CS 2210 Discrete Structures

Algorithms and Complexity

Fall 2017

Sukumar Ghosh

What is an algorithm

A finite set (or sequence) of precise instructions

for performing a computation.

Example: Maxima finding

procedure *max* ($a_1, a_2, \ldots, a_n$: integers)

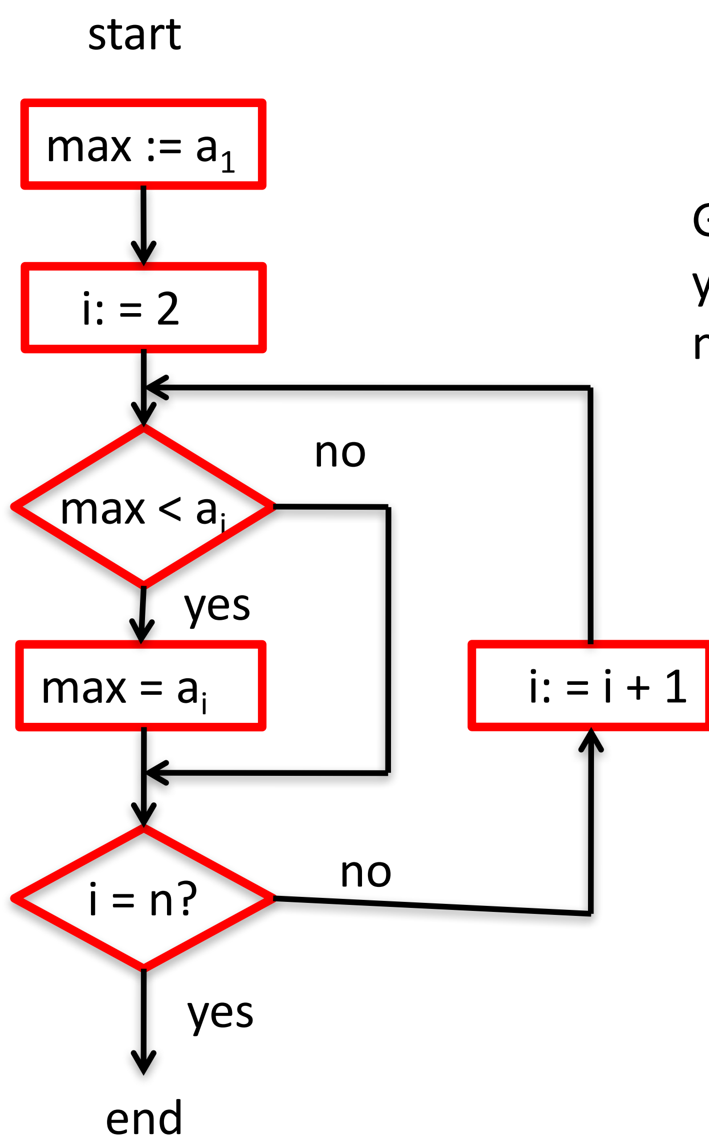
$max := a_1$

for i :=2 to n

    if $max < a_i$ then $max := a_i$

return *max* {the largest element}

Flowchart for maxima finding



Given n elements, can you count the total number of operations?

Time complexity of algorithms

Counts the largest number of basic operations required to execute an algorithm.

Example: Maxima finding

procedure *max* (*a1, a2, …, an*: integers)

max := *a1* *1 operation*

for i :=*2* to *n* *1 operation i:=2*

if *max* < *a1* then *max* := *ai* *{n-1 times}*

{2 ops + 1 op to check if i > n + 1 op to increment i}

return *max* {the largest element}

The total number of operations is 4(n-1)+2 = 4n-2

Time complexity of algorithms

Example of linear search (Search x in a list $a_1\ a_2\ a_3 \ldots a_n$)

k := 1 {1 op}

while k ≤ n do {n ops k ≤ n}

{if x = a_k then *found* else k: = k+1} {2n ops + 1 op}

search failed

The maximum number of operations is 3n+2. If we are lucky, then search can end even in the first iteration.

Time complexity of algorithms

Binary search (Search x in a sorted list $a_1 < a_2 < a_3 < \ldots < a_n$*)*

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$ {search failed}

How many operations? Roughly log n. Why?

