Lecture 22 Complexity, Reductions

Tuesday, April 16, 2019 9:25 PM

- reduction
  - problem A can be reduced to problem B, if one can design an algorithm for problem A using an algorithm for problem B as a subroutine
  - if A can be reduced to B

    A is "easier" than B $\quad A \leq B$
  - example

    LIS: input $\{4, 5, 2, 3, 6, 9, 7\}$

    goal: find the longest subsequence, increasing

    $\{2, 3, 6, 7\}$

    LCS: input $\{3, 2, 1, 5, 4\}$ $\{2, 5, 4, 3, 1\}$

    goal: find longest sequence that is a subsequence of both inputs

    $\{2, 5, 4\}$

```
LIS(X)
{
    Y = sort(X);
    return LCS(X, Y);
}
```

$X = \{4, 5, 2, 3, 6, 9, 7\}$

$Y = \{2, 3, 4, 5, 6, 7, 9\}$

$\{2, 3, 6, 7\}$

Proof: ① every common subseq. of X and Y is an increasing subseq. of X.

② every increasing subseq of X is also a common subseq of X and Y.

$LIS \leq LCS$

- complexity
  - P (polynomial time): class of decision problems that

can be solved in polynomial time.

- NP (<u>nondeterministic</u> polynomial time) set of decision problems that can be verified in polynomial time.

```
Verify (input, answer, explanation)
{
    if answer == YES
        if explanation supports answer
            accept
        else
            reject
}
```

run in polynomial time.

if true answer is yes, then there exists an explanation that makes verify( ) accept.

if true answer is no, then no matter what explanation verify( ) will reject.

- NP-complete problem

a problem B is NP-complete if

(1) $B \in NP$

(2) for every $A \in NP$, there is a polynomial time reduction from A to B. $(A \leq B)$