# Decrease-and-Conquer

Murat Osmanoglu

# Decrease-and-Conquer

- for a given instance of a problem, take advantage of relationship between its solution and solution of its smaller instance

# Decrease-and-Conquer

- for a given instance of a problem, take advantage of relationship between its solution and solution of its smaller instance
  - reduce problem instance to its smaller instance
  - solve the smaller instance
  - extend the solution of smaller instance to obtain a solution for the original problem

# Decrease-and-Conquer

- for a given instance of a problem, take advantage of relationship between its solution and solution of its smaller instance
  - reduce problem instance to its smaller instance
  - solve the smaller instance
  - extend the solution of smaller instance to obtain a solution for the original problem
- three variations :
  - decrease by a constant (usually 1),
  - decrease by a constant factor (usually 2)
  - decrease by a variable size

# Decrease-and-Conquer

Decrease-by-a-Constant
(Exponentiation Problem)

# Decrease-and-Conquer

## Decrease-by-a-Constant
(Exponentiation Problem)

- Given a nonzero number a and a nonnegative integer n, compute $a^n$

# Decrease-and-Conquer

## Decrease-by-a-Constant
(Exponentiation Problem)

- Given a nonzero number a and a nonnegative integer n, compute $a^n$
- the formula $\boldsymbol{a^n = a^{n-1}.a}$ can be used to obtain the relationship between an instance of size n and an instance of size n – 1

# Decrease-and-Conquer

## Decrease-by-a-Constant
(Exponentiation Problem)

- Given a nonzero number a and a nonnegative integer n, compute $a^n$
- the formula $\boldsymbol{a^n = a^{n-1}.a}$ can be used to obtain the relationship between an instance of size n and an instance of size n – 1

$$f(n) = \begin{cases} f(n-1).a & if \;\; n > 0 \\ 1 & if \;\; n = 0 \end{cases}$$

# Decrease-and-Conquer

## Decrease-by-a-Constant
## (Exponentiation Problem)

- Given a nonzero number a and a nonnegative integer n, compute $a^n$
- the formula $\boldsymbol{a^n = a^{n-1}.a}$ can be used to obtain the relationship between an instance of size n and an instance of size n – 1

$$f(n) = \begin{cases} f(n-1).a & if \;\; n > 0 \\ 1 & if \;\; n = 0 \end{cases}$$

## Decrease-by-a-Constant-Factor
## (Exponentiation Problem)

# Decrease-and-Conquer

## Decrease-by-a-Constant
## (Exponentiation Problem)

- Given a nonzero number a and a nonnegative integer n, compute $a^n$
- the formula $\boldsymbol{a^n = a^{n-1}.a}$ can be used to obtain the relationship between an instance of size n and an instance of size n - 1

$$f(n) = \begin{cases} f(n-1).a & if \;\; n > 0 \\ 1 & if \;\; n = 0 \end{cases}$$

## Decrease-by-a-Constant-Factor
## (Exponentiation Problem)

- the formula $\boldsymbol{a^n = \left(a^{n/2}\right)^2}$ can be used to obtain the relationship between an instance of size n and an instance of half of its size

# Decrease-and-Conquer

## Decrease-by-a-Constant (Exponentiation Problem)

- Given a nonzero number a and a nonnegative integer n, compute $a^n$
- the formula $\boldsymbol{a^n = a^{n-1}.a}$ can be used to obtain the relationship between an instance of size n and an instance of size n – 1

$$f(n) = \begin{cases} f(n-1).a & if \;\; n > 0 \\ 1 & if \;\; n = 0 \end{cases}$$

## Decrease-by-a-Constant-Factor (Exponentiation Problem)

- the formula $\boldsymbol{a^n = \left(a^{n/2}\right)^2}$ can be used to obtain the relationship between an instance of size n and an instance of half of its size

$$f(n) = \begin{cases} \left(a^{n/2}\right)^2 & if\ n\ is\ even \\ \left(a^{(n-1)/2}\right)^2.a & if\ n\ is\ odd \\ 1 & if\ n = 0 \end{cases}$$

