Discrete Mathematics

THE FUNDAMENTALS:
ALGORITHMS & THE INTEGERS

FPT University

Department of Mathematics

Quynhon, 2023

Outline of Lecture

Textbook: Discrete Mathematics and Its Applications, Seventh edition, K.Rosen.

FPTU-QN

Upcoming ...

FPTU-QN

Algorithm

An **algorithm** is a finite set of precise instructions for performing a computation or for solving a problem.

Example. Describe an algorithm to solve quadratic equations.
Input: a, b, c: integers (coefficients)
Output: Solutions if they exists.

Algorithm:

- **Step 1**. If $a = 0$ then Print (This is not a quadratic equation).
- **Step 2**. Compute $\Delta = b^2 - 4ac$.
- **Step 3**. If $\Delta < 0$ then Print (No solution).
- **Step 4**. If $\Delta = 0$ then compute $x = -b/2a$.
- **Step 5**. If $\Delta > 0$ then compute
  $x_1 = (-b + \sqrt{\Delta})/(2a), \quad x_2 = (-b - \sqrt{\Delta})/(2a)$.

FPTU-QN

Properties of Algorithms

1. **Input**: An algorithm has input values from a specified set.
2. **Output**: From each set of input values an algorithm produces output values from a specified set, and they are solutions to the problem.
3. **Definiteness**: The steps of an algorithm must be defined precisely.
4. **Correctness**: An algorithm should produce the correct output values for each set of input values.
5. **Finiteness**: An algorithm should produce the desired output after a finite number of steps.
6. **Effectiveness**: It must be possible to perform each step of an algorithm exactly and in a finite amount of time.
7. **Generality**: An algorithm should be applicable for all problems of the desired form, not just a particular set of input values.

FPTU-QN

Some common algorithms

1. Find maximum/minimum element of a finite sequence.
2. Searching algorithms:
   - Linear search algorithm
   - Binary search algorithm
3. Sorting algorithms:
   - Bubble sort algorithm
   - Insertion sort algorithm.
4. Greedy change-making algorithm.

FPTU-QN

Finding Maximum Element

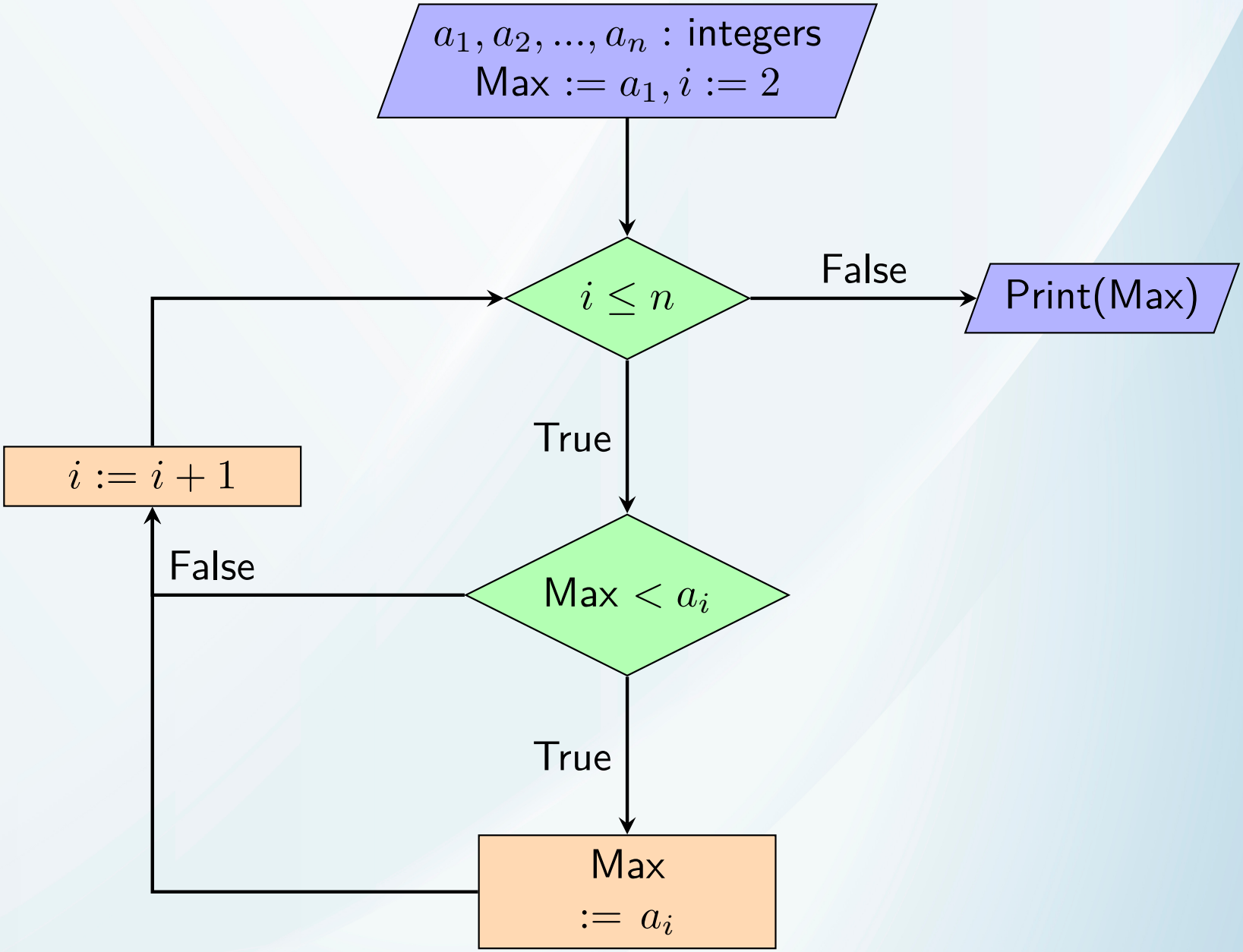
Input: Sequence of integers $a_1, a_2, \ldots, a_n$.
Output: The maximum number of the sequence.

Algorithm:

- **Step 1**. Set the temporary maximum be the first element.
- **Step 2**. Compare the temporary maximum to the next element, if this element is larger then set the temporary maximum to be this integer.
- **Step 3**. Repeat **Step 2** if there are more integers in the sequence.
- **Step 4**. Stop the algorithm when there are no integers left. The temporary maximum at this point is the maximum of the sequence.