Bubble Sort

```
procedure bubblesort ( A : list of items )
n = length (A)
repeat
      for i = 1 to n-1 do
            if A[i-1] > A[i] then swap (A[i-1], A[i])
            end if
      end for
      n: = n - 1
until n=0
end procedure
```

Bubble Sort

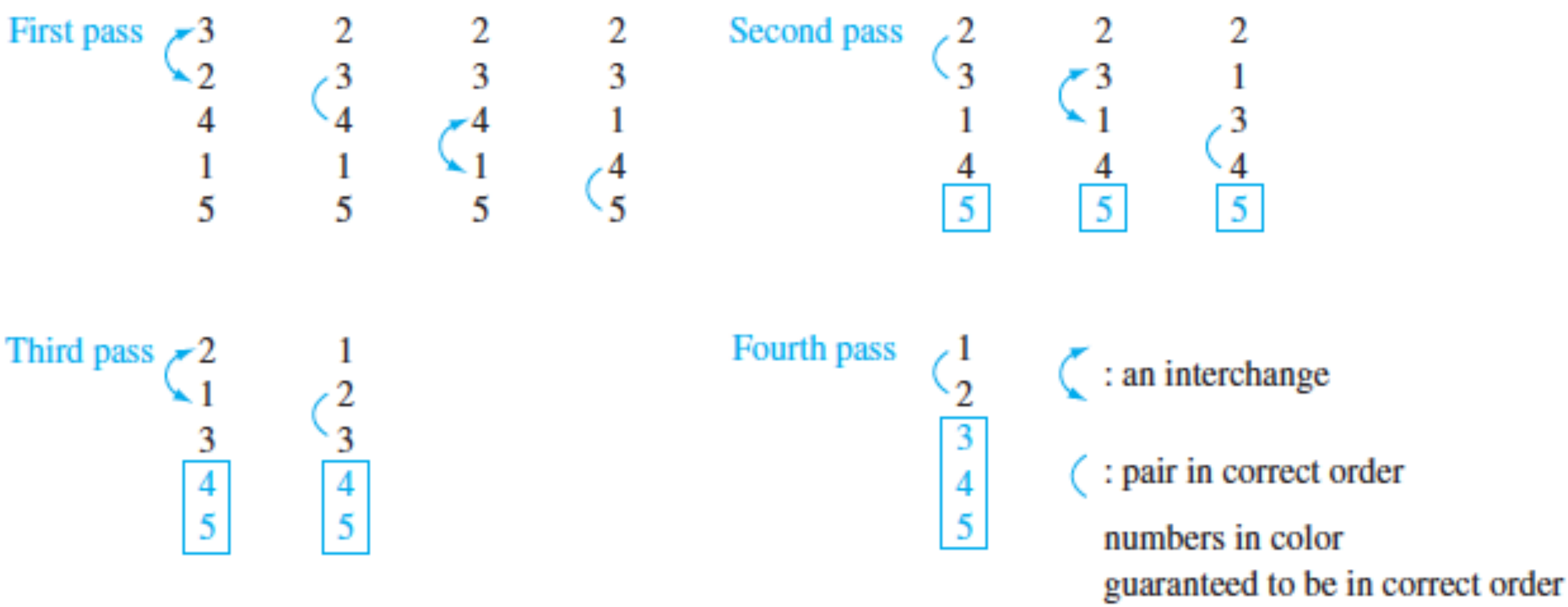


FIGURE 1 **The Steps of a Bubble Sort.**

Bubble Sort

3	2	4	1	5		n-1 operations
2	3	1	4	5	(first pass)	n-2 operations
2	1	3	4	5	(second pass)	n-3 operations
1	2	3	4	5	(third pass)	…
1	2	3	4	5	(fourth pass)	1

