steps for some constant C, base b

P = polynomial: $\leq p(N) = a_0 + a_1 N + a_2 N^2 + \ldots + a_d N^d$ steps.

Polynomial bounds generally mean the algorithms remain feasible for big N, as computing speeds improve.

EXAMPLES in complexity class P:

- Connectivity: if $G=(V,E)$, one can check connectivity in $\leq C\cdot N^2$ steps where $N=|V|$, via breadth-first search.

- Eulerian-ness: again letting $N=|V|$, can check connectivity in $\leq cN^2$ steps and then check $\deg_G(x)$ even $\forall x\in V$ in $\leq N$ steps. (a total of $\leq cN^2+N$ steps).

- Planarity: somewhat surprisingly, the Hopcroft-Tarjan Algorithm (1974) can decide this in linear (!) time, i.e. $\leq cN$ steps, where $N=|V|$.

- Degree-sequence-ness: any of the criteria by Havel-Hakimi or Erdös-Gallai or Ruch-Gutman give rise to algorithms that take $\leq c\cdot N^2$ steps to check if $\underline{d}=(d_1\geq d_2\geq\ldots\geq d_N)=\underline{d}(G)$ for a simple graph G.

- Primality: Very surprisingly, the Agrawal-Kayal-Saxena Test (2002) decides primality of n in $\leq c\cdot N^8$ steps, where $N:=\log_2(n)$

1971: Cook
1973: Levin $\Big\}$ introduce the complexity class

NP := $\left\{\begin{array}{l}\text{decision problems where an algorithm} \\ \text{exists to check purported certificates} \\ \text{for a YES answer in} \le p(N) \text{ steps,} \\ \quad \text{for some polynomial } p(N)\end{array}\right\}$.

"non-deterministic polynomial time"

One has $P \subseteq NP$, because any problem in P has an algorithm that produces a valid certificate and checks it in $\le p(N)$ steps for a YES, or similarly produces and checks a certificate for NO.

But NP also contains...

- Hamiltonicity: given $G=(V,E)$, letting $N=|V|$, if someone purports that a given ordering $(x_1, x_2, \ldots, x_N)$ of V gives a Hamilton cycle, you can verify its correctness in N steps.
- Graph isomorphism: given $G_1=(V_1,E_1)$, $G_2=(V_2,E_2)$ can check if $f_1: V_1 \to V_2$, $f_2: E_1 \to E_2$ give an isomorphism, in $\le N$ steps, where $N=|V_1|=|V_2|$.
- Boolean Satisfiability: given a Boolean formula $f(x_1, \ldots, x_n)$ of length N, if someone purports that a particular $(x_1, x_2, \ldots, x_n) \in \{T,F\}^n$ makes $f(x_1, \ldots, x_n) = T$, you can verify that in N steps.

More importantly, they showed...

Cook-Levin Theorem: (1971) (1973) Boolean satisfiability is actually

NP-complete: if one had a polynomial-time algorithm to solve it, one could convert that to an algorithm to solve any other problem in NP in polynomial time!

(Hence if Boolean satisfiability lies in P, then all of NP lies in P, so P = NP!)

Cook even showed NP-completeness for the special case called 3-satisfiability, where

$$f(x_1, \dots, x_n) = F_1 \wedge F_2 \wedge \dots \wedge F_N$$

and each F_i looks like $x_a \vee x_b \vee x_c$

or $\bar{x}_a \vee x_b \vee x_c$

or $\bar{x}_a \vee \bar{x}_b \vee x_c$

or $\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c$

Karp's List of 21 NP-complete problems (1972):

Showed many important problems in graph theory and elsewhere are also NP-complete, e.g. Hamiltonicity.

$1,000,000 Clay Math Prize QUESTION:
Is P = NP?

When surveyed, many computer scientists think NO, i.e. the known NP-complete problems will probably never have polynomial-time algorithms, so they are inherently harder than those in P.

Surprisingly, Babai's Theorem (2015) showed Graph Isomorphism is quasi-polynomial, i.e. it can be decided in $\leq b^{(\log_2 N)^c}$ steps for some base b, and constant c.