Izmir Institute of Technology

CENG 115
Discrete Structures

Slides are based on the Text
***Discrete Mathematics & Its Applications* (6th Edition)**
by Kenneth H. Rosen

Algorithms

Discrete Mathematics, Rosen, 6th ed., § 3.1

What is an algorithm?

- Formal definition: An *algorithm* is a finite and ordered set of precise instructions for performing a computation or for solving a problem.
- It is the core of computer programming.
- A computer *program* is simply a description of an algorithm in a precise language (C, Java etc.)
- We say that a program (or a code) *implements* (or "is an implementation of") an algorithm.

Algorithms of your personal life

- Subtraction using borrowing (primary school math).
- Your favorite cooking recipe.
- How to register for classes at IYTE.

Algorithm Example

Find the maximum of a given sequence of integers.

1) Set the temporary maximum equal to the first integer.
2) Compare the next integer in the sequence with the temporary maximum. If it is larger, set the temporary maximum equal to this integer.
3) Repeat previous step until the end of the sequence.
4) Stop. The temporary maximum now is the largest integer in the sequence.

Pseudocode

- To describe algorithms, instead of English or a certain language like C or Java, we often prefer an informal, Pascal-like "*pseudo-code*" language.
- E.g.

```
begin
    sum := 0
    for i := 1 to n
        sum := sum + i
end
```

- Please review Appendix-3 in the book.

Max procedure in Pseudocode

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

$max := a_1$ {largest element so far}

for $i := 2$ **to** n {go thru rest of elements}

if $a_i > max$ **then**

$max := a_i$ {if bigger, change}

{at this point *max*' s value is the same as the largest integer in the list}

Executing an Algorithm

- When you write a piece of code/program, that code and its algorithm are *run (executed)* by the computer.
- Given the pseudocode of an algorithm, you can also execute it on paper, by working through all of its steps.
- In fact, until 20th century, the word ‘computer’ referred to a *person* who carry out computations.

Executing the Max algorithm

- Let $\{a_i\}$=7,12,3,11,8. Find its maximum…
- Set $max = a_1 = 7$.
- Look at next element: $a_2 = 12$.
- Is $a_2 > max$? Yes, so change *max* to 12.
- Look at next element: $a_3 = 3$.
- Is 3>12? No, leave *max* alone.
- $a_4 = 11$. Is 11>12? No, leave *max* alone.
- $a_5 = 8$. Is 8>12? No, leave *max* alone.

Another example task

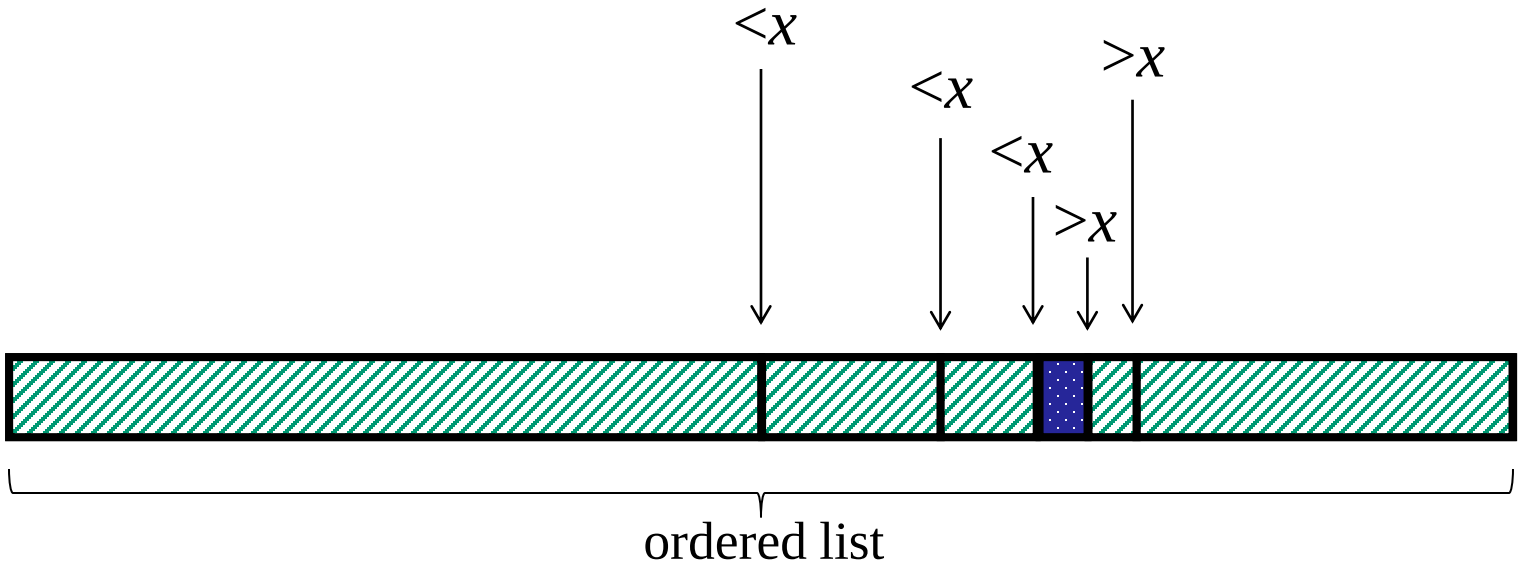
- Problem of *searching an ordered list*.
  - Given a list *L* of *n* elements that are sorted into a definite order (*e.g.*, numeric, alphabetical),
  - And given a particular element *x*,
  - Determine whether *x* appears in the list,
  - and if so, return its index (position) in the list.
- Problem occurs often in many contexts.
- Let's find an *efficient* algorithm!

