# Science 1000: Lecture #4 (Wareham):

# Mission Impossible: Proving Computational Intractability

Some easy.
Some seem hard. Just me?
Turns out no!

# The Crux of the Matter

- Some problems are solvable in polynomial time, e.g., binary search, list sorting, and can be solved in practice for large input sizes; some, e.,g., bin packing, cannot.
- With problems that are not known to be solvable in polynomial time, have we just not thought of a good algorithm yet, or are they genuinely intractable?

| |
|---|
| HOW CAN WE PROVE INTRACTABILITY? |

# Foundations of Complexity Analysis: Arm Wrestling

Arnold

Betty

# Best in Two?
# The Logic of Pairwise Comparison

easy to beat

**Arnold** *beaten by* **Betty** .

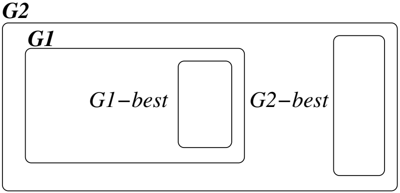
hard to beat

- Establish better arm wrestler by a two-person match.
- If Arnold is beaten by Betty:
  1. Arnold is no better than Betty
     (if Betty is easy to beat then Arnold is easy to beat)
  2. Betty is at least as good as Arnold
     (if Arnold is hard to beat then Betty is hard to beat)

# Best in Group?
# Pairwise Comparison in Groups


- Establish best arm wrestler in group $G$ by a tournament composed of two-person matches.
- The winner of a tournament is at least as good as everybody else in the group.

# Better than Group?
# The Logic of Group Inclusion




- Suppose we have two groups $G_1$ and $G_2$ such that $G_1$ is contained (but not necessarily fully contained) in $G_2$.
- If a person is the best for $G_2$, then they are better than anyone in $G_1$ modulo the conjecture that $G_1$ is fully contained in $G_2$, i.e., $G_1 \neq G_2$.

# Foundations of Complexity Analysis
# Reductions between Problems

- A **reduction** from problem $\mathcal{A}$ to problem $\mathcal{B}$ ($\mathcal{A}$ **reduces to** $\mathcal{B}$) is an algorithm for solving $\mathcal{A}$ that uses an algorithm for solving $\mathcal{B}$.

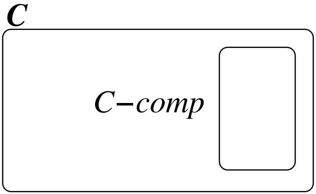
```
Algorithm solveA:
    blah blah
    blah blah blah
    blah blah
      ........

      x = solveB(x, y, z)
      .........
    return answer
```

# Hardest in Two?
# The Logic of Reducibility

$$\begin{array}{ccc} & \xleftarrow{\text{easy to solve}} & \\ \textbf{Problem A} & \textit{reduces to} & \textbf{Problem B} \\ & \xrightarrow[\text{hard to solve}]{} & \end{array}$$

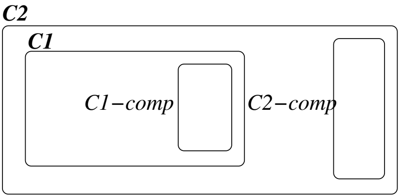
- Establish harder problem by poly-time reduction.
- If problem $\mathcal{A}$ reduces to problem $\mathcal{B}$:
  1. $\mathcal{A}$ is no harder than $\mathcal{B}$
     (if $\mathcal{B}$ is easy to solve then $\mathcal{A}$ is easy to solve)
  2. $\mathcal{B}$ is at least as hard as $\mathcal{A}$
     (if $\mathcal{A}$ is hard to solve then $\mathcal{B}$ is hard to solve)

# Hardest in Class?
# Reducibility in Classes



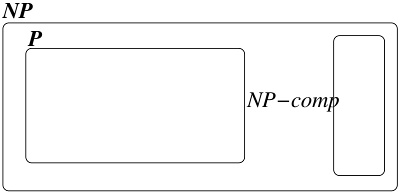

- Establish hardest problem in class $C$ by reductions.
- The hardest problems in $C$ (the $C$-Complete problems) are at least as hard as any problem in $C$.

# Harder than Class? The Logic of Class Inclusion




- Suppose we have two classes $C_1$ and $C_2$ such that $C_1$ is contained (but not necessarily fully contained) in $C_2$.
- If a problem is $C_2$-Complete, then that problem is harder than any problem in $C_1$ modulo the conjecture that $C_1$ is fully contained in $C_2$, i.e., $C_1 \neq C_2$.

# Harder than Poly-Time?
# The Logic of $NP$-Completeness




- Let $P$ be the class of poly-time solvable problems and $P$ be contained (but not necessarily fully contained) in class $NP$.
- If a problem is $NP$-Complete, then that problem is not poly-time solvable modulo the conjecture that $P$ is fully contained in $NP$, i.e., $P \neq NP$.

# Dealing with Intractability

- First $NP$-Complete problem proven in 1971; thousands proven since (including Bin Packing and many other industrially-important problems).
- Unless $P = NP$, no NP-complete problem can be solved in poly-time ... but we still need to solve these problems!!!

HOW DO WE SOLVE $NP$-COMPLETE PROBLEMS?

# Science 1000: Lecture #4 (Wareham):

## Mission Impossible:
## Proving Computational Intractability

Some easy.
Some seem hard. Just me?
Turns out no!