

# Complexity Theory

CS 800

- Algorithms is about algorithms
- Complexity theory is about problems

*In brief:* Complexity theory can prove lower bounds on problems

## Sorting

- Sorting algorithms have  $\Theta(n^2)$ ,  $\Theta(n \log n)$
- Can do better?

## Comparison algorithms

- Input is an array  $a$  of size  $n$  over an ordered set
- boolean  $\text{isLessThan}(i, j)$ 
  - compares  $a[i]$ ,  $a[j]$
- no other info about input is available

## Theorem

- The running time of any comparison sort algorithm is  $\Omega(n \log n)$

## Decision Problems

- A *decision problem* is defined as a set of things
- A *solution* to a decision problem is an algorithm that determines whether a thing is in the set.

## Examples

- PRIMES = {  $n$  |  $n$  is prime }
- TRAVELING-SALESMAN {  $(G, n)$  |  $G$  is a graph with weighted edges and there is a path visited all vertices with total weight at most  $n$  }

## Decision problems vs. other problems

## P vs. NP

- $P = \cup \text{DTIME}(n^k)$
- $NP = \cup \text{NTIME}(n^k)$

## P vs. NP

- All P problems are in NP
- Some (maybe all) NP problems are in P
- Many problems are neither

## NP-Hardness

- A problem is NP-hard if it is “as hard” as any other NP problem

## NP-Complete

- A problem is NP-complete if it is NP-hard and NP-complete

## P

- HORNSAT = { F | F is a boolean horn formula having a satisfying assignment }
- SHORTPATH = { (G, n) | G is a graph having a path no longer than n }
- EUTOUR = { G | G has an Euler tour }

## ??

- ISO = { (G, H) | G and H are graphs that are isomorphic to each other }

## NP-Complete

- SAT = { F | F is a boolean formula having a satisfying assignment to its variables }
- LONGPATH = { (G, n) | G is a graph having a simple path no shorter than n }
- HAMCYCLE = { G | G has a Hamiltonian cycle }

## Many-one reductions

- Used to prove that a problem is “as hard” as another problem
- $S \leq_r T$  mean T is “as hard” as S

## P TIME many-one reductions

- $S \leq_m T$  (S many-one reduces to T) iff there exists a total function  $f: S \rightarrow T$  such that, for all  $s$ ,  $s$  is in S if and only if  $f(s)$  is in T.
- If  $f$  runs in (deterministic) polynomial time, then S “polynomial-time many-one reduces” to T

## Example

- 3-CNF SAT = { F | F is a 3-CNF formula having a satisfying assignment }
- Use it to prove IND-SET = { (G, n) | G has a independent set no smaller than n }

## Another Example

- VERTEX-COVER
  - INSTANCE: a graph  $G$ , and a number  $k$
  - QUESTION: is there a subset  $U$  of  $V(G)$  of cardinality  $k$  such that, for every edge  $\{u,v\}$  in  $E(G)$ ,  $u$  or  $v$  is in  $U$ ?

## Determinism vs. Nondeterminism

- Based on Turing machines
- Nondeterministic Turing machines vs. deterministic ones.

## Determinism vs. nondeterminism

```
SEARCH (G, w)
// is there a vertex w?
foreach (v in V(G))
  color(v) := white
foreach (v in V(G))
  if (color(v) == white)
    VISIT (G, v, w)
```

## Determinism

```
VISIT (G, v, w)
color(v) := black
foreach u in adj(v)
  if (color(u) == white)
    VISIT (G, u, w)
```

## Nondeterminism

```
VISIT (G, v, w)
color(v) := black
foreach u in adj(v)
  if (color(u) == white)
    fork (VISIT (G, u, w))
```

## DTIME vs. NTIME

- $DTIME(f(n)) = \{ S \mid S \text{ is a decision problem having a solution that runs in } O(f(n)) \}$
- $NTIME(f(n)) = \{ S \mid S \text{ is a decision problem having a nondeterministic solution that runs in } O(f(n)) \}$

## Another characterization

- A set  $S$  is in NP if there exists a deterministic polynomial-time Turing machine  $D$  such that  $x$  is in  $S$  if and only if  $D(x)$  accepts.
- A set  $S$  is in NP if there exists a polynomial  $p$  and a deterministic polynomial-time Turing machine  $D$  such that  $x$  is in  $S$  if and only if there exists a string  $w$  (called a witness) such that  $|w| \leq p(|x|)$  and  $D(x,w)$  accepts