# Decrease-and-Conquer

Decrease-by-a-Constant-Factor
(Exponentiation Problem)

- Given n coins, all the same except for one fake coin that is lighter than the others, and a balance scale allowing us to compare any two piles of coins, find the lighter coin with the minimum possible number of weighs

# Decrease-and-Conquer

Decrease-by-a-Constant-Factor
(Exponentiation Problem)

- Given n coins, all the same except for one fake coin that is lighter than the others, and a balance scale allowing us to compare any two piles of coins, find the lighter coin with the minimum possible number of weighs
  - divide n coins into two piles of $\lfloor n/2 \rfloor$, leave one coin aside if n is odd, and put two piles on the scale
  - if the piles weigh same, the extra coin must be fake
  - otherwise, proceed with the lighter pile

# Decrease-and-Conquer

Decrease-by-a-Constant-Factor
(Exponentiation Problem)

- Given n coins, all the same except for one fake coin that is lighter than the others, and a balance scale allowing us to compare any two piles of coins, find the lighter coin with the minimum possible number of weighs
  - divide n coins into two piles of $\lfloor n/2 \rfloor$, leave one coin aside if n is odd, and put two piles on the scale
  - if the piles weigh same, the extra coin must be fake
  - otherwise, proceed with the lighter pile

Decrease-by-Variable-Size
(Euclid's Algorithm)

- Given two integers m and n, find the largest number dividing both of them

# Decrease-and-Conquer

## Decrease-by-a-Constant-Factor
(Exponentiation Problem)

- Given n coins, all the same except for one fake coin that is lighter than the others, and a balance scale allowing us to compare any two piles of coins, find the lighter coin with the minimum possible number of weighs
  - divide n coins into two piles of $\lfloor n/2 \rfloor$, leave one coin aside if n is odd, and put two piles on the scale
  - if the piles weigh same, the extra coin must be fake
  - otherwise, proceed with the lighter pile

## Decrease-by-Variable-Size
(Euclid's Algorithm)

- Given two integers m and n, find the largest number dividing both of them
- the formula $\boldsymbol{gcd(m, n) = gcd(n, m \bmod n)}$ can be used to obtain the relationship between an instance of size m and an instance of size n decrease-by-a-variable-size,

# Decrease-and-Conquer

Decrease-by-a-Constant-Factor
(Exponentiation Problem)

- Given n coins, all the same except for one fake coin that is lighter than the others, and a balance scale allowing us to compare any two piles of coins, find the lighter coin with the minimum possible number of weighs
  - divide n coins into two piles of $\lfloor n/2 \rfloor$, leave one coin aside if n is odd, and put two piles on the scale
  - if the piles weigh same, the extra coin must be fake
  - otherwise, proceed with the lighter pile

Decrease-by-Variable-Size
(Euclid's Algorithm)

- Given two integers m and n, find the largest number dividing both of them
- the formula $\boldsymbol{gcd(m,n) = gcd(n, m\ mod\ n)}$ can be used to obtain the relationship between an instance of size m and an instance of size n decrease-by-a-variable-size, use the following recursive definition

$$f(m,n) = \begin{cases} f(n, m\ mod\ n) & if\ \ n > 0 \\ m & if\ \ n = 0 \end{cases}$$

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \leq a_2' \leq \cdots \leq a_n'$

# Decrease-by-a-Constant

## Sorting Problem (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \leq a_2' \leq \cdots \leq a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?

# Decrease-by-a-Constant

Sorting Problem
(Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \ldots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \ldots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \ldots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \ldots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \ldots, a_n \rangle$)**

**input :** a sequence of orderable elements
**output:** sorted sequence in nondecreasing order

```
for i = 2 to n
    temp ← a_i
    j ← i − 1
    while j ≥ 0 and a_j > temp
        a_{j+1} ← a_j
        j ← j-1
    a_{j+1} ← temp
```

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \leq a_2' \leq \cdots \leq a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \dots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp ← $a_i$
    j ← $i - 1$
    **while** $j \geq 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1} \leftarrow$ temp

