Logistics

• HW7 due tonight
• Thursday's class: REVIEW
• Final exam on Thursday Dec 8, 8am-11am, LEDDN AUD
  • Note card allowed
  • Bring photo ID, pens
  • New seating map to be posted
  • New review guide to be posted
  • No review session outside class; office hours instead.
Time complexity (a bird's-eye-view tour)

- Section 7.1: time complexity, asymptotic upper bounds.
- Section 7.2: polynomial time, $P$
- Section 7.3: $NP$, polynomial verifiers, nondeterministic machines.
Today's learning goals

- Describe how the runtime of Turing machines can be used to compare problems: which is harder?
- Compute the big-O class of the runtime of a TM from its implementation-level description.
- Distinguish between implementation-level decisions that impact the big-O class of the runtime and those that don’t.
- Define the time complexity class P and name some problems in P
- Distinguish between polynomial and exponential DTIME
- Define nondeterministic running time
- Analyse a (nondeterministic) algorithm to determine whether it is in P (respectively, NP)
- Define the class NP and name some problems in NP
- State and explain $P=NP$?
- Define NP-completeness
- Explain the connection between $P=NP$? and NP-completeness
- Describe how reductions are used both for questions of decidability and of complexity.
Time complexity

**Goal:** Which decidable questions are intrinsically **easier** (faster) or **harder** (slower) to compute?

*Algorithms that halt might take waaaaaaaaaaaaaaay too long*

*…………………………………………………..*

*e.g., too long for any reasonable applications.*
Measuring time

- For a given **algorithm** working on a given **input**, how long do we need to wait for an answer? How does the running time depend on the input in the worst-case? average-case? Expect to have to spend more time on larger inputs.

- What's in common among all problems that are **efficiently solvable**?
Measuring time

- For a given **algorithm** working on a given **input**, how long do we need to wait for an answer? **Count steps!** How does the running time depend on the input in the worst-case? average-case? **Big-O**

- What's in common among all problems that are **efficiently solvable**? **Time(n)**
Time complexity

For M a deterministic decider, its **running time** or **time complexity** is the function $f: \mathbb{N} \rightarrow \mathbb{R}^+$ given by

$$f(n) = \text{maximum number of steps } M \text{ takes before halting, over all inputs of length } n.$$
Time complexity

For $M$ a deterministic decider, its **running time** or **time complexity** is the function $f: \mathbb{N} \rightarrow \mathbb{R}^+$ given by

$$f(n) = \text{maximum number of steps } M \text{ takes before halting, over all inputs of length } n.$$ 

Instead of calculating precisely, **estimate** $f(n)$ by using big-O notation.
TM analysis

\[ M_1 = \text{"On input string } w:\text{"} \]

1. If the current tape location is blank, halt and accept.
2. Otherwise, cross off this cell’s contents and move the tape head one position to the right.
3. If the current tape location is blank, halt and reject.
4. Otherwise, cross off this cell’s contents and move the tape head one position to the right.
5. Go to step 1."
TM analysis

$M_1 = "On input string w:"

1. If the current tape location is blank, halt and accept.
2. Otherwise, cross off this cell’s contents and move the tape head one position to the right.
3. If the current tape location is blank, halt and reject.
4. Otherwise, cross off this cell’s contents and move the tape head one position to the right.
5. Go to step 1."

Is $M_1$ a decider?
A. Yes.
B. No.
C. It depends on $w$.
D. I don't know.
TM analysis

\[ M_1 = "\text{On input string } w:\) \\
1. \text{If the current tape location is blank, halt and accept.} \\
2. \text{Otherwise, cross off this cell's contents and move the tape head one position to the right.} \\
3. \text{If the current tape location is blank, halt and reject.} \\
4. \text{Otherwise, cross off this cell's contents and move the tape head one position to the right.} \\
5. \text{Go to step 1.}" \]

One step is one transition: a (possible) change in internal state, a change in current symbol on the tape, and a move for the tape head.

How many steps are executed by \( M_1 \) on 1010?

A. 1 
B. 4 
C. 5 
D. None of the above. 
E. I don't know.
TM analysis

$M_1 = \text{"On input string } w:"

1. If the current tape location is blank, halt and accept.
2. Otherwise, cross off this cell’s contents and move the tape head one position to the right.
3. If the current tape location is blank, halt and reject.
4. Otherwise, cross off this cell’s contents and move the tape head one position to the right.
5. Go to step 1.”

$L(M_1) = \{ w \text{ such that } |w| \text{ is even}\}$. $M_1$ takes $n+1$ steps to halt on input of size $n$. Running time of $M_1$ is $O(n)$.
\{0^k1^k \mid k \geq 0 \}\}

M_2 = “On input w:
1. Scan across the tape and reject if w not of the form 0*1*.
2. Repeat the following while there are both 0s and 1s on tape:
   Scan across tape, each time crossing off a single 0 and a single 1.
3. If 0’s remain after all 1’s checked off, reject.
4. If 1’s remain after all 0’s checked off, reject.
5. Otherwise, accept.”