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

FPTU-QN


$a_1, a_2, ..., a_n$: integers
Max $:= a_1, i := 2$
$i \leq n$
False
Print(Max)
True
$i := i + 1$
False
Max $< a_i$
True
Max
$:= a_i$

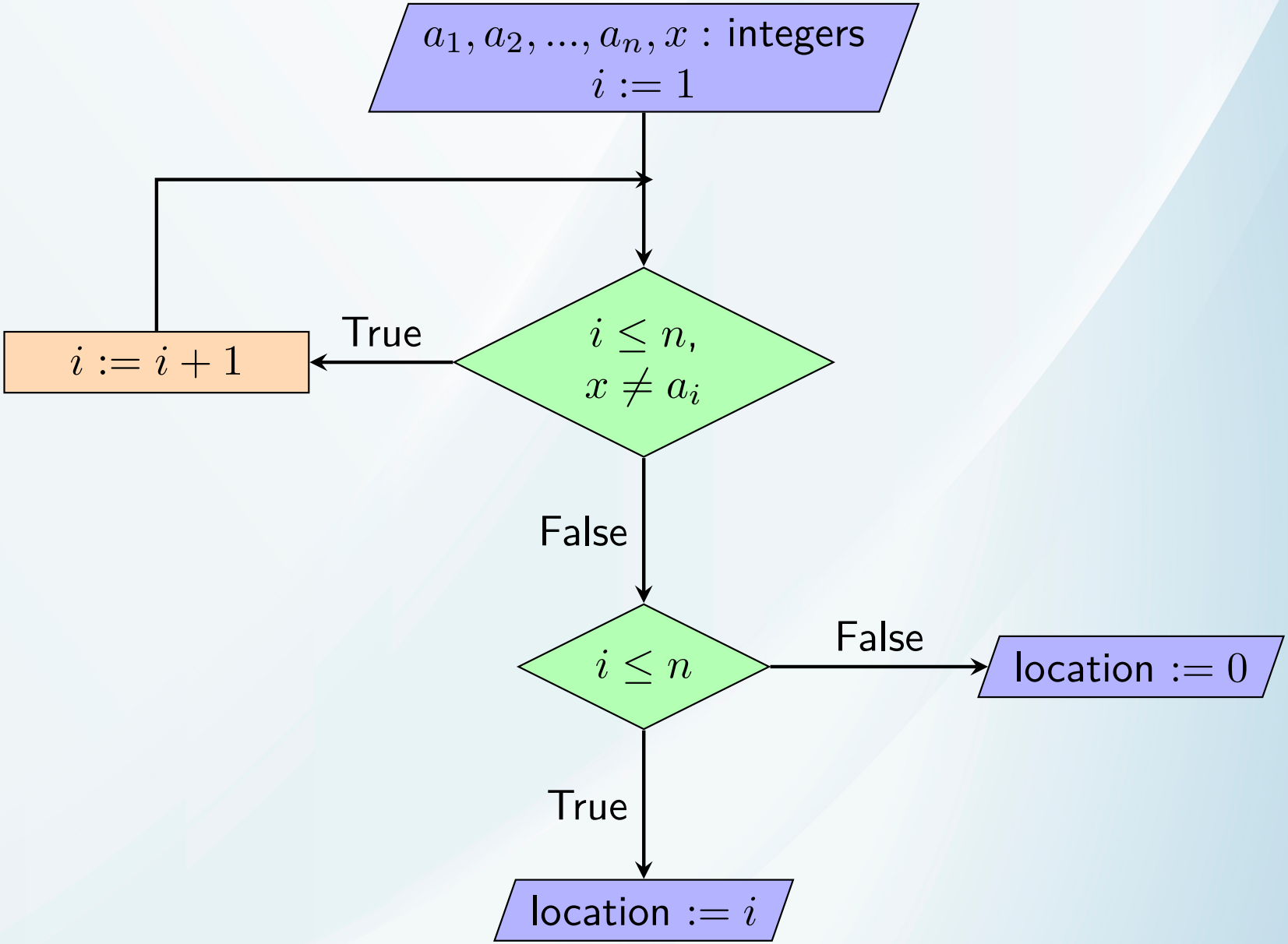
Linear Search

Input: A sequence of distinct integers $a_1, a_2, \ldots, a_n$, and an integer x.
Output: The location of x in the sequence (is 0 if x is not in the sequence).

Algorithm: Compare x successively to each term of the sequence until a match is found.

Procedure LinearSearch($a_1, a_2, \ldots, a_n$: distinct integers, x: integer)
while $(i \leq n)$ and $(x \neq a_i)$
  $i := i + 1$
if $i \leq n$ **then** location $:= i$
else location $:= 0$
return location

FPTU-QN


$a_1, a_2, ..., a_n, x$: integers
$i := 1$
$i \leq n,$
$x \neq a_i$
True
$i := i + 1$
False
$i \leq n$
False
location $:= 0$
True
location $:= i$

