Math 135 - Algorithms

Note Title 9/26/2012

Announcements

- HW due Friday
- Review Monday
- Midterm Wednesday

Algorithm

A set of instructions for solving a problem.

(NOT necessarily a program!)

Examples

- tieing a shoe
- solving puzzle
- recipe

We often use pseudocode to write down computer algorithms

Common Concepts: (see 3.1)

- if statements
- loops
- variable
- functions or procedures
- input / output

Example: Given a list of numbers, how do you find the largest?

Go element by element:

Keep current max

if current element is > max

make max = current element

Pseudocode:

inputs: a list of value

procedure

```
FINDMAX (a_1, a_2, ..., a_n):
    max := a_1
    for i := 2 to n
        if max < a_i
            max := a_i
    return max
```

hands max back to outside work

x := y says store y's value in x

why not =?

Properties of an algorithm:

- Input : usually specified outside our program
- Output : should do something
- Definiteness : each step should be clear
- Correctness : should work
- Finiteness : should not run forever
- Generality : should work for any inputs

Searching

Suppose I give you a list of numbers $a_1, \ldots, a_n$ and ask if $x \in \{a_1, \ldots, a_n\}$.

How would you check?

Check each element in the list to see if $= x$

If x is in list, return i such that $x = a_i$

If x is not in list, return 0

-2, 5, -11, 6, 32

x = 52

i = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

```
LINEAR SEARCH (x, a1, a2, ..., an):

    i := 1
    while (i ≤ n and x ≠ ai)
        i := i + 1
    if i ≤ n
        location := i
    else
        location := 0
    return location
```

location = 0

Another way to search:

Ex: Suppose I say "take out your book and find page 194". What do you do?

check middle, go left or right

Is this the same as our last search algorithm?

NO

Binary Search

When searching in a sorted list, we can do better than linear search.

- Compare to middle element of list.
  - If middle element is > x:
    search left half
  - If middle element is < x:
    search right half

sorted list

```
BINARY SEARCH(x, a_1, ..., a_n):
    l := 1
    r := n
    location := 0
    while l < r
        m := ⌊(l + r)/2⌋

        if x > a_m
            then l := m + 1
        else r := m
        if x = a_m then location := m

    return location
```