Lecture 14 (last, but not least)

- Bounded $A_{TM}$, revisited.
- Fixed parameter algorithms.
- Randomized (coin flipping) algorithms.
Bounded $A_{TM}$, Revisited

- Bounded $A_{TM}$: Given encoding $\langle M \rangle$ of non-deterministic TM, an input $w$, time bound $1^k$ in unary, does $M$ have an accepting computation of $w$ in $k$ steps or less?

- Bounded $A_{TM}$ is NP complete, via a “generic” reduction.

- Finding reduction was wasy.

- Proving Bounded $A_{TM}$ is in NP seemed less obvious.
Bounded Atm in NP

An instance of the problem has the form \( \langle M, w, 1^k \rangle \). The universal NTM, \( U \), decides this language:

- \( U \) does not write anything on input tape.
- Copies \( q_0 w \) to second tape.
- Simulates \( M \) step by step, keeping its configuration on second tape.
- Sets up a unary **counter** on third tape, initialize to zero and increments it for each simulated step of \( M \).
- If counter reaches \( k + 1 \), \( U \) rejects.
- How many steps of \( U \) does it take to simulate \( k \) steps of \( M \)?
Bounded Atm in NP

- How many steps $U$ takes to simulate $k$ steps of $M$?
- To simulate one step of $M$, NTM $U$ has to find an entry in $M$’s transition function that matches current simulated state and letter under head.
- This requires scanning all of $M$’s transition function (on input tape), which takes length of $\langle M \rangle$ steps of $U$.
- So to simulate $k$ steps of $M$, NTM $U$ takes $k \cdot |\langle M \rangle|$ steps.
- Denote $n = |\langle M, w, 1^k \rangle|$. What is $k \cdot |\langle M \rangle|$ in terms of $n$?
- $k \cdot |\langle M \rangle| = \theta(n^2)$, so $A_{TM} \in NTIME(n^2)$.
- Question: What would the NTIME complexity be if $k$ would be encoded in binary?
Fix Parameter Algorithms

Many optimization problems have a natural parameter.

- **Vertex cover**: Given a graph with $n$ vertices and $m$ edges, find a vertex cover of size $k$ (or report that none exists).

- **Clique**: Given a graph with $n$ vertices and $m$ edges, find a clique of size $k$ (or report that none exists).

- Both problems solvable in time $n^k$
Fix Parameter Algorithms

- Both $k$-VC and $k$-clique are solvable in time $n^k$.
- Is this the best we can do?
- The class fix parameter tractable – FPT (Downey and Fellows, 1992) contains all parameterized optimization problems with $f(k)n^c$ time algorithms.
Fix Parameter Algorithms

- The class **FPT** (Downey and Fellows, 1992) contains all parameterized optimization problems with $f(k)n^c$ time algorithms.

- The inherent "combinatorial explosion" of such problems is limited to the function $f(k)$.

- What is this good for? Even though $f(k)$ will be exponential (or worse), for small values of the parameter, $f(k)n^c$ may be feasible, whereas $n^k$ may be infeasible.

- Hey, this is all very nice, but isn’t this fictitious class **FPT** actually empty...
VC $\in$ FPT

VC Reminder: Given a graph $(V, E)$
Find the smallest set of vertices $C$ such that for each edge in the graph, $C$ contains at least one endpoint.

- We describe a branching algorithm
- a problem instance is repeatedly split into smaller problem instances
- leading to a tree of subproblems in which problems at the leaves supply solutions.
- our tree will be binary, of depth $k$.
- $\implies$ complexity will be $2^k n^c$. 
**VC ∈ FPT**

Branching algorithm:

- **Initialization:** $C = \emptyset$, $H = E$
- Pick an edge $(u, v) \in H$. Split the problem into two subproblems:
  - $(H_u, C \cup u)$ where $H_u$ equal to $H$ with all edges incident to $u$ removed and isolated nodes thrown away.
  - $(H_v, C \cup v)$ where $H_v$ equal to $H$ with all edges incident to $v$ removed and isolated nodes thrown away.
- Stop when $C$ is of size $k$. Check if it is a cover.
VC ∈ FPT

Branching algorithm correctness and time analysis:

- For every edge \((u, v) \in E\), each minimum cover must contain at least one of \(u, v\).
- This property also holds for graphs in subproblems.
- Therefore if there is a cover \(C\) of size \(k\), the algorithm finds it.
- At each node of branching tree, \(O(|E|)\) steps required.
- So overall, runtime is \(O(2^k|E|)\),

\[
\implies VC \in \text{FPT}.
\]
FPT Odds and Ends

- Best parameterized VC algorithm known to date in \( O(1.2852^k + kn) \) (Chen, Kanj and Jia, 2001).
- Enables finding covers of size up to \( k = 100 \).

