Math 135 - Algorithms (3.1)

Note Title 2/24/2010

Announcements

- HW is due on Wed. the 20th
- On midterm problem with #10
  Resubmit #10 on Friday

#8 Show $\sqrt{2}$ is irrational.

Suppose $\sqrt{2}$ is rational.

$\Rightarrow \sqrt{2} = \frac{p}{q}$, $p, q \in \mathbb{Z}$ & $q \neq 0$.

then $(\sqrt{2})^2 = \left(\frac{p}{q}\right)^2$

so $2 = \frac{p^2}{q^2}$, $\frac{p^2}{q^2}$ is a rational #

Since $\sqrt{2}$ is irrational.

11a

$f(S) = \#$ of 1 bits

is onto:

Take any $n \in \mathbb{N}$ and show a bitstring S so that $f(S) = n$

Let S be the string with n 1's + no zeroes.

$$S = \underbrace{1 \ldots 1}_{n \text{ times}}$$

$$A \times B = \{(a,b) \mid a \in A,\ b \in B\}$$

#5) Suppose 2 sets with $A \times B = \emptyset$.

$$A = \emptyset$$

$$B = \{1, 2\}$$

$$A \times B = \emptyset$$

#9) IH: $\displaystyle\sum_{k=0}^{n-1} 2^k = 2^n - 1$

IS: $\displaystyle\sum_{k=0}^{n} 2^k = \underbrace{\sum_{k=0}^{n-1} 2^k}_{2^n - 1} + 2^n = 2^n + 2^n - 1$

$$= 2 \cdot 2^n - 1$$

$$= 2^{n+1} - 1$$

Algorithm (Section 3.1)

A set of instructions for solving a problem
(NOT necessarily a program!)

Examples:
- tying a shoe
- driving
- walking
- recipes

We often use pseudocode to write down computer algorithms.

Common programming concepts:

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

Ex: Pseudocode to find the maximum element in a sequence $a_1 .. a_n$

```
FINDMAX(a_1, a_2, ..., a_n):

    max := a_1
    for i := 2 to n
        if max < a_i
            max := a_i
    return max
```

Two important structures:

for loops & if statements

Searching

Suppose I give you a list of numbers $a_1, \dots, a_n$ and ask if $x \in \{a_1, \dots, a_n\}$.
How would you check?

for loop to check each element

while (bool) → if bool is true, execute the instructions inside the loop

LINEARSEARCH($x, a_1, \dots, a_n$):

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

$i = \cancel{1}, \cancel{2}, 3$, $n = 5$

10 12 -1 3 5

$x = -1$

$i = 1$?

Another Search Strategy:

Ex: Take out your book & open it to page 171.

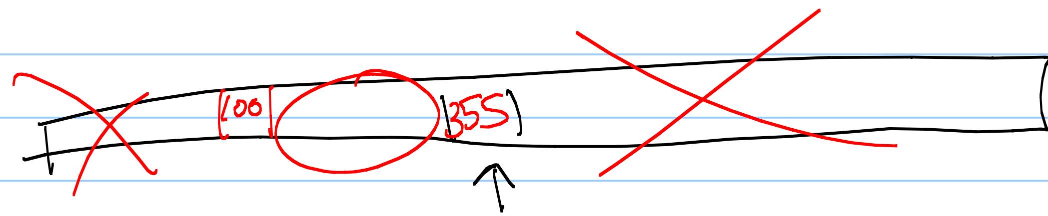
How does your algorithm to do this differ from the linear search algorithm?

binary search - check the middle

(assume the list is sorted)

When searching in a sorted list, we can do a faster search called <u>binary search</u>.

- Compare to middle element of list.



- If that element is bigger than x, search in the left half   x=171

- If that element is smaller than x, search in the right

(pseudocode in book)

Sorting:

Fundamental CS problem:

Given a list of n things, put them in order

How?

Comparing elements, & swap if out of order

Bubble Sort:

Compare adjacent elements + switch them if in wrong order

3 2 4 1 5

2 3 4 1 5

2 3 1 4 5

Pseudocode

```
BUBBLESORT(a_1 ... a_n):
    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}
```

Insertion Sort

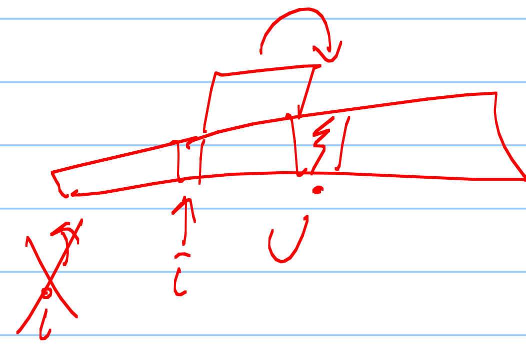
- If first i items are sorted, take $(i+1)^{st}$ and put it in correct spot.

1	2	3	4	5	1
$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\uparrow$
a_1	a_2	a_3	a_4	a_5	a

Pseudo code

INSERTION SORT $(a_1 .. a_n)$:

for $j := 2$ to n
    $i := 1$
    while $a_j > a_i$
        $i := i+1$
    temp $:= a_j$
    for $k := j$ to $i+1$
        $a_k := a_{k-1}$
    $a_i :=$ temp



Complexity of Algorithms

Comparing which algorithms are "better" can be tricky.

Issues:

- time them on a machine
  - is input good?
  - different on different machines

So:

We define complexity in terms of the number of operations.

Usually, an operation is:

- add 2 things (or subtract or multiply)
- compare 2 things
- set a variable equal to something

atomic operations

But still - how do we compare?

We just saw 2 searching algorithms,
linear search & binary search.

One is <u>not</u> always better.

Linear search: 1
Binary search: 3

Find(36): | 36 | 40 | 58 | 100 | 101 | 125 |

Linear search: 4
Binary search: 1

Find(36): | 1 | 11 | 25 | 36 | 41 | 42 | 100 |

So how can we compare worst case performance?

☆ worst case performance

Ex: What is worst case complexity of FindMax?

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

1 ← max := a_1

$n-1$ ← for $i := 2$ to n

1 ← if max < a_i

1 ← max := a_i

1 ← return max

$$1 + (n-1)(2) + 1$$

$$= 2n - 2 + 2 = 2n$$