Basic Algorithms - Sec. 19 (Extra)

P vs. NP

P → polynomial time solvable

NP → non-deterministic poly time.

E.g., 3 SAT

TSP

Hamiltonian Path

Knapsack

Change making

Multiple gene alignment

Sudoku / Sokoban

Chess

... not solvable in poly time (yet).
Efficient...
Polytime $\rightarrow$ good
everything else $\rightarrow$ bad

(Problems: $n^{50}$ is nice
Las Vegas/Monte Carlo $\rightarrow$ not nice.)

what is a problem in $\Omega$?
types of problems.
Decision vs Search
  e.g. is graph cyclic?
  e.g. find cycle in graph.

Theory of computing--
define encodings of problems as a bit string.
e.g. a graph can be written as

\[
\begin{array}{c}
4 \leftrightarrow 3 \\
2 \leftrightarrow 1
\end{array}
\]

\[
\begin{array}{cccccc}
5 & 5 \\
1 & 2 & 4 \\
3 & 4 & 35 \\
45 & & & & &
\end{array}
\]

$\rightarrow$

5 5 1 2 2 4 3 4 3 5 5

$\rightarrow$

bit string
101 101 001 010 ...
so we have a set $h = \{ \text{encoding}(p) \mid p \text{ a problem} \}$.

algorithm $A : h \rightarrow \{0, 1\}$

$A(p) = 1$ if YES
$0$ if NO

$h_1 = \{ p \in h \mid A(p) = 1 \}$.

These sets are called "languages".

decision $(\leq)$ recognizing $h_1$ as a language

---

Search $\iff$ verification.

An algorithm which gives back an answer needs to be verifiable....

e.g. in the cycle, we can check if all edges exist in $G$.

def $h \rightarrow$ encodings of all problems.

$S \rightarrow$ encodings of all outputs (correct or not).

Verifier $V : h \times S \rightarrow \{0, 1\}$.

$V(p, s) = 1$ if $s$ is a solution to $p$

$0$ $\text{otherwise}$.
Connections betw
search & decision

\[ \exists s \in V, (p,s) = 1 \implies A(p) = 1 \]

↓ existential

intuitively decisions are easier than searching

\[ s : \text{solution/witness/certificate} \]

to help the verifier.

---

P : all decision problems solvable in polytime

NP : all decision problems verifiable in polytime

with a certificate

---

How is it same as non-deterministic?

Randomly generate a certificate.

if \( A(p) = 1 \implies \text{non-zero probability of generating correct certificate} \)

Open: \( P \overset{?}{=} NP \).

if we can verify fast, can we also solve it fast?
Why are problems hard?

when → more interesting.

E.g. 3-SAT

a formula

\( (x_1 \lor x_2 \lor x_3) \land (x_4 \lor x_5 \lor x_6) \land \ldots \)

Here we want to give assignments to variables to satisfy them.

A random assignment \( \sigma \): \( X \rightarrow \{0, 1\} \)
satisfies a clause with probability = \( \frac{7}{8} \)

Fact: No polynomial time algorithm is known to solve 3-SAT.

But: most 3-SAT formulas are easy to solve.

Intuition: the less number of formulas/variables ratio, the more likely to satisfy random assignments by the larger the ratio \( \Rightarrow \) more overlap between formulas \( \Rightarrow \) harder to satisfy everyone.
Example: consider sparse cases.
i.e. \# clauses = \alpha \cdot \# variables
\[ m = \alpha \cdot n \]
\[ \Sigma \geq \# assignments which satisfy the formula. \]

we want to find \( \alpha \) such that
with high probability the clause is unsatisfiable.
i.e. we want to show that \( \Pr (Z > 0) \) is low.

Markov
\[ \Pr (Z > 1) \leq \frac{E[Z]}{1} \]
\[ \Rightarrow \Pr (Z > 0) \leq E[Z] \]

\[ E[Z] = \sum_{\text{assignment}} E[1_{0}] \]

\( 1_{0} \) := indicator of whether \( \sigma \) satisfies the formula.

\[ E[1_{0}] = \Pr (\sigma \text{ satisfying}) \]
\[ = \frac{m}{n} \Pr (\sigma \text{ satisfies clause } i) = \prod_{i=1}^{m} 7/8 \]
\[ = (7/8)^m \]
\[ E[z] = \sum_0^\infty (7/8)^m \]

\[ = 2^m \cdot (7/8)^m \]

\[ = \left[ 2 \cdot (7/8)^{\alpha} \right]^m \]

Now \( 2 \cdot (7/8)^{\alpha} \leq 1 \Rightarrow \Pr(Z > u) \to 0 \) as \( m \to \infty \)

\[ \Rightarrow \text{take } \alpha = \log_{7/8} 2 \approx 5.2 \]

So, if we have more than \( \alpha = 5.2 \) clauses to variables ratio, we probably won't satisfy the formula.