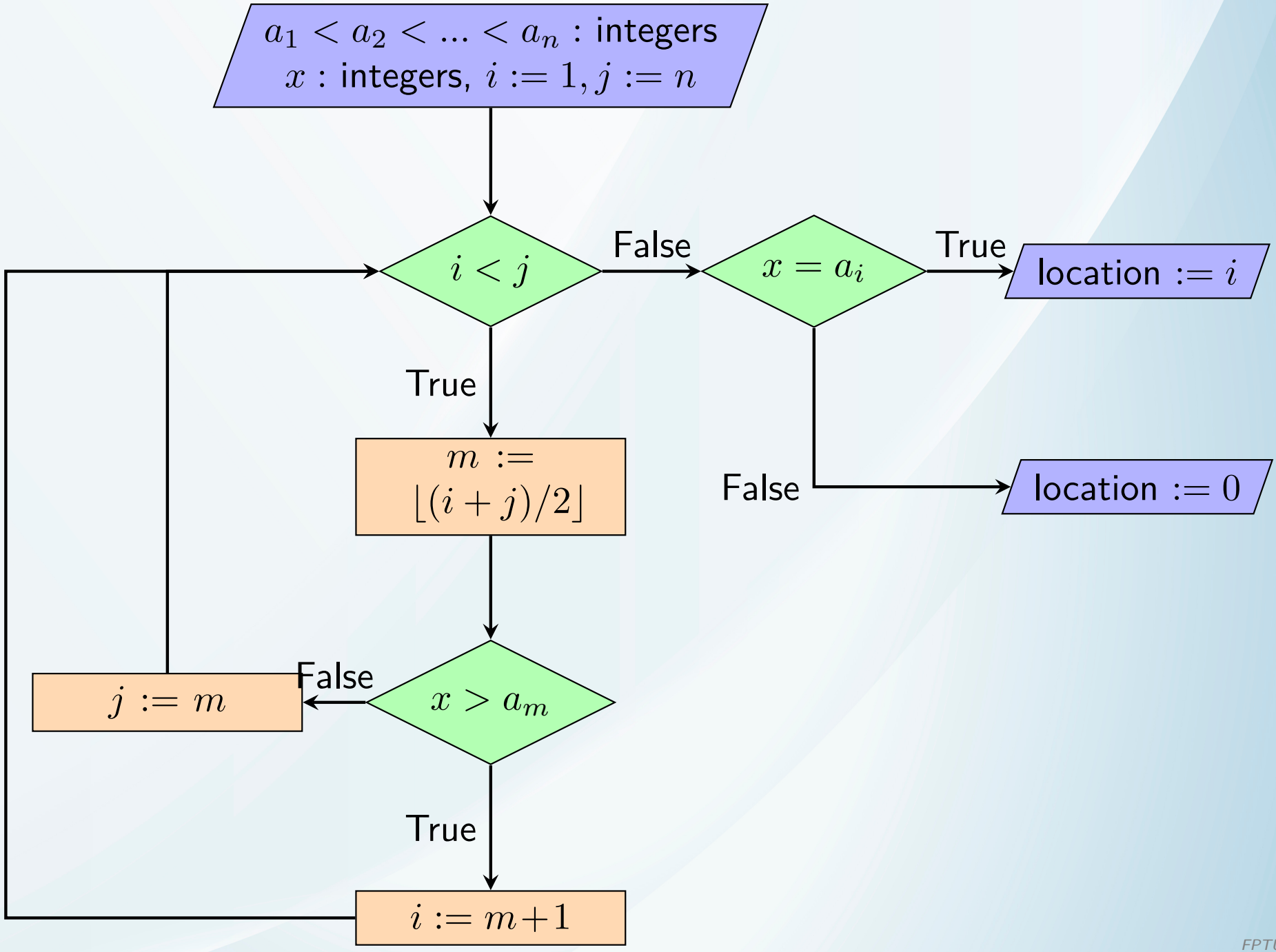
Binary Search

Input: An increasing sequence of integers $a_1 < a_2 < \cdots < a_n$ and an integer x.
Output: The location of x in the sequence (is 0 if x is not in the sequence).

Algorithm: Compare x to the element at the middle of the list, then restrict the search to either the sublist on the left or the sublist on the right.

Procedure BinarySearch($a_1 < a_2 < \cdots < a_n,\ x$: integer)
$i := 1,\ j := n$
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 $:= 0$
return location

FPTU-QN


$a_1 < a_2 < ... < a_n$: integers
x : integers, $i := 1, j := n$
$i < j$
False
$x = a_i$
True
location := i
True
$m := \lfloor (i + j)/2 \rfloor$
False
location := 0
$j := m$
False
$x > a_m$
True
$i := m+1$

FPTU-QN

Bubble Sort

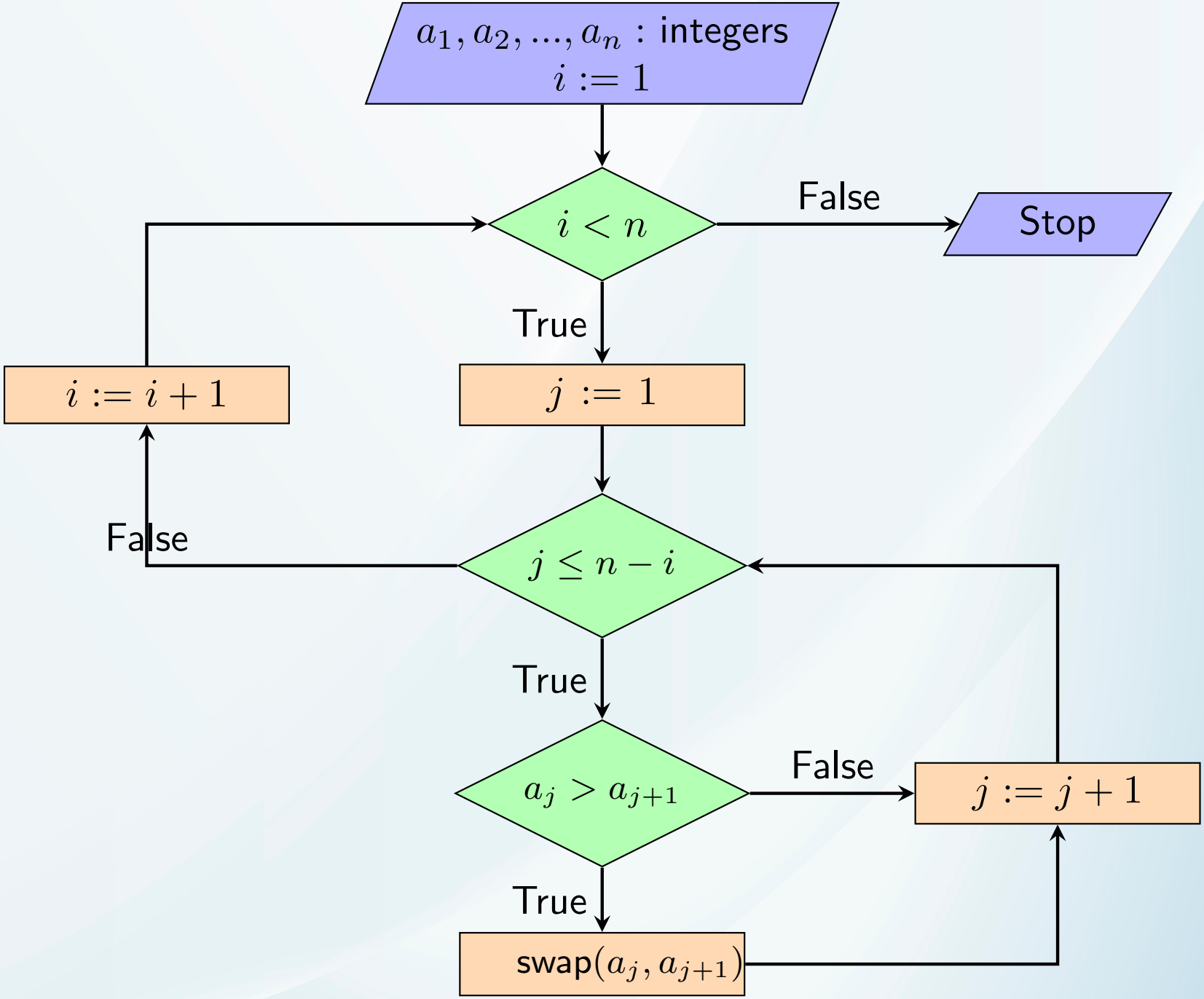
Input: A sequence of integers $a_1, a_2, \ldots, a_n$
Output: The sequence in the increasing order.

