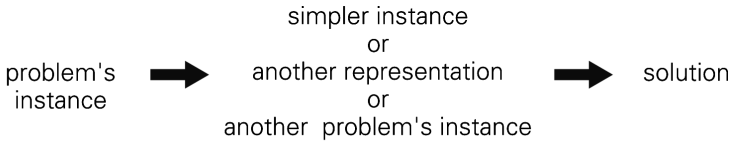
Class 13

TRANSFORM-AND-CONQUER
PRESORTING
HORNER'S RULE

Transform-and-Conquer

Transform-and-Conquer is about transforming a problem instance.

- *Instance Simplification*: a simpler/more convenient instance of the same problem.
- *Representation Change*: a different representation of the same instance
- *Problem Reduction*: a different problem for which an algorithm is already available



To Presort or Not to Presort?

Some problems can be solved more efficiently, if the elements in a list are sorted.

Examples:

- Searching for an element
- Checking element uniqueness in an array
- Calculating the median value

To Presort or Not to Presort?

- Sorting typically takes $\Theta(n \log n)$ time.
- Notice that maintaining a sorted data structure (while adding elements to it) takes $\Theta(n \log n)$ time in case of balanced search trees, but not for
  - Arrays
  - Linked Lists
  - ArrayLists
- Can we solve the unsorted problem in less than $(n \log n)$ time?
- When might it pay off to sort anyway?

Sorting Theory

In section 11.2, we will see that the lower bound for comparison based sorting is $\Omega(n \log n)$

How about non-comparison based sorts, you ask.

- Constant time sorting
- Linear time sorting

Evaluation of Polynomials

Problem: Consider the following expression:

$$2x^4 - x^3 + 3x^2 + x - 5$$

Develop a sequence of instructions to evaluate this particular expression.

You may not use exponentiation.

Horner's Rule for Polynomial Evaluation

Horner's Rule: nest the polynomial.

Example: $p(x) = 2x^4 - x^3 + 3x^2 + x - 5$

$= x\,(2x^3 - x^2 + 3x + 1) - 5$

$= x\,(x\,(2x^2 - x + 3) + 1) - 5$

$= x\,(x\,(x\,(2x - 1) + 3) + 1) - 5$

$= x\,(x\,(x\,(x\,(2) - 1) + 3) + 1) - 5$

Horner's Rule for Polynomial Evaluation

Problem: Given a polynomial of degree n

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

and a specific value of x, find the value of p at that point.

ALGORITHM *Horner*$(P[0..n], x)$

```
//Evaluates a polynomial at a given point by Horner's rule
//Input: An array P[0..n] of coefficients of a polynomial of degree n
//       (stored from the lowest to the highest) and a number x
//Output: The value of the polynomial at x
p ← P[n]
for i ← n − 1 downto 0 do
    p ← x * p + P[i]
return p
```

Horner's Rule Example

Example. Compute p(3) for

$$p(x) = 2x^4 - x^3 + 3x^2 + x - 5 = x(x(x(x(2) - 1) + 3) + 1) - 5$$

Coefficients	2	–1	3	1	–5
x = 3	2	2*3–1=5	5*3+3=18	18*3+1=55	55*3–5=160

ALGORITHM *Horner*$(P[0..n], x)$
//Evaluates a polynomial at a given point by Horner's rule
//Input: An array $P[0..n]$ of coefficients of a polynomial of degree n
// (stored from the lowest to the highest) and a number x
//Output: The value of the polynomial at x
$p \leftarrow P[n]$
for $i \leftarrow n - 1$ **downto** 0 **do**
$p \leftarrow x * p + P[i]$
return p