The worst case time complexity is

$$(n-1) + (n-2) + (n-3) + \ldots + 2$$
$$= n(n-1)/2 - 1$$

The Big-O notation

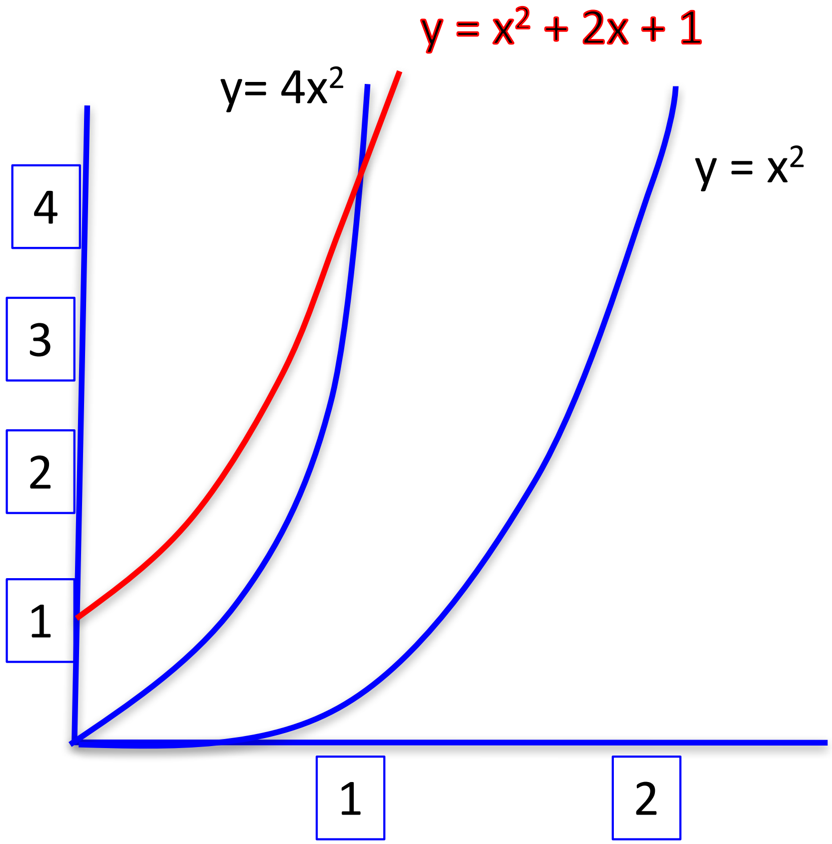
It is a measure of the growth of functions and often used to measure the complexity of algorithms.

DEF. Let f and g be functions from the set of integers (or real numbers) to the set of real numbers. Then f is O(g(x)) if there are constants **C** and **k**, such that

$$|f(x)| \leq C|g(x)| \qquad \text{for all } x > k$$

Intuitively, f(x) grows “slower than” some multiple of g(x) as x grows without bound. Thus O(g(x)) defines an upper bound of f(x).

The Big-O notation



$x^2 + 2x + 1 = O(x^2)$

Since = **4** $x^2 > x^2 + 4x + 1$

whenever x > **1**, $4x^2$ defines an upper bound of the growth of $x^2 + 2x + 1$

Defines an upper bound of the growth of functions

The Big-Ω (omega) notation

DEF. Let f and g be functions from the set of integers (or real numbers) to the set of real numbers. Then f is $\Omega(g(x))$ if there are constants C and k, such that

$$|f(x)| \geq C|g(x)| \qquad \text{for all } x > k$$

Example. $7x^2 + 9x + 4$ is $\Omega(x^2)$, since $7x^2 + 9x + 4 \geq 1.\ x^2$ for all x

Thus Ω defines the lower bound of the growth of a function

Question. Is $7x^2 + 9x + 4$ $\Omega(x)$?

The Big-Theta (Θ) notation

DEF. Let f and g be functions from the set of integers (or real numbers) to the set of real numbers. Then f is $\Theta(g(x))$ if there are constants C_1 and C_2 a positive real number k, such that

$$C1.|g(x)| \leq |f(x)| \leq C2.|g(x)| \qquad \text{for all } x > k$$

Example. $7x^2 + 9x + 4$ is $\Theta(x^2)$,

since $1.\, x^2 \leq 7x^2 + 9x + 4 \leq 8.\, x^2$ for all $x > 10$

Average case performance

EXAMPLE. Compute the average case complexity of the ***linear search*** algorithm.

a_1 a_2 a_3 a_4 a_5 ….. a_n (Search for x from this list)

If x is the 1st element then it takes 5 steps

If x is the 2nd element then it takes 8 steps

If x is the i^{th} element then it takes (3i + 2) steps

So, the average number of steps = 1/n [5+8+…+(3n+2)] = ?

Classification of complexity

Complexity	**Terminology**
$\Theta(1)$	Constant complexity
$\Theta(\log n)$ $\Theta(\log n)^c$	Logarithmic complexity Poly-logarithmic complexity
$\Theta(n)$	Linear complexity
$\Theta(n^c)$	Polynomial complexity
$\Theta(b^n)$ $(b>1)$	Exponential complexity
$\Theta(n!)$	Factorial complexity

We also use such terms when Θ is replaced by O (big-O)

Exercise

Complexity of n^5	$O(2^n)$	True or false?
Complexity of 2^n	$O(n^5)$	True or false?
Complexity of $\log(n!)$	$\Theta(n \log n)$	True or false?
Complexity of $1^2+2^2+3^2+\ldots+n^2$	$\Omega(n^3)$	True or false?”