Algorithm:

1. Successively comparing two consecutive elements of the list to push the largest element to the bottom of the list.
2. Repeat the above step for the first $n-1$ elements of the list.

Procedure BubbleSort($a_1, a_2, \ldots, a_n$: integers)
for $i := 1$ **to** $n-1$
  for $j := 1$ to $n-i$
    if $a_j > a_{j+1}$ **then** swap(a_j, a_{j+1})

FPTU-QN


$a_1, a_2, ..., a_n$: integers
$i := 1$
$i < n$
False
Stop
True
$i := i + 1$
$j := 1$
False
$j \leq n - i$
True
$a_j > a_{j+1}$
False
$j := j + 1$
True
swap(a_j, a_{j+1})

Insertion Sort

Input: Sequence of integers $a_1, a_2, \ldots, a_n$
Output: The sequence in the increasing order.

Algorithm:

1. Sort the first two elements of the list.
2. Insert the third element to the list of the first two elements to get a list of 3 elements of increasing order.
3. Insert the fourth element to the list of the first three elements to get a list of 4 elements of increasing order.

$\vdots$

n. Insert the nth element to the list of the first $n-1$ elements to get a list of increasing order.

FPTU-QN

Insertion Sort (Cont')

Procedure InsertionSort($a_1, a_2, \ldots, a_n$: integers)
for $j := 2$ **to** n
begin
  $i := 1$
  while $a_j > a_i$
    $i := i + 1$
  $m := a_j$
  $k := j$
  while $k > i$
    $a_k := a_{k-1}$
    $k := k - 1$
  $a_i := m$
end

FPTU-QN

Greedy Change-Making Algorithm

Input: n cents
Output: The least number of coins using quarters ($= 25$ cents), dimes ($= 10$ cents), nickles ($= 5$ cents) and ($= 1$ cent).

- Greedy change-making algorithms are usually used to solve optimization problems: Finding out a solution to the given problem that either minimizes or maximizes the value of some parameter.
- Selecting the best choice at each step, instead of considering all sequences of steps that may lead to an optimal solution.
- Some problems:
  - Finding a route between two cities with smallest total mileage (number of miles that a person passed).
  - Determining a way to encode messages using the fewest bits possible.
  - Finding a set of fiber links between network nodes using the least amount of fiber.

Algorithm: Read textbook!

FPTU-QN

Upcoming ...

FPTU-QN

The Growth of Functions

In calculus, we learned following basic functions listed in the **increasing** order of their **complexity**:

$$1,\ \log n,\ n^k,\ a^n,\ n!$$

where $k > 0$ and $a > 1$. It means that

$$\lim_{n\to+\infty} \frac{\log n}{1} = +\infty,\ \lim_{n\to+\infty} \frac{n^k}{\log n} = +\infty,\ \lim_{n\to+\infty} \frac{a^n}{n^k} = +\infty,\ \lim_{n\to+\infty} \frac{n!}{a^n} = +\infty.$$

Ordering of Basic Functions by Growth

$$1,\ \log n,\ \sqrt[3]{n},\ \sqrt{n},\ n,\ n\log n,\ n^2,\ n^3,\ 2^n,\ 3^n,\ n!.$$

Question. Estimate the complexity (the growth) of functions like

$$f(n) = \frac{(n+2)(n\log n + n!)}{3^n + (\log n)^2}$$

via simpler functions.

Big-O

The function $f(x)$ is called **big-O** of $g(x)$, write $f(x)$ is $O(g(x))$, if there exists a constant C and k such that

$$|f(x)| \leq C|g(x)|$$

whenever $x > k$.

Note.

1. The definition that $f(x)$ is $O(g(x))$ says that $f(x)$ grows slower that some fixed multiple of $g(x)$ as x grows without bound.
2. The fact that $f(x)$ is $O(g(x))$ is sometimes written $f(x) = O(g(x))$. However, the equals sign in this notation does not represent a genuine equality.
3. However, it is acceptable to write $f(x) \in O(g(x))$ because $O(g(x))$ represents the set of functions that are $O(g(x))$.

FPTU-QN

Input Complexities Ordered from Smallest to Largest

1. Constant Complexity: $\mathrm{O}(1)$.
2. Logarithmic Complexity: $\mathrm{O}(\log n)$.
3. Radical complexity: $\mathrm{O}(\sqrt{n})$.
4. Linear Complexity: $\mathrm{O}(n)$.
5. Linearithmic Complexity: $\mathrm{O}(n \log n)$.
6. Quadratic complexity: $\mathrm{O}(n^2)$.
7. Cubic complexity: $\mathrm{O}(n^3)$.
8. Exponential complexity: $\mathrm{O}(b^n)$ where $b > 1$.
9. Factorial complexity: $\mathrm{O}(n!)$.

FPTU-QN

Examples

Example 1. Show that $f(x) = 2x^3 + 3x$ is $O(x^3)$.
Solution. With $C = 3$ and $k = 2$, we have $x^3 > 3x$ whenever $x > 2$ which implies

$$|2x^3 + 3x| = 2x^3 + 3x < 3x^3 = 3|x^3|$$

whenever $x > 2$.

Example 2. Show that $f(x) = 5x^2 - 10000x + 7$ is $O(x^2)$.
Solution. We have to be a little more careful about negative values here because of the $-10000x$ term, but as long as we take $k \geq 2000$, we will not have any negative values since the $5x^2$ term is larger there. We have

$$\begin{aligned} |5x^2 - 10000x + 7| &= 5x^2 - 10000x + 7 \\ &= 5x^2 + 7x^2 \\ &= |12x^2| = 12|x^2|. \end{aligned}$$

FPTU-QN

Student's Work

1. Show that $x^5 - 2x^2 + 7$ is $O(x^5)$.
2. Show that $f(x) = x^2$ is not $O(\sqrt{x})$.
3. Prove that $x^5 - 2x^2 + 7$ is not $O(x^4)$.

FPTU-QN

Big-O for Polynomials

Theorem

Any degree-n polynomial, $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x^1 + a_0$ is $O(x^n)$.

Note.

1. Let $p(x)$ be a polynomial of degree n. Then, $p(x)$ is not big-O of any power of x that is less than n.
2. Any constant value is $O(1)$.

Example.