For input w of length n, how many steps does stage 1 take?
A. O(1)
B. O(n)
C. O(n^2)
D. It depends on n in a different way from B, C
E. I don't know.
\{0^k1^k \mid k \geq 0 \}

M_2 = “On input w:
1. Scan across the tape and reject if w not of the form 0*1*.
2. Repeat the following while there are both 0s and 1s on tape:
   - Scan across tape, each time crossing off a single 0 and a single 1.
3. If 0’s remain after all 1’s checked off, reject.
4. If 1’s remain after all 0’s checked off, reject.
5. Otherwise, accept.”

How many times do we repeat stage 2 (in the worst case)?

A. \(n\)  
B. \(2n\)  
C. \(n/2\)  
D. \(n^2\)  
E. I don't know.
Big-O review

For \( f, g : \mathbb{N} \to \mathbb{R}^+ \), \( f(n) = O(g(n)) \) means

\[ \exists c \in \mathbb{N} \exists n_0 \in \mathbb{N} \forall n \geq n_0 \ ( f(n) \leq c \cdot g(n) ) \]

- **To add** big-O terms: pick maximum
- Can drop / ignore **constants**
- **Polynomials**: use highest order term
- \( \log_a b = O(\log_2 b) = O(\log b) \)
\{0^k1^k \mid k \geq 0 \}

M_2 = “On input w:
1. Scan across the tape and reject if w not of the form 0*1*.
2. Repeat the following while there are both 0s and 1s on tape:
   Scan across tape, each time crossing off a single 0 and a single 1.
3. If 0’s remain after all 1’s checked off, reject.
4. If 1’s remain after all 0’s checked off, reject.
5. Otherwise, accept.”

Running time of M_2 is \(O(n) + O(n^2) + O(n) + O(n) = O(n^2)\).

Running time of \(M_2\) is \(O(n^2)\).
Time complexity classes

\[ \text{TIME}(t(n)) = \{ L \mid L \text{ is decidable by a TM running in } O(t(n)) \} \]

- Exponential
- Polynomial: \[ P = \bigcup_{k} \text{TIME}(n^k) \]
- Logarithm
Why is it okay to group all polynomial running times?

- Contains all the "feasibly solvable" problems.
- Invariant for all the "usual" deterministic TM models
  - multitape machines (Theorem 7.8)
  - multi-write
Working with P

• Problems encoded by languages of strings
  • Need to make sure coding/decoding of objects can be done in polynomial time.

• Algorithms can be described in high-level or implementation level

**CAUTION:** not allowed to guess / make non-deterministic moves.
Graph problems

- **PATH**
  \[ \{ < G, s, t > | G \text{ a directed graph with directed path from } s \text{ to } t \} \]

- **CONNECTED**
  \[ \{ < G > | G \text{ a directed graph with single connected component} \} \]

etc.

Compute running time of graph algorithms in terms of number of nodes!
PATH

M = “On input <G,s,t> where G is digraph, s and t are nodes in G:
1. Place mark on node s
2. Repeat until no additional nodes are marked
   Scan edges of G. If edge (a,b) is found where a is marked and b is unmarked, mark b.
3. If t is marked, accept; otherwise, reject.”
PATH

M = “On input <G,s,t> where G is digraph, s and t are nodes in G:
1. Place mark on node s
2. Repeat until no additional nodes are marked
   Scan edges of G. If edge (a,b) is found where a is marked and b
   is unmarked, mark b.
3. If t is marked, accept; otherwise, reject.”

Running time of M is $O(1) + O(n \cdot n^2) + O(1) = O(n^3)$
PATH is in P.
Time complexity classes

\[ \text{TIME}(t(n)) = \{ \text{L} \mid \text{L is decidable by a TM running in } O(t(n)) \} \]

- **Exponential**
  \[ \text{EXPTIME} = \bigcup_k \text{TIME}(2^{n^k}) \]
  Brute-force search

- **Polynomial**
  \[ P = \bigcup_k \text{TIME}(n^k) \]
  Invariant under many models of TMs

- **Logarithmic**
  May not need to read all of input
Which machine model?

- **Deterministic computation**
  - From $q_0$ to $q_{\text{rej}}$

- **Non-deterministic computation**
  - From $q_0$ to $q_{\text{rej}}$, $q_{\text{acc}}$, and $q_{\text{acc}}$
Time complexity

For M a deterministic decider, its running time or time complexity is the function $f: \mathbb{N} \rightarrow \mathbb{R}^+$ given by

$$f(n) = \text{maximum number of steps } M \text{ takes before halting, over all inputs of length } n.$$

For M a nondeterministic decider, its running time or time complexity is the function $f: \mathbb{N} \rightarrow \mathbb{R}^+$ given by

$$f(n) = \text{maximum number of steps } M \text{ takes before halting on any branch of its computation, over all inputs of length } n.$$
Time complexity classes

\[ \text{DTIME} \left( t(n) \right) = \{ \text{L} | \text{L is decidable by } O( t(n) ) \text{ deterministic, single-tape TM} \} \]

\[ \text{NTIME} \left( t(n) \right) = \{ \text{L} | \text{L is decidable by } O( t(n) ) \text{ nondeterministic, single-tape TM} \} \]

Is \( \text{DTIME}(n^2) \) a subset of \( \text{DTIME}(n^3) \)?