Let S = {0, 1, 2, …, n}. Think of an algorithm that generates all the subsets of three elements from S, and compute its complexity in big-O notation.

Greedy Algorithms

In optimization problems, many algorithms that use the best choice at each step are called greedy algorithms.

Example. Devise an algorithm for making change for n cents using quarters, dimes, nickels, and pennies using the least number of total coins?

Greedy Change-making Algorithm

Let $c_1, c_2, \ldots, c_r$ be the denomination of the coins,

(and c_i)

for i:= 1 to r

while n ≥ c_i

begin

add a coin of value c_i to the change

n := n- c_i

end

Let the coins be
1, 5, 10, 25 cents. For making 38 cents, you will use

1 quarter
1 dime
3 cents

The total count is 5, and it is optimum.

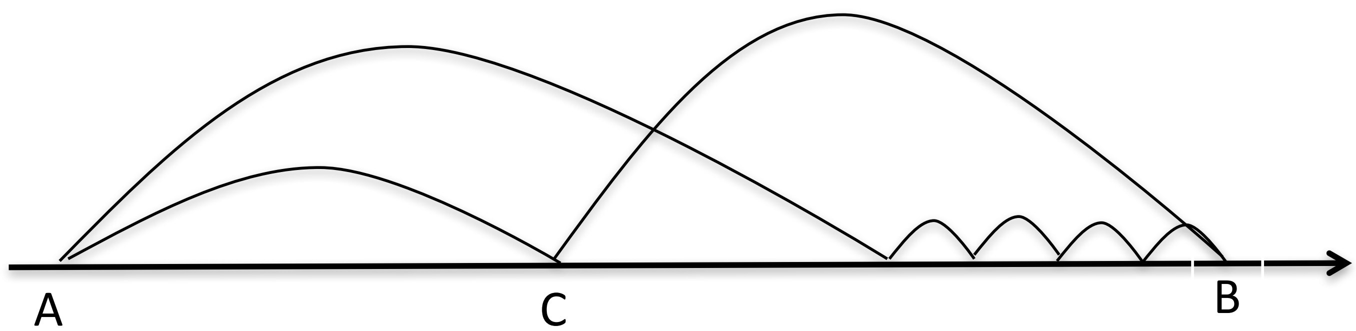
Question. Is this optimal? Does it use the least number of coins?

Greedy Change-making Algorithm

But if you don’t use a nickel, and you make a change for 30 cents using the same algorithm, the you will use 1 quarter and 5 cents (total 6 coins). But the optimum is 3 coins (use 3 dimes!)

So, greedy algorithms produce results, but the results may be sub-optimal.

Greedy Routing Algorithm



If you need to reach point B from point A in the fewest number of hops, Then which route will you take? If the knowledge is local, then you are tempted to use a greedy algorithm, and reach B in 5 hops, although it is possible to reach B in only two hops.

Other classification of problems

- Problems that have polynomial worst-case complexity are called tractable. Otherwise they are called intractable.

- Problems for which no solution exists are known as unsolvable problems (like the halting problem). Otherwise they are called solvable.

- Many solvable problems are believed to have the property that no polynomial time solution exists for them, but a solution, if known, *can be checked in polynomial time*. These belong to the class NP (as opposed to the class of tractable problems that belong to class P)

Estimation of complexity

	10	50	100	300	1000
$5n$	50	250	500	1500	5000
$n \times \log n$	33	282	665	2469	9966
n^2	100	2500	10000	90000	1 million (7 digits)
n^3	1000	125000	1 million (7 digits)	27 million (8 digits)	1 billion (10 digits)
2^n	1024	a 16-digit number	a 31-digit number	a 91-digit number	a 302-digit number
$n!$	3.6 million (7 digits)	a 65-digit number	a 161-digit number	a 623-digit number	unimaginably large
n^n	10 billion (11 digits)	an 85-digit number	a 201-digit number	a 744-digit number	unimaginably large

(The number of protons in the known universe has 79 digits.)
(The number of microseconds since the Big Bang has 24 digits.)