1. The function $p(x) = 2x^3 + 10x$ is not in $O(x^2)$.
2. The function $f(x) = 2x^3 + 10x$ is $O(x^4)$.

Question. Find the smallest integer n such that

1. $x^3 + x^5 \log x$ is $O(x^n)$.
2. $x^5 + x^3(\log x)^4$ is $O(x^n)$.

FPTU-QN

Properties of Big-O

Theorem

Assume that $f(x)$ is $O(F(x))$ and $g(x)$ is $O(G(x))$. Then,

1. $cf(x)$ is $O(F(x))$ where c is a constant.
2. $f(x) + g(x)$ is $O(\max\{|F(x)|, |G(x)|\})$.
3. $f(x) \cdot g(x)$ is $O(F(x) \cdot G(x))$.

Example. Let $f(x) = 2x^2$ and $g(x) = 4x$. Then, $f(x)$ is $O(x^2)$ and $g(x)$ is $O(x)$ which implies $f(x) + g(x)$ is $O(\max\{x^2, x\}) = O(x^2)$.

Theorem

Let f, g, h be functions where $f(x)$ is $O(g(x))$ and $g(x)$ is $O(h(x))$. Then, $f(x)$ is $O(h(x))$.

FPTU-QN

Big-theta

Big-O estimates only give upper-bounds; to give sharper bounds, we use big-theta notation.

The function $f(x)$ is called **big-theta** of $g(x)$, write "$f(x)$ is $\Theta(g(x))$", if $f(x)$ is $O(g(x))$ and $g(x)$ is $O(f(x))$. In the other words, $f(x)$ is $\Theta(g(x))$ if there are constants $C_1, C_2 > 0$ such that

$$C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$$

whenever $x > k$.

Example. Show that $3x^2 + 8x \log x$ is $\Theta(x^2)$.
Solution. Since $0 \leq 8x \log x \leq 8x^2$, it follows that $3x^2 + 8x \log x \leq 11x^2$ for $x > 1$. Consequently, $3x^2 + 8x \log x$ is $O(x^2)$ and x^2 is $O(3x^2 + 8x \log x)$. Hence, $3x^2 + 8x \log x$ is $\Theta(x^2)$.

FPTU-QN

Question

1. Show that $f(x) = 2x^3 + x^2 + 3$ is $\Theta(x^3)$.
2. Is the function $f(x) = x^2 \log x + 3x + 1$ big-theta of x^3?
3. Show that $f(x) = \left\lfloor \frac{x}{2} \right\rfloor$ is $\Theta(x)$.

FPTU-QN

Upcoming ...

FPTU-QN

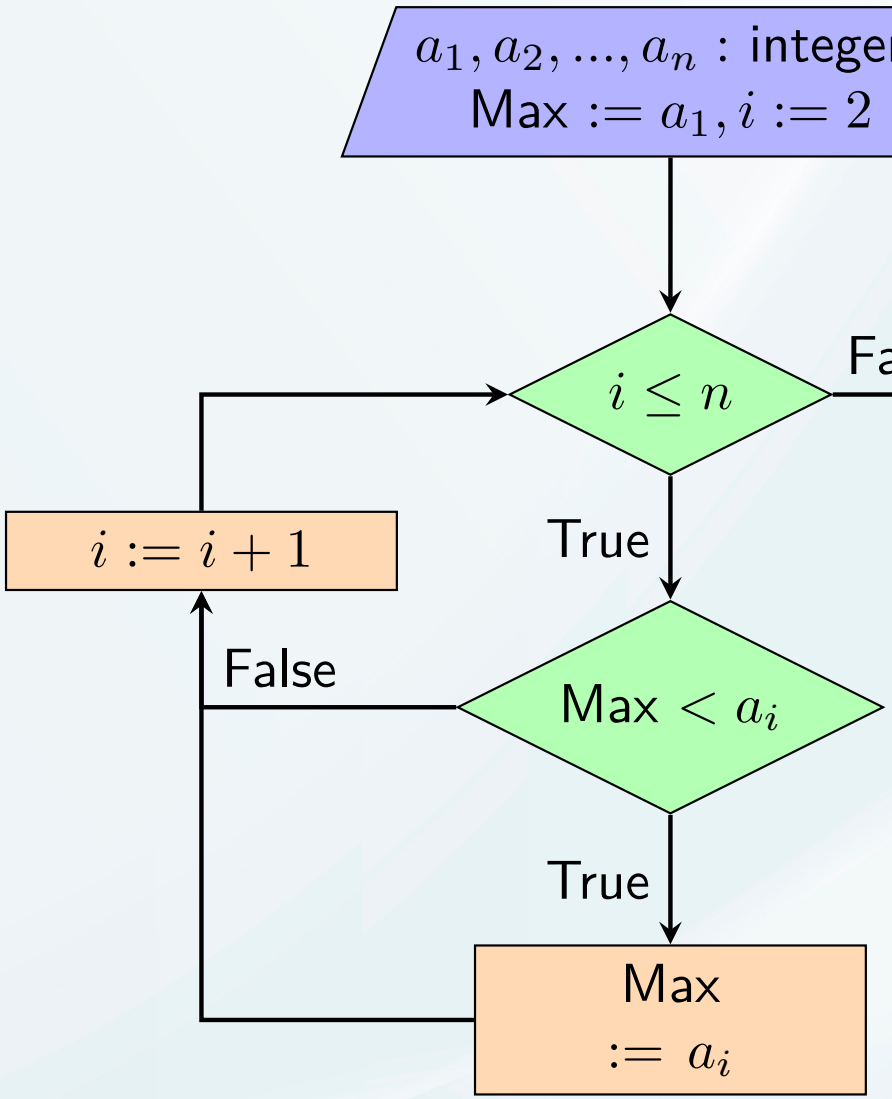
Complexity of Algorithms

- **Space complexity**: Computer memory required to run the algorithm.
- **Time complexity**: Time required to run the algorithm. Time complexity can be expressed in terms of the number of operations used by the algorithm. Those operations can be comparisons or basic arithmetic operations.

In this lecture, we analyze times complexity of some algorithms studied in previous sections.