- What about \( k \)-clique?
- Using \textit{parameterized reductions}, can show clique is \textit{\&*\%\$-hard} (for an appropriate class \textit{\&*\%\$}), so clique unlikely in \textit{FPT} unless \textit{\&*\%\$}=\textit{FPT}.
- For further details, see Downey & Fellows’ book, or relevant course in Wellington (NZ), Newcastle (OZ), Tubingen (GE), Victoria (CA), \ldots
Coin Fipping TMs
Randomized Computation

We can sometimes use randomization to solve problems that are difficult to solve deterministically.

Examples:
- determining if a polynomial is identically zero
- primality testing
Randomized Computation

We are given a multivariant polynomial
\[ \pi(x_1, \ldots, x_m). \]
We wish to know if \( \pi(x_1, \ldots, x_m) \) is identically zero.

One strategy: Fully expand \( \pi(x_1, \ldots, x_m) \), and check if all resulting coefficients are zero.

This strategy may not be very efficient.
For example, consider
\[ \pi(x_1, \ldots, x_m) = (x_1 - y_1)(x_2 - y_2) \cdots (x_n - y_n) \]

A second example deals with the determinant of a symbolic matrix (a matrix containing both constants and variables’ symbols).
Randomized Computation

Recall the determinant of a matrix:

\[
\text{det } \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc
\]

\[
\text{det } A = \sum_{\pi} \sigma(\pi) \prod_{i=1}^{n} a_{i,\pi(i)}
\]

Where

- \( \pi \) ranges over all permutations
- \( \sigma \) is 1 if \( \pi \) is product of even number of transpositions
- \( \sigma \) is -1 otherwise
Randomized Computation

Can compute determinants by Gaussian Elimination

\[
\begin{pmatrix}
1 & 3 & 2 & 5 \\
1 & 7 & -2 & 4 \\
-1 & -3 & -2 & 2 \\
0 & 1 & 6 & 2
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
1 & 3 & 2 & 5 \\
0 & 4 & -4 & -1 \\
0 & 0 & 0 & 7 \\
0 & 1 & 6 & 2
\end{pmatrix}
\]
Gaussian Elimination

- subtract multiples of first row from $2, \ldots, n$
- to make first column element zero
- continue with subsequent rows . . .

\[
\begin{pmatrix}
1 & 3 & 2 & 5 \\
0 & 4 & -4 & -1 \\
0 & 0 & 0 & 7 \\
0 & 1 & 6 & 2
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
1 & 3 & 2 & 5 \\
0 & 4 & -4 & -1 \\
0 & 0 & 0 & 7 \\
0 & 0 & 6 & 2\frac{1}{4}
\end{pmatrix}
\]
Randomized Computation

If pivot element is zero:

\[
\begin{pmatrix}
1 & 3 & 2 & 5 \\
0 & 4 & -4 & -1 \\
0 & 0 & 0 & 7 \\
0 & 0 & 6 & 2\frac{1}{4}
\end{pmatrix}
\]
Randomized Computation

Then

- transpose rows
- multiply determinant by $-1$
- if this is impossible, determinant is zero.

$$
\begin{pmatrix}
1 & 3 & 2 & 5 \\
0 & 4 & -4 & -1 \\
0 & 0 & 6 & 2\frac{1}{4} \\
0 & 0 & 0 & 7
\end{pmatrix}
$$
Randomized Computation

At the end, matrix has upper triangular form:

\[
\begin{pmatrix}
1 & 3 & 2 & 5 \\
0 & 4 & -4 & -1 \\
0 & 0 & 6 & 2\frac{1}{4} \\
0 & 0 & 0 & 7
\end{pmatrix}
\]

Determinant is

- product of diagonal elements: 196
- adjusted for row interchanges: \(-196\)
Randomized Computation

Claim: We can calculate the determinant of an $n \times n$ matrix in polynomial time.