Search Alg. #1: Linear Search

procedure *linear search*
(x: integer, $a_1, a_2, \ldots, a_n$: ordered integers)
$i := 1$
while $(i \leq n \wedge x \neq a_i)$
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
{*location* is the index of the term equal to x or 0 if x is not found in the list}

Search Alg. #2: Binary Search

- Basic idea: On each step, look at the *middle* term of the remaining list to eliminate half of it. It is much more efficient.



Search Alg. #2: Binary Search

procedure *binary search* (x:int, a_1,..,a_n: ordered ints)
  $i := 1$ {left endpoint of search interval}
  $j := n$ {right endpoint of search interval}
  while $i<j$ {while interval has >1 item}
  begin
    $m := \lfloor (i+j)/2 \rfloor$ {midpoint}
    if $x>a_m$ **then** $i := m+1$ **else** $j := m$
  end
  if $x = a_i$ **then** *location* $:= i$ **else** *location* $:= 0$
  {*location* is the index of x, or 0 if x is not found}

Exercise

- Devise an algorithm that finds the sum of all the integers in a list. [2 minutes]

procedure *sum*(a_1, a_2, …, a_n: integers)
    s := 0 {sum of elements so far}
    for i := 1 **to** n {go thru all elements}
        s := $s + a_i$ {add current item}
    {s is the sum of all items}

Sorting

- Problem of *sorting an arbitrary list*.
  - We are given a list *L* of *n* elements (*e.g.*, numeric, alphabetical).
  - Sorting is putting these elements into a list in which the elements are in increasing order.
- There are different approaches. Widely studied for both education and scientific purposes.
- Let's find an *efficient* algorithm!

Selection Sort

procedure selection sort(a_1,a_2,...,a_n)
for i:=1 to n-1 {at each turn}
begin
  m:=i
  for j:=i+1 to n {selects the smallest element in the so far unsorted part of the list}
    if a_j<a_m **then**
      m:=j
  interchange a_m and a_i {put that element in the correct order}
end
{a_1,...,a_n are sorted}

https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
http://www.cs.oswego.edu/~mohammad/classes/csc241/samples/sort/Sort2-E.html

Insertion Sort

procedure insertion sort($a_1,a_2,...,a_n$: real numbers with $n \geq 2$)
for j:=2 to n {takes an element at a time}
begin
  i:=j-1
  while $a_j < a_i \wedge i > 0$ {compares it with all the previous (so far sorted) elements until finding a larger element}
    i:=i-1
  m:=a_j {and locates the correct place for it}
  for k:=0 to j-i-2
    a_{j-k}:=a_{j-k-1} {shift the elements that are bigger than the current element to the right}
  a_{i+1}:=m
end

Home exercise:
What does this algorithm do?

procedure change (c_1,c_2,...,c_r: values of denominations of coins, where $c_1 > c_2 > ... > c_r$ and n: a positive integer)

for i:=1 to r

  while $n \geq c_i$

  begin

    add a coin with value c_i to the change

    $n := n - c_i$

  end