FPTU-QN

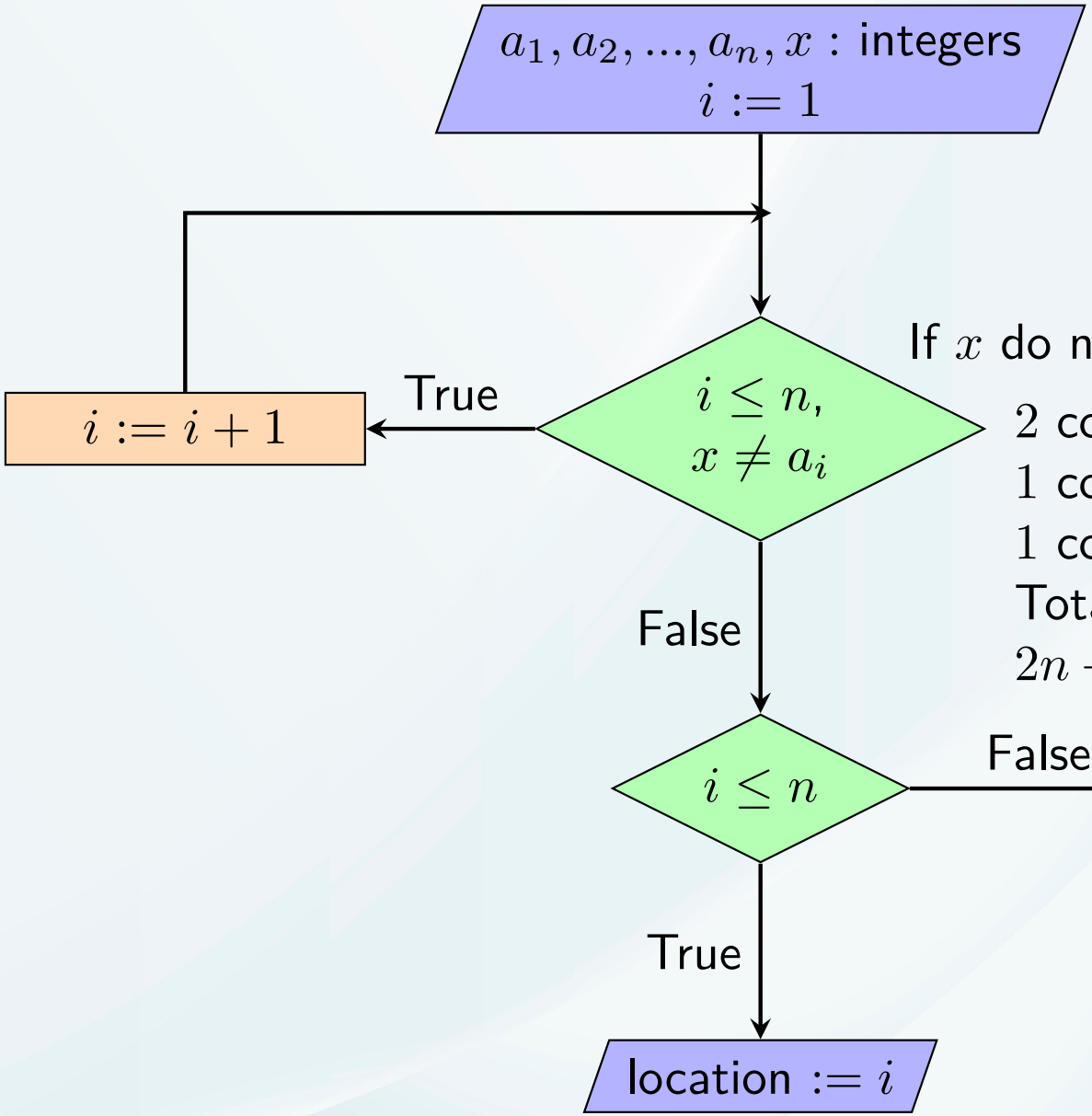
Complexity of Algorithm of Finding Maximum Element



Number of loops: $n - 1$
Number of comparisons in each loop: 2
Number of comparisons to exit the loop: 1
Total number of comparisons:
$2(n-1) + 1 = 2n - 1 = \mathrm{O}(n)$.

FPTU-QN

Complexity of Linear Search Algorithm



If x do not appear in the list: n loops.

2 comparisons in each loop.

1 comparison to exit the loop.

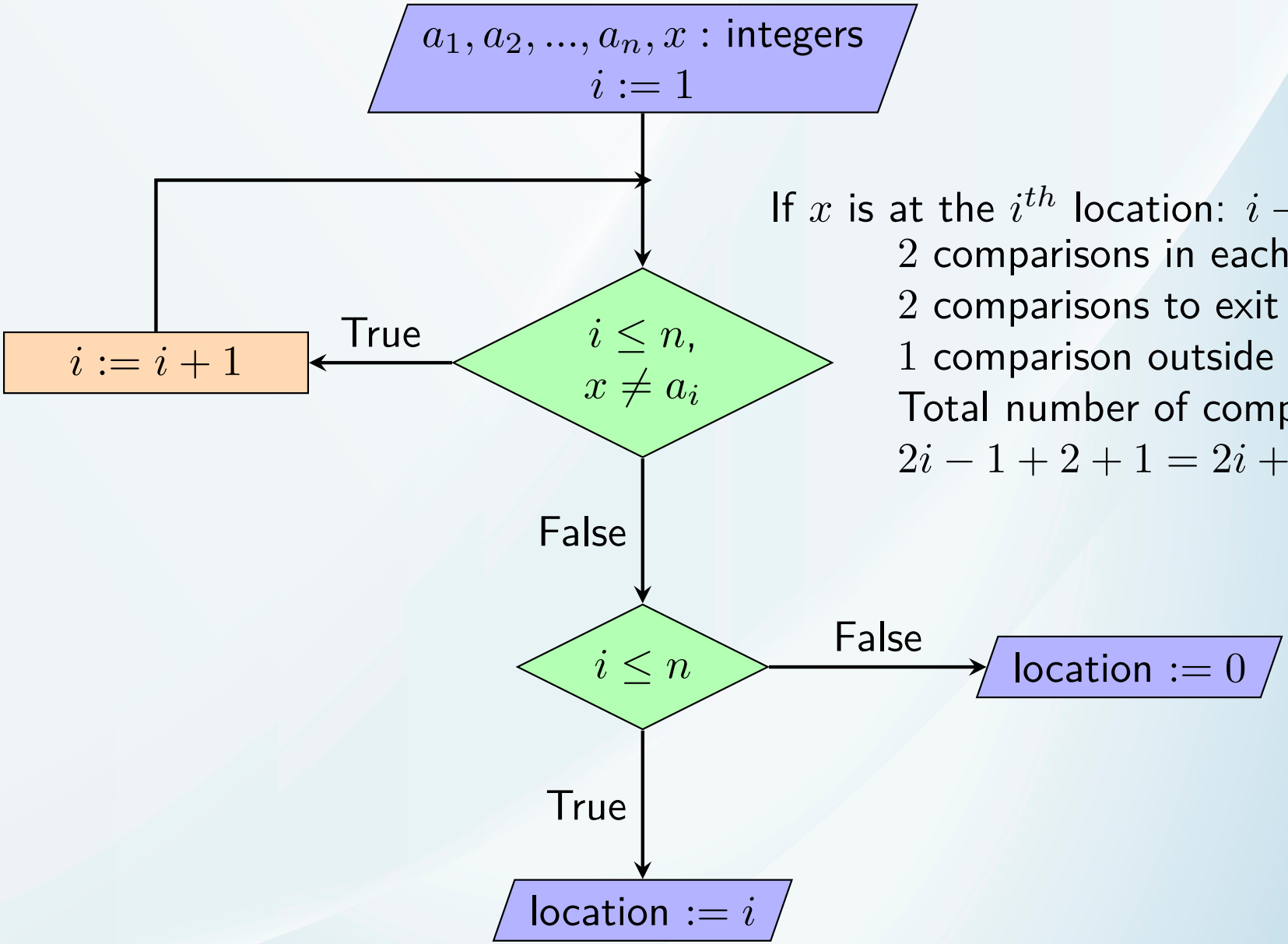
1 comparison outside the loop.