Must check

- if entries are integers of $b$ bits,
- result not exponentially long in $b$

This isn’t difficult.
Randomized Computation

What about determinants of symbolic matrices?

\[
\begin{vmatrix}
  x & w & z \\
  z & x & w \\
  y & z & w
\end{vmatrix}
\implies
\begin{vmatrix}
  x & w & z \\
  0 & \frac{x^2-zw}{x} & \frac{wx-z^2}{x} \\
  0 & \frac{zx-wy}{x} & \frac{zy}{x}
\end{vmatrix}
\implies
\begin{vmatrix}
  x & w \\
  0 & \frac{x^2-zw}{x} \\
  0 & 0
\end{vmatrix}
- \frac{yz(xz-xw)+(zx-wy)(wx-x^2)}{x(x^2-xw)}
\]

Is there a problem here?

Slides modified by Benny Chor, based on original slides by Maurice Herlihy, Brown University.
Randomized Computation

We do have a problem.

Applying Gaussian elimination to symbolic matrices works, but

- intermediate terms are rational functions
- it can be shown that their size become exponentially large!

Oh, oh.
Randomization to the Rescue

Schwartz’ Lemma

Let $\pi(x_1, \ldots, x_m)$ be a polynomial

- not identically zero
- in $m$ variables
- of degree at most $d$ in each variable
- and let $M > 0$ be an integer.

Then

- the number of $m$-tuples $(x_1, \ldots, x_m) \in \{0, 1, \ldots, M\}$
- satisfying $\pi(x_1, \ldots, x_m) = 0$
- is at most $mdM^{m-1}$
Randomization to the Rescue

Using Schwartz’ Lemma, choose $M > 0$
large enough so that $m d M^{m-1} / M^m = m d / M < \varepsilon$.

Pick $(x_1, \ldots, x_m) \in \{0, 1, \ldots, M\}$ at random.

Then, if $\pi$ is not identically zero,
$Pr(\pi(x_1, \ldots, x_m) = 0) < \varepsilon$
Bipartite Matching

Surprisingly enough, this approach can be applied to find perfect matching in bipartite graphs.
Randomized Computation: Terminology

A **Monte Carlo** algorithm.
- runs efficiently
- very small probability of error

A **Las Vegas** algorithm.
- no error
- very small (but positive) probability of long running time
Random Walk Algorithms

Here is a Monte Carlo algorithm for satisfiability:
M: on input $\langle \phi \rangle$ where $\phi$ is a CNF formula:

- Repeat $r$ times:
  - If all clauses satisfied, accept.
  - pick unsatisfied clause (all literals false)
  - pick literal at random
  - “flip” its value in truth assignment.
Random Walk Algorithms

Claim: If we find a satisfying assignment, the formula is satisfiable.

Claim: If we fail to find a satisfiable assignment after “enough” repetitions, the formula is “probably” not satisfiable.

Question: Is a polynomial number of repetitions sufficient to get a satisfying assignment with probability $\geq \frac{1}{2}$?
Random Walk Algorithms

Let’s try an easy case first.

**Claim:** For 2SAT, \( r = 2n^2 \) repetitions are enough to find a satisfying assignment with probability \( \geq \frac{1}{2} \).
Random Walk Algorithms

Let

- $\hat{T}$ be the satisfying assignment
- $T$ the current assignment
- the distance $i$ between $\hat{T}$ and $T$ is the number of variable assignments in which they differ ($0 \leq i \leq n$).
- let $t(i)$ be the expected time to reach $\hat{T}$ from $T$. 
Random Walk Algorithms

In any unsatisfied clause
- both literals are false
- at least one literal is true in $\hat{T}$

Random choice takes $T$
- closer to $\hat{T}$ ($i \leftarrow (i - 1)$) with probability $\geq \frac{1}{2}$
- further from $\hat{T}$ ($i \leftarrow (i + 1)$) with probability $\leq \frac{1}{2}$ (provided $i < n$).
Random Walk Algorithms