SELECTION

# Decrease-by-a-Constant

**Sorting Problem**
**(Decrease-by-One)**

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \dots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp $\leftarrow a_i$
    j $\leftarrow i - 1$
    **while** $j \ge 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1} \leftarrow$ temp

i = 2

temp ← E

| S | E | L | E | C | T | I | O | N |
|---|---|---|---|---|---|---|---|---|

# Decrease-by-a-Constant

## Sorting Problem
(Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1{}', a_2{}', \dots, a_n{}' \rangle$ such that $a_1{}' \le a_2{}' \le \cdots \le a_n{}'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort**($\langle a_1, a_2, \dots, a_n \rangle$)

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp ← $a_i$
    j ← $i - 1$
    **while** $j \ge 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1}$ ← temp

i = 2

temp ← E

S E L E C T I O N

S

# Decrease-by-a-Constant

## Sorting Problem
(Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \dots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

```
for i = 2 to n
    temp ← a_i
    j ← i − 1
    while j ≥ 0 and a_j > temp
        a_{j+1} ← a_j
        j ← j-1
    a_{j+1} ← temp
```

i = 2

temp ← E

S E L E C T I O N

S

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1{}', a_2{}', \dots, a_n{}' \rangle$ such that $a_1{}' \le a_2{}' \le \cdots \le a_n{}'$

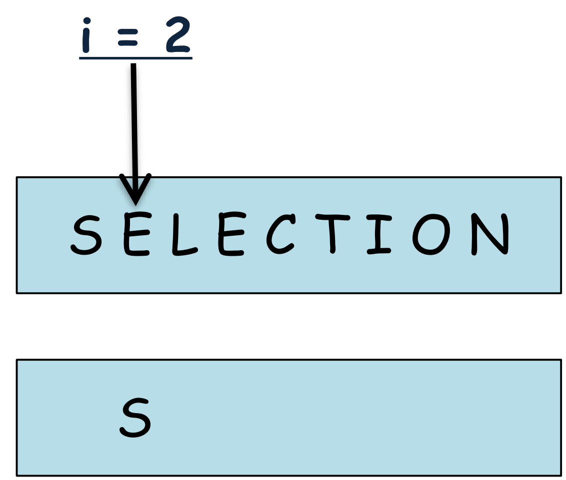
- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \dots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp ← $a_i$
    j ← $i - 1$
    **while** $j \ge 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1}$ ← temp

temp ← E



# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \leq a_2' \leq \cdots \leq a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \dots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
  temp $\leftarrow a_i$
  j $\leftarrow i - 1$
  **while** $j \geq 0$ and $a_j >$ temp
    $a_{j+1} \leftarrow a_j$
    j ← j-1
  $a_{j+1} \leftarrow$ temp

i = 2

temp ← E

S E L E C T I O N

E S

# Decrease-by-a-Constant

Sorting Problem
(Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort**($\langle a_1, a_2, \dots, a_n \rangle$)

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
  temp ← $a_i$
  j ← $i - 1$
  **while** $j \ge 0$ and $a_j >$ temp
    $a_{j+1} \leftarrow a_j$
    j ← j-1
  $a_{j+1} \leftarrow$ temp

i = 3

temp ← L

| S | E | L | E | C | T | I | O | N |

| E | S |

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \ldots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \ldots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \ldots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \ldots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort**($\langle a_1, a_2, \ldots, a_n \rangle$)

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp $\leftarrow a_i$
    j $\leftarrow i - 1$
    **while** $j \ge 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j $\leftarrow$ j-1
    $a_{j+1} \leftarrow$ temp

i = 3

temp ← L

S E L E C T I O N

E S

# Decrease-by-a-Constant

## Sorting Problem
(Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \ldots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \ldots, a_n' \rangle$ such that $a_1' \leq a_2' \leq \cdots \leq a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \ldots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \ldots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \ldots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp ← $a_i$
    j ← $i - 1$
    **while** $j \geq 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1}$ ← temp