Total number of comparisons:

$2n + 1 + 1 = 2n + 2 = O(n)$.

FPTU-QN

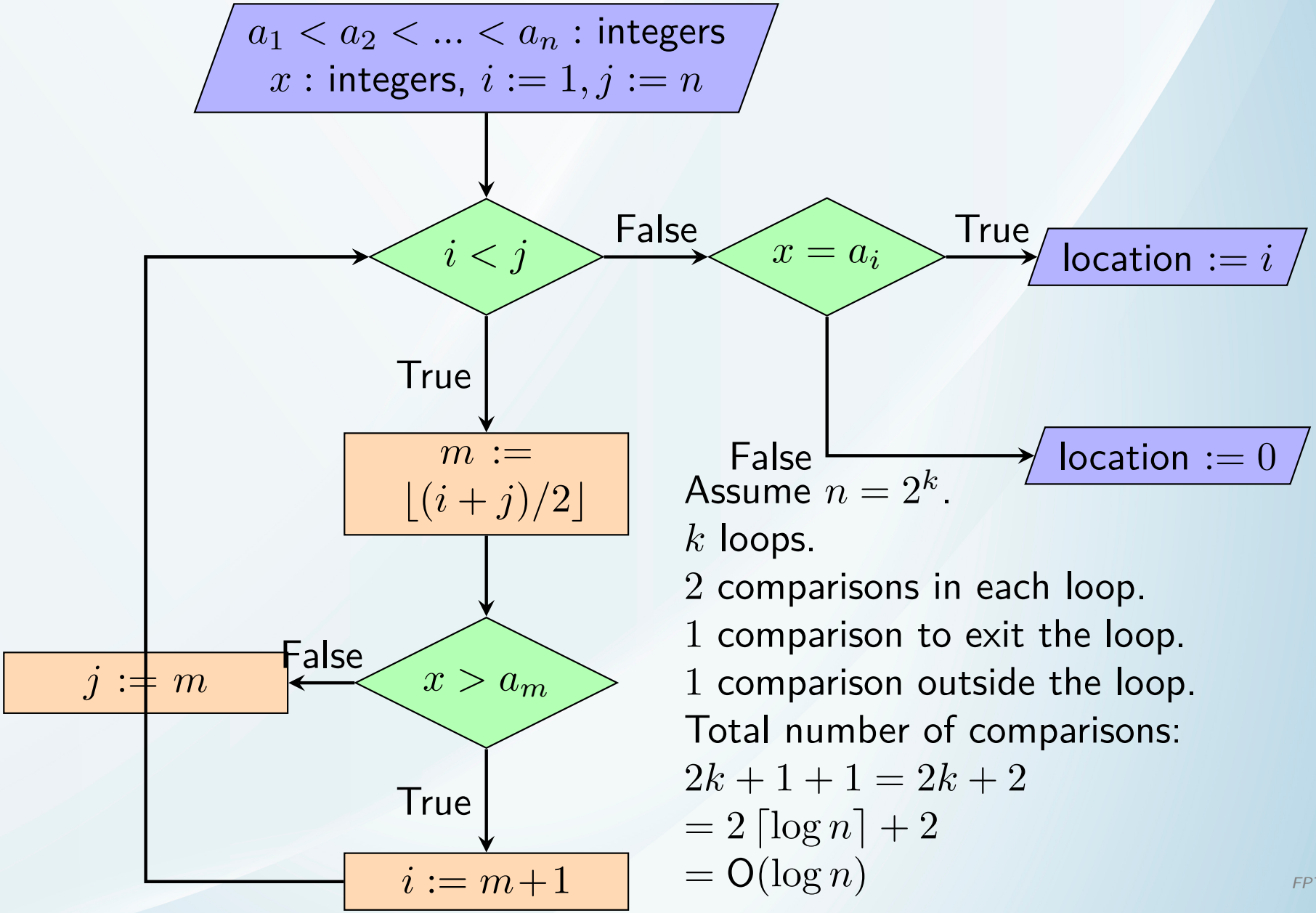
Complexity of Linear Search Algorithm



If x is at the i^{th} location: $i - 1$ loops.
2 comparisons in each loop.
2 comparisons to exit the loop.
1 comparison outside the loop.
Total number of comparisons:
$2i - 1 + 2 + 1 = 2i + 1$.

FPTU-QN

Complexity of Binary Search Algorithm



Assume $n = 2^k$.

k loops.

2 comparisons in each loop.

1 comparison to exit the loop.

1 comparison outside the loop.

Total number of comparisons:

$2k + 1 + 1 = 2k + 2$

$= 2\lceil \log n \rceil + 2$

$= O(\log n)$

FPTU-QN

Upcoming ...

FPTU-QN

The Integers and Division

Let a, b be integers with $a \neq 0$. We say the integer a divides b if there is an integer m such that $b = ma$.

Note. If a divides b, we also write:

- b is divisible by a.
- b is a multiple of a.
- a is a factor of b.
- $a|b$.

Theorem

Let a, b, c be integers. Then

- If $a|b$ and $a|c$ then $a|(b + c)$.
- If $a|b$ then $a|bc$ for all c.
- If $a|b$ and $b|c$ then $a|c$.

FPTU-QN

The Division Algorithm

- Let a be an integer and d be a positive integer. Then, there are unique integers q and r with $0 \leq r < d$ such that $a = qd + r$.
- In this division algorithm, a is called the dividend, d is the divisor, q is the quotient and r is the remainder. We write,

$$q = a \text{ div } d, \quad r = a \text{ mod } d.$$

Question. Find the remainder and the quotient of the division:

1. -23 is divided by 7.
2. -125 is divided by 11.