Source: D. Harel. *Algorithmics: The Spirit of Computing* . Addison-Wesley, 2nd edition, 1992

The Halting Problem

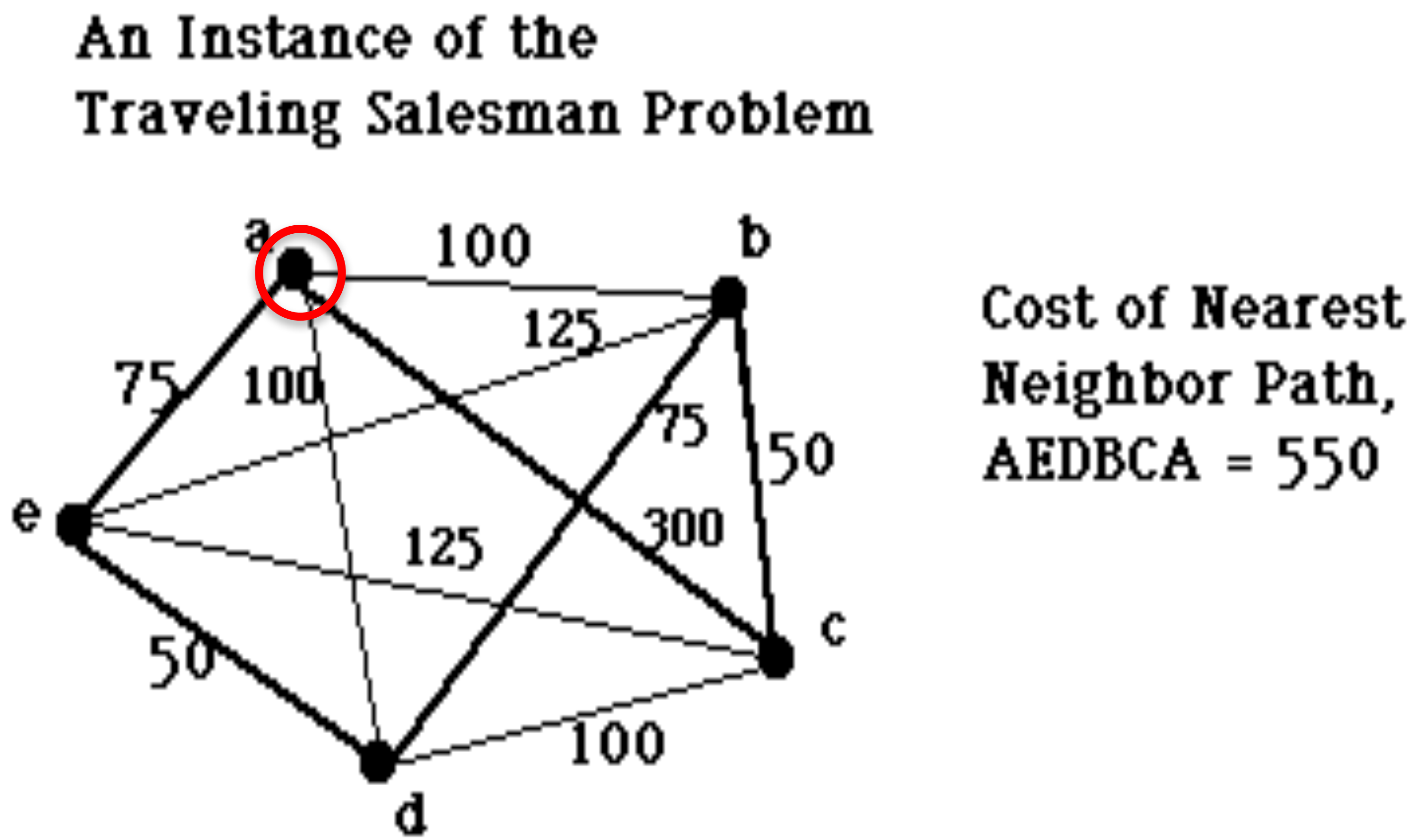
The **Halting problem** asks the question.

> *Given a program and an input to the program, determine if the program will eventually stop when it is given that input.*

- Run the program with that input. If the program stops, then we know it stops.
- But if the program doesn't stop in a reasonable amount of time, then **we cannot conclude that it won't stop**. Maybe we didn't wait long enough!

The question is not decidable in general!

The Traveling Salesman Problem



Starting from a node, you have to visit every other node and return
To you starting point. Find the shortest route? NP-complete

3-Satisfiability Problem

Consider an expression like this:

$$(x$$

Does there exist an assignment of values of x, y, z
so that this formula is true? NP-Complete problem!