http://www.cs.unh.edu/~ruml/cs758

2 handouts: slides, asst 11
NP-Completeness
P vs NPC vs EXPTIME

- shortest path vs longest path
- Euler tour (each edge) vs hamiltonian cycle (each vertex)
- minimum spanning tree vs shortest total all-pairs path length spanning tree
- spanning tree vs vertex cover
- maximum flow vs minimum edge-cost flow (meeting demand)
- minimum cut vs maximum cut
- maximum bipartite matching vs minimum maximal matching
- addition vs subset sum
- 2-CNF satisfiability vs 3-CNF
- interval scheduling vs job shop scheduling
- value of move in checkers, Go
## Exponentials

If 1 step = 1 μsecond:

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>40</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>.00002 sec</td>
<td>.00004 sec</td>
<td>.00006 sec</td>
</tr>
<tr>
<td>n²</td>
<td>.0004 sec</td>
<td>.0016 sec</td>
<td>.0036 sec</td>
</tr>
<tr>
<td>n³</td>
<td>.008 sec</td>
<td>.064 sec</td>
<td>.216 sec</td>
</tr>
<tr>
<td>n⁵</td>
<td>3.2 sec</td>
<td>1.7 min</td>
<td>13 min</td>
</tr>
<tr>
<td>2ⁿ</td>
<td>1.0 sec</td>
<td>12.7 days</td>
<td>366 cent</td>
</tr>
<tr>
<td>3ⁿ</td>
<td>58 min</td>
<td>3855 cent</td>
<td>10¹³ cent</td>
</tr>
</tbody>
</table>

### (non-)effect of Moore’s Law:

<table>
<thead>
<tr>
<th></th>
<th>curr size</th>
<th>100×</th>
<th>1000×</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>N</td>
<td>100N</td>
<td>1000N</td>
</tr>
<tr>
<td>n²</td>
<td>N</td>
<td>10N</td>
<td>31.6N</td>
</tr>
<tr>
<td>n³</td>
<td>N</td>
<td>4.64N</td>
<td>10N</td>
</tr>
<tr>
<td>n⁵</td>
<td>N</td>
<td>2.5N</td>
<td>3.98N</td>
</tr>
<tr>
<td>2ⁿ</td>
<td>N</td>
<td>N + 6.64</td>
<td>N + 9.97</td>
</tr>
<tr>
<td>3ⁿ</td>
<td>N</td>
<td>N + 4.19</td>
<td>N + 6.29</td>
</tr>
</tbody>
</table>
tractable: polynomial in (non-unary) input
P: solvable in polynomial time
NP: verifiable in polynomial time
NP-Hard: as hard as any problem in NP (via polytime reduction)
NP-Complete: NP-Hard and in NP

optimization vs decision: if opt were easy, decision would be too
reduce a to b: a → b in polytime, decide b, → decision for a
b hard by reduction from a: if a → b in polytime and b polytime, could solve a
“I can’t find an efficient algorithm, I guess I’m just too dumb.”
"I can’t find an efficient algorithm, because no such algorithm is possible!"
“I can’t find an efficient algorithm, but neither can all these famous people.”
<table>
<thead>
<tr>
<th>NP-Completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problems</td>
</tr>
<tr>
<td>Exponentials</td>
</tr>
<tr>
<td>Terms</td>
</tr>
<tr>
<td>Why</td>
</tr>
</tbody>
</table>

- asst 10
- asst 11
- wildcard
NP

Definitions
NP-Completeness
EOLQs
**Definitions**

\[ P = \{ L \subseteq \{0, 1\}^* : \exists \text{ algorithm that decides } L \text{ in poly time } \} \]

\[ A(x, y) \text{ verifies } L \text{ iff for any input } x \in L \exists \text{ certificate } y \text{ that proves } x \in L \text{ and } \nexists \text{ certificate iff } x \notin L \]

\[ \text{NP} = \{ L \subseteq \{0, 1\}^* : \exists \text{ algorithm } A(x, y) \text{ that can use certificate } y \text{ with } |y| = O(|x|^c) \text{ to verify } L \text{ in polynomial time } \} \]

\[ P \neq \text{NP?} \]

\[ \text{co-NP} = \{ L \subseteq \{0, 1\}^* : \overline{L} \in \text{NP} \}. \]

\[ \text{NP} \neq \text{co-NP? eg } L \in \text{NP} \Rightarrow \overline{L} \in \text{NP}? \]
polynomial-time reducible: $L_1 \leq_P L_2$ iff $\exists$

polynomial-time computable function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that for all $\{0, 1\}^*$, $x \in L_1$ iff $f(x) \in L_2$.

$L$ is NP-Complete iff $L \in \text{NP}$ and $\forall L' \in \text{NP}$, $L' \leq_P L$. 

For example:

- What’s still confusing?
- What question didn’t you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*