FPTU-QN

Modular Arithmetic

Let a, b be integers and m be a positive integer. We say a is **congruent** to b **modulo** m is the have the same remainders when being divided by m. We use notation $a \equiv b \mod m$. If they are not congruent, we write $a \not\equiv b \mod m$.

Properties

1. $a \equiv b \mod m \Leftrightarrow a - b \equiv 0 \mod m \Leftrightarrow a = b + km$ for some integer k.
2. If $a \equiv b \mod m$ and $c \equiv d \mod m$ then $a + c \equiv b + d \mod m$ and $ac \equiv bd \mod m$.

FPTU-QN

Applications of Congruences

Pseudorandom numbers

Pseudorandom numbers can be generated using **Linear congruential method**:

- x_0 is given, and
- $x_n = ax_{n-1} + c \mod m,\ n = 2, 3, 4, \ldots$
  where m is called **modulus**, a is the **multiplier**, c is the **increment** and x_0 is the **seed**.

Cryptography

0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	G	H	I	J	K	L	M
13	14	15	16	17	18	19	20	21	22	23	24	25
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Caesar's cipher.

$$f(p) = p + 3 \mod 26.$$

FPTU-QN

Upcoming ...

FPTU-QN

Primes and Greatest Common Divisors

- A positive integer p greater than 1 is called a **prime number** if the only prime factors of p are 1 and p.
- An integer greater than 1 that is not prime is called **composite number**.

The Fundamental Theorem of Arithmetic

Any integer greater than 1 can be written uniquely as a product of powers of distinct primes.

Theorem

There are infinitely many primes.

FPTU-QN

Let a and b be two integers, not both 0. The greatest integer d that is a divisor of both a and b is called **greatest common divisor** of a and b, denoted by $\gcd(a, b)$.

Let a and b be two positive integers. The smallest positive integer d that is divisible by both a and b is called the **least common multiple** of a and b, denoted by $\mathrm{lcm}(a, b)$.

Find gcd and lcm

To find $\gcd$ and lcm of a and b, we write a, b as products of powers of distinct primes:

$$a = p_1^{a_1} p_2^{a_2} \cdots p_n^{a^n},$$
$$b = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n}.$$

Then, we obtain

- $\gcd(a, b) = p_1^{\min(a_1, b_1)} p_2^{\min(a_2, b_2)} \cdots p_n^{\min(a_n, b_n)}$.
- $\mathrm{lcm}(a, b) = p_1^{\max(a_1, b_1)} p_2^{\max(a_2, b_2)} \cdots p_n^{\max(a_n, b_n)}$.

FPTU-QN

Pairwise Relatively Prime

Theorem

Let a, b be positive integers. Then,

$$\mathbf{ab} = \gcd(\mathbf{a}, \mathbf{b}) \cdot \mathrm{lcm}(\mathbf{a}, \mathbf{b}).$$

1. Two integers a, b are called **relatively prime** if $\gcd(a, b) = 1$.
2. A set of integers are called **pairwise relatively prime** if any two integers in the set are relatively prime.

Question. Which sets are pairwise relatively prime?

1. $\{13,\ 24,\ 49\}$.
2. $\{14,\ 23,\ 35,\ 61\}$.

FPTU-QN

Upcoming ...

FPTU-QN

Representations of Integers

Let b be an integer greater than 1 and n be a positive integer. Then, n can be expressed uniquely in the form

$$n = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_0$$

where $a_k, \ldots, a_1, a_0$ are nonnegative integers and less than b. This representation is called **base b expansion of n**, and denoted by $n = (a_k a_{k-1} \ldots a_0)_b$

Some Important Representations

1. **Binary** expansion: $0,\ 1$.
2. **Octal** expansion: $0,\ 1,\ 2, \ldots,\ 7$.
3. **Hexadecimal** expansion: $0,\ 1,\ 2, \ldots,\ 9,\ A,\ B,\ C,\ D,\ E,\ F$.

Example.

1. Find the binary expansion of 35.
2. Find the hexadecimal expansion of $(132)_5$.

FPTU-QN

Some representations of the Integers 0 through 15

Decimal	0	1	2	3	4	5	6	7
Hexadecimal	0	1	2	3	4	5	6	7
Octal	0	1	2	3	4	5	6	7
Binary	0	1	10	11	100	101	110	111
Decimal	8	9	10	11	12	13	14	15
Hexadecimal	8	9	A	B	C	D	E	F
Octal	10	11	12	13	14	15	16	17
Binary	1000	1001	1010	1011	1100	1101	1110	1111

FPTU-QN

Algorithms for Integer Operations

Let a and b be in the binary expansions.

$$a = (a_{n-1}a_{n-2}\dots a_0)_2, \quad b = (b_{n-1}b_{n-2}\dots b_0)_2.$$

Addition Algorithm

Procedure Addition (a, b)
$c := 0$
for $j := 0$ **to** $n-1$
    $d := \lfloor (a_j + b_j + c)/2 \rfloor$
    $s_j := a_j + b_j + c - 2d$
    $c := d$
$s_n := c$
return $(s_0, s_1, ..., s_n)$

Multiplication Algorithm

Procedure Multiplication (a, b)
for $j := 0$ **to** $n-1$
    if $b_j = 1$ **then** $c_j := a$ shifted j places
    else $c_j := 0$
$p := 0$
for $j := 0$ **to** $n-1$
    $p := p + c_j$
return p

FPTU-QN

Euclidean Algorithm

Theorem

Let $a > b$ be positive integers. Put $r = a \mod b$. Then,

$$\gcd(a, b) = \gcd(b, r).$$

GCD Algorithm

Procedure GCD(a, b: positive integers)
$x := a$
$y := b$
while $y \neq 0$
    $r := x \mod y$
    $x := y$
    $y := r$
Print(x)

Modular Exponentiation Algorithm

Problem. Let b and m be positive integers. Compute $b^n \mod m$.

Example. Compute $3^{71} \mod 13$

Procedure ModExp(b, m: positive integers, $n = (a_k, \ldots, a_0)_2$)
$x := 1$
power $:= b \mod m$
for $i := 0$ **to** k
  if $a_i = 1$ **then** $x := (x * \text{power}) \mod m$
  power $:=$ power $*$ power $\mod m$
Print(x)

FPTU-QN

Thank you!

Call it a day

"Arithmetic is numbers you squeeze from your head to your hand to your pencil to your paper till you get the answer."

Carl Sandburg

FPTU-QN