This random walk has

- one absorbing barrier \((i = 0, \text{ namely reaching } \hat{T})\)
- one reflecting barrier \((i = n)\)
Random Walk Algorithms: Analysis

\[ t(i) = \frac{t(i - 1)}{2} + \frac{t(i + 1)}{2} + 1 \]

\[ t(0) = 0 \]

\[ t(n) = t(n - 1) + 1 \]

Let’s compute \( t(i) \).
Random Walk Algorithms

\[ t(i) = \frac{t(i - 1)}{2} + \frac{t(i + 1)}{2} + 1 \]

\[ \sum_{i=1}^{n-1} t(i) = \frac{1}{2} \left( \sum_{i=1}^{n-1} t(i - 1) + \sum_{i=1}^{n-1} t(i + 1) \right) + n - 1 \]

\[ = \frac{1}{2} \left( \sum_{i=0}^{n-2} t(i) + \sum_{i=2}^{n} t(i) \right) + n - 1 \]

\[ = \sum_{i=2}^{n-2} t(i) + (t(0) + t(1) + t(n - 1) + t(n)) / 2 + n - 1 \]

\[ \Rightarrow \]

\[ t(1) + t(n - 1) = \frac{1}{2} \left( t(1) + t(n - 1) + t(n) \right) + n - 1 \]
Random Walk Algorithms

Recall that

\[ t(1) + t(n - 1) = \frac{1}{2} (t(1) + t(n - 1) + t(n)) + n - 1 \]

\[ \implies \]

\[ t(1) + t(n - 1) = t(n) + 2(n - 1) \]

Substitute \( t(n) = t(n - 1) + 1 \) to get

\[ t(1) = 2n - 1 \]
Random Walk Algorithms

From these two equations

\[ t(1) = 2n - 1 \]
\[ t(1) = \frac{t(2)}{2} + 1 \]

we have that \( t(2) = 4n - 4 \).

In general \( t(i) = 2in - i^2 \), so \( t(n) = n^2 \).

Even if the initial assignment, \( T \), is completely wrong, the expected time to find a satisfying assignment is \( n^2 \) steps.

Q.: What if formula has no satisfying assignment?
Random Walk Algorithms

What about probabilities?

**Lemma:** Let $x$ be a random non-negative integer variable with expected value $E(x)$. Markov inequality states that

$$\text{prob}[x \geq k \cdot E(x)] \leq \frac{1}{k}$$

**Corollary:** If $x$ is the number of steps until the satisfying assignment is discovered, then the probability that $2n^2$ steps will suffice to discover it is $\geq \frac{1}{2}$. 
Random Walk Algorithms

What about 3CNF? Consider

$$(x_1) \land \ldots \land (x_n),$$

it is only satisfied by the all-true assignment.

Replace each $$(x_i)$$ by $$(x_i \lor \overline{x_j} \lor \overline{x_k})$$, for all distinct $i, j, k \ ((n - 1)(n - 2) \text{ clauses}).$$
Random Walk Algorithms

Consider same random walk algorithm:

-的选择 correct literal $x_i$ with probability $\frac{1}{3}$
- chooses one of two wrong literals $\overline{x_j}, \overline{x_k}$ with probability $\frac{2}{3}$

Can show: This random walk takes exponential time to reach satisfying assignment (absorbing barrier)!
Heuristics

Main Entry heuristic
Pronunciation: hyu-’ris-tik
Function: adjective
Etymology: German heuristisch, from New Latin heuristicus, from Greek heuriskein – to discover;

involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods (heuristic techniques; a heuristic assumption);
also : of or relating to exploratory problem-solving techniques that utilize self-educating techniques (as the evaluation of feedback) to improve performance (a heuristic computer program)
Heuristics

Heuristics are widely used in almost every area with hard optimization problems. They are typically based on solid intuition, but their run-time analysis "in practice" is beyond current knowledge. **Examples:** Simulated annealing, genetic (evolutionary) algorithms, neural networks.

When all else fails, a smart heuristic may do wonders.
The Dreaded Exam

- All material covered in class and recitations, from three parts of course.
- Both multiple choice ("closed") and "open" questions.
- You can use four double sided A4 (normal size) pages.

- Piece of cake.
Famous Last Words

You have brains in your head.
You have feet in your shoes.
You can steer yourself
any direction you choose.
You’re on your own. And you know what you know.
And YOU are the guy who’ll decide where to go.

Hebrew translation by Leah Naor.