---

This bound is not tight.

We can refine it more by using Chebyshev's inequality (not for today).

Also still need to prove other way...
A lot of NP-problems in graphs become easier if you know the density of the graph.

E.g. independent set: low size if lots of edges.

Clique (complete subgraph): low size if less # edges.

etc.

How to give a probability distribution on graphs?

→ choose each graph with \( \frac{1}{2^\binom{n}{2}} \) probability.

Possible & horrible...

→ Very simple model by Erdős Rényi.

→ Generative model.

"Create" a graph with a probability \( p \) on each edge → an edge is included with probability \( p \).
E-R graphs have a lot of “thresholding” properties, \( g(n, p) \to \text{random E-R graph on } n \) edges.

E.g. 1) if \( p > \frac{d \log n}{n} \), \( d \geq 1 \)
then \( \supset \) graph is connected
\[ p < \frac{d \log n}{n} \quad \lambda < 1 \]
--- --- disconnected

2) \( np < 1 \) then \( \supset \) the largest connected component is \( O(\log n) \)
   (similar to depth of quicksort / quick select)

3) \( np = 1 \) --- of size \( n^{2/3} \)
   (edge case in previous proof).

etc. for giant components, eigenvalues, society detection.

Caveat: This is not the most ideal modeling of the real world, but it is very simple & enlightening.
Thus \( \text{let } p = \frac{d \log u}{n} \)

1) if \( d \gg 1+\varepsilon \) \( \Rightarrow \) \( G(n, p) \) is connected w.h.p.
2) if \( d \leq 1-\varepsilon \) \( \Rightarrow \) \( G(n, p) \) is disconnected w.h.p.

**Proof of 2.**

Let \( I_i \) \( \rightarrow \) indicator for node \( i \) being isolated

\[
I = \sum_{i=1}^{n} I_i \rightarrow \text{# of isolated nodes.}
\]

\[
\Pr(I_i = 1) = (1-p)^{n-1} \approx e^{-pn} = e^{-d \log u} - d
\]

\[
\Rightarrow \mathbb{E}[I] = \sum \mathbb{E}[I_i]
\]

\[
= u \cdot u^{-d} = u^{1-d} \rightarrow \infty \quad \text{as } n \rightarrow \infty \text{ because } d < 1
\]

But this is not enough

need \( \Pr(I = 0) \rightarrow 0 \text{ as } n \rightarrow \infty \)

\[
\Rightarrow \Pr(I = 0) \leq \Pr\left(\left|I - \mathbb{E}[I]\right| \geq \mathbb{E}[I]\right)
\]

\[
\leq \frac{\text{Var}(I)}{\mathbb{E}[I]^2} \leq \frac{\mathbb{E}[I]}{\mathbb{E}[I]^2}
\]

\[
\rightarrow 0 \quad \text{as } n \rightarrow \infty
\]
with high probability graph is disconnected

Proof of 1)

\[ E[I] = n^{1-\lambda} \to 0 \quad \text{as} \quad n \to \infty \quad \text{if} \quad \lambda > 1 \]

\( \Rightarrow \) graph is connected (even though there are no isolated nodes, we could have a forest).

-----

if disconnected then we have a set of \( k \) nodes

\( \Rightarrow \) they are not connected to the rest of \( n-k \) nodes.

let \( D_k \) be the event that there are \( k \) such nodes

\[ \Pr(D_k = 1) = \frac{\binom{n}{k}}{\binom{n}{k}} (1-p)^k \]

\( \Rightarrow \) \( D = \sum D_k \) := event that graph is disconnected

\( \Rightarrow \) \( \Pr(D > 0) \leq E[D] \)

\[ = \sum_{k=1}^{n^2} \frac{\binom{n}{k}}{\binom{n}{k}} (1-p)^k (n-k) \]

\[ \to 0 \quad \text{as} \quad n \to \infty \quad \text{(proof is unimportant)} \]

"math"...
Importance of phase transitions:

- Designing AI systems:
  - When do the "Heuristics" used by the system guarantee an answer?
  - Experimental evidence for finding thresholds to complement "paper work".

- A lot of models need to be updated constantly
  - How much do current models capture...

- What things don't change after certain stage?
  - E.g. if you have a lot of constraints in 3 SAT, a lot of variables will share the same value across solutions.