A. Yes
B. No
C. Not enough information to decide
D. I don't know
Time complexity classes

\[ \text{DTIME} ( t(n) ) = \{ L \mid L \text{ is decidable by } O( t(n) ) \] 
\[ \quad \text{deterministic, single-tape TM} \} \]

\[ \text{NTIME} ( t(n) ) = \{ L \mid L \text{ is decidable by } O( t(n) ) \]
\[ \quad \text{nondeterministic, single-tape TM} \} \]

Is \( \text{DTIME}(n^2) \) a subset of \( \text{NTIME}(n^2) \)?

A. Yes
B. No
C. Not enough information to decide
D. I don’t know
Time complexity classes

\[
\text{DTIME}(t(n)) = \{ L \mid L \text{ is decidable by } O(t(n)) \text{ deterministic, single-tape TM} \}
\]

\[
\text{NTIME}(t(n)) = \{ L \mid L \text{ is decidable by } O(t(n)) \text{ nondeterministic, single-tape TM} \}
\]

Is \( \text{NTIME}(n^2) \) a subset of \( \text{DTIME}(n^2) \)?
A. Yes
B. No
C. Not enough information to decide
D. I don’t know
"Feasible" i.e. P
\[ P = \bigcup_{k} \text{TIME}(n^k) \]

- Can't use nondeterminism
- Can use multiple tapes

*Often need to be "more clever" than naïve / brute force approach*

**Examples**

PATH = \{<G,s,t> | G is digraph with n nodes there is path from s to t\}

RELPRIME = \{ <x,y> | x and y are relatively prime integers\}

Use Euclidean Algorithm to show in P

L(G) = \{w | w is generated by G\} \quad \text{where G is any CFG}

Use Dynamic Programming to show in P
"Verifiable" i.e. NP

- Best known solution is brute-force
- Look for some "certificate" – if had one, could check if it works quickly

\[ NP = \bigcup_{k} NTIME(n^k) \]

\[ P = NP? \]
Examples in NP for graphs

HAMPATH = \{ <G,s,t> | G is digraph with a path from s to t that goes through every node exactly once \}

CLIQUE = \{ <G,k> | G is an undirected graph with a k-clique \}

VERTEX-COVER = \{ <G,k> | G is an undirected graph with a k-node vertex cover \}

Complete subgraph with k nodes

Subset of k nodes s.t. each edge incident with one of them
Examples in NP for graphs

$\text{CLIQUE} = \{ <G,k> \mid G \text{ is an undirected graph with a } k\text{-clique} \}$

How many possible $k$-cliques are there?
How long does it take to confirm "clique-ness"?
TSP = \{ <G,k> \mid G \text{ is complete weighted undirected graph where weight between node } i \text{ and node } j \text{ is } "\text{distance}" \text{ between them; there is a tour of all cities with total distance less than } k \}\}

How many possible tours are there?
How long does it take to check the distance of a single tour?
Examples in NP for numbers

COMPOSITES = \{ x \mid x \text{ is an integer } >2 \text{ and is not prime}\}

SUBSET-SUM = \{ <S,t> \mid S=\{x_1,..,x_k\} \text{ and some subset sums to } t\}
Examples in NP for logic

SAT = \{ <\varphi> \mid \varphi \text{ is a satisfiable Boolean formula} \}

Is \( <(x \lor \bar{y}) \lor (\bar{x} \land y) > \) in SAT?
A. Yes
B. No
C. Not enough information to decide
D. I don’t know
### P vs. NP

<table>
<thead>
<tr>
<th>Problems in P</th>
<th>Problems in NP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Membership in any) CFL</td>
<td>Any problem in P</td>
</tr>
<tr>
<td>PATH</td>
<td>HAMPATH</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>CLIQUE</td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td>VERTEX-COVER</td>
</tr>
<tr>
<td>Addition, multiplication of integers</td>
<td>TSP</td>
</tr>
<tr>
<td>...</td>
<td>SAT</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Decidable

NP?

P

CF

Regular
1970s Stephen Cook and Leonid Levin independently and in parallel lay foundations of NP-completeness

Intuitively: if an NP-complete problem has a polynomial algorithm, then all NP problems are polynomial time solvable.

A language B is NP-complete if it is in NP and every A in NP is polynomial-time reducible to it.

Cook-Levin Theorem: SAT is NP-complete.
Reductions to the rescue

1970s Stephen Cook and Leonid Levin independently and in parallel lay foundations of NP-completeness.

Intuitively: if an NP-complete problem has a polynomial algorithm, then all NP problems are polynomial-time solvable.

A language B is **NP-complete** if it is in NP and every A in NP is polynomial-time reducible to it.

Cook-Levin Theorem: SAT is NP-complete.

What would prove that P = NP?

A. Showing that a problem solvable by brute-force methods has a nondeterministic solution.
B. Showing that there are two distinct NP-complete problems.
C. Finding a polynomial time solution for an NP-complete problem.
D. Proving that an NP-complete problem is not solvable in polynomial time.
E. I don't know