# Announcement

No quiz this week

General suggestion: read the textbook!

---

## §3.1 Algorithms

Def: An <u>algorithm</u> is a finite sequence of precise steps

**Properties:**

- Input
- Output
- Definiteness: Steps are precisely-defined
- Correctness: Always gives the right answer
- Finiteness: Finite # steps for any input
- Effectiveness: You can actually do each step
- Generality: Works for all possible inputs

---

Ex: Making change

Basic idea: we have a value of $n$ "cents" and we want to make change using coins of values $c_1, c_2, \ldots, c_r$

Greedy Change-Making Algorithm: pos. int.

procedure change($c_1, c_2, \ldots, c_r$ : values of coins,
where $c_1 > c_2 > \cdots > c_r$ ; $n$ : pos. int.)

$a := 5$
means set $a = 5$

for $i := 1$ to $r$

  $d_i := 0$ (di is the num. coins of value $c_i$)

  while $n \geq c_i$

    $d_i := d_i + 1$ (adds a coin of value $i$)

    $n := n - c_i$ ($c_i$ less value remaining)

return $d_1, d_2, \ldots, d_r$

This is an example of an <u>optimization problem</u>

Optimization problem: maximize/minimize some parameter
e.g. Give change using the fewest num. of coins possible

Greedy algorithm: Try to solve the optimization problem by making the "best" choice at each step

*doesn't always give the optimal solution*

E.g. $c_1 = 13$ $\quad n = 18$

$c_2 = 9$

$c_3 = 1$

Start: 16 cents, making change w/ coins worth 13, 8, and 1 cent(s).

Step 1: Give 1 13-cent coin

Since $13 \cdot 1 = 13 \leq 18$

But $13 \cdot 2 = 26 > 18$

Remaining change: 5 cents

Step 2: Give 0 8-cent coins

Since $8 \cdot 1 > 5$

Remaining change: 5 cents

Step 3: Give 5 1-cent coins

Since $1 \cdot 5 = 5$

Remaining change: 0 cents

Gave 1 13-cent coin & 5 1-cent coins

(Would have been more efficient to give 2 9-cent coins)

Ex: Finding max. elt. in a finite sequence

procedure max($a_1, \ldots, a_n$: integers)

$m := a_1$

for $i := 2$ to $n$

    if $m < a_i$ then $m := a_i$ — set $m$ equal to $a_i$

return $m$

Check properties:

- Input ✓
- Output ✓
- Definiteness: Yes, because we're only incrementing, assigning value, and checking conditions
- Correctness: Yes, always returns max. elt.
- Finiteness: Yes, goes through the list once and terminates
- Effectiveness: Yes, because we're only incrementing, assigning value, and checking conditions
- Generality: Yes, works for all finite lists of integers

# Searching algorithms:

General problem: locate an elt. $x$ in a list of distinct elts. $a_1, \ldots, a_n$, or determine it's not in the list

Linear search algorithm

(Use when list is unordered)

Input: integer $x$

list of integers: $a_1, \ldots, a_n$

Output: Location of $x$ in list (or 0 if not found)

Algorithm:

Set $i := 1$

while ( $i \leq n$ and $x \neq a_i$ )

$i := i+1$

$$\text{location} := \begin{cases} i, & \text{if } i \leq n \\ 0, & \text{otherwise} \end{cases}$$

return location

Binary search algorithm
(Use when list is ordered)

Input: integer $x$
list of integers: $a_1, \ldots, a_n$ w/ $a_1 < a_2 < \cdots < a_n$

Output: Location of $x$ in list (or 0 if not found)

Algorithm:

Let $i := 1$
Let $j := n$
(endpoints of search interval

While $i < j$
    Let $m := \lfloor \frac{i+j}{2} \rfloor$
    if $x > a_m$
        $i := m+1$
    else
        $j := m$

location $:= \begin{cases} i, & \text{if } x = a_i \\ 0, & \text{otherwise} \end{cases}$

return location

# Sorting algorithms:

General problem: Sort a list in increasing order

Many different algorithms

Bubble sort algorithm:

Input: list of integers $a_1, \ldots, a_n$

Output: list of integers which is the original list in inc. order

Algorithm:

for $i := 1$ to $n-1$

  for $j := 1$ to $n-i$

    if $a_j > a_{j+1}$

      Swap $a_j$ and $a_{j+1}$

 return $a_1, \ldots, a_n$

Class activity: perform bubble sort on the list

$3, 2, 4, 1, 5$

Insertion sort algorithm (if time):

Input: list of integers $a_1, \ldots, a_n$

Output: list of integers which is the original list in inc. order

Algorithm:

for $j := 2$ to $n$

Let $i := 1$

While $a_j > a_i$ (finding the spot for $a_j$)

$i := i+1$

Let $m := a_j$ (insert $a_j$ into spot $m$)

for $k := 0$ to $j-i-1$

$a_{j-k} := a_{j-k-1}$ (shift other elts. to make room)

$a_i =: m$

return $a_1, \ldots, a_n$

Class activity: perform insertion sort on the list

3, 2, 4, 1, 5