i = 3

temp ← L

S E L E C T I O N

E S

# Decrease-by-a-Constant

## Sorting Problem
(Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

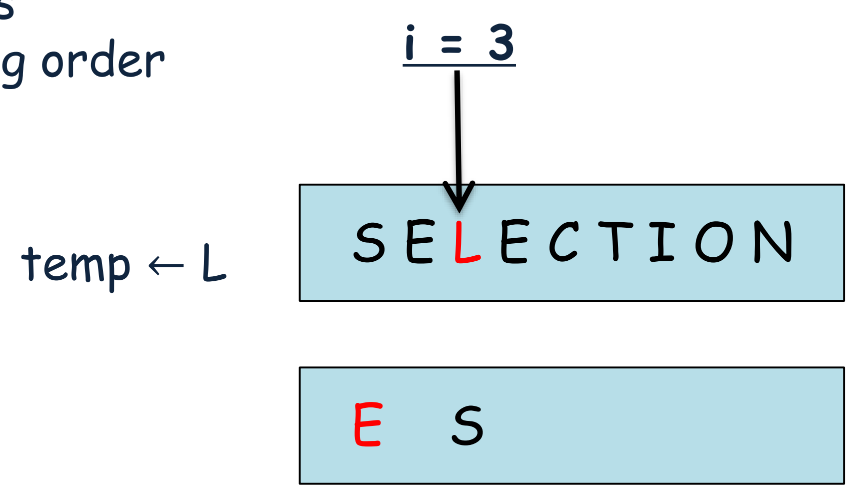
  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort**($\langle a_1, a_2, \dots, a_n \rangle$)

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp $\leftarrow a_i$
    j $\leftarrow i - 1$
    **while** $j \ge 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j $\leftarrow$ j-1
    $a_{j+1} \leftarrow$ temp



# Decrease-by-a-Constant

## Sorting Problem (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \leq a_2' \leq \cdots \leq a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \dots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp ← $a_i$
    j ← $i - 1$
    **while** $j \geq 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1} \leftarrow$ temp

i = 3

temp ← L

| S E L E C T I O N |
|---|

| E L S |
|---|

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort(**$\langle a_1, a_2, \dots, a_n \rangle$**)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp ← $a_i$
    j ← $i - 1$
    **while** $j \ge 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1} \leftarrow$ temp

i = 4

temp ← E

S E L E C T I O N

E L S

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

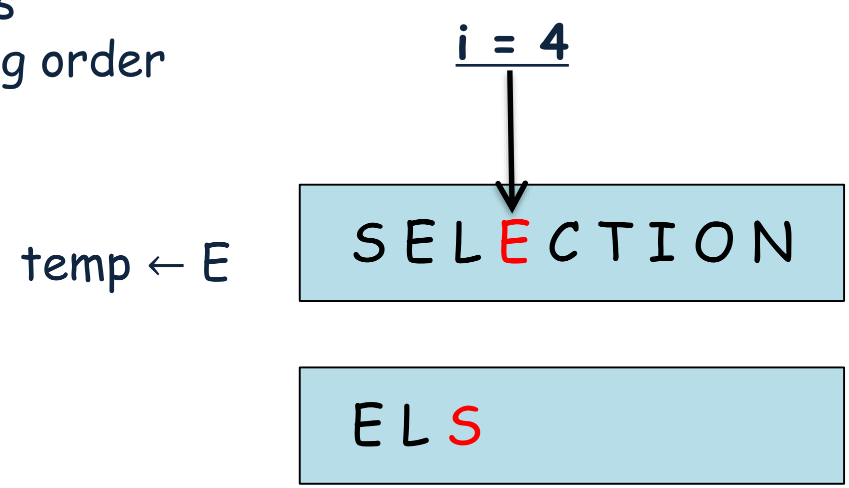
  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort(**$\langle a_1, a_2, \dots, a_n \rangle$**)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp ← $a_i$
    j ← $i - 1$
    **while** $j \ge 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1} \leftarrow$ temp



# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort**($\langle a_1, a_2, \dots, a_n \rangle$)

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp ← $a_i$
    j ← $i - 1$
    **while** $j \ge 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1}$ ← temp

i = 4

temp ← E

S E L E C T I O N

E L   S

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1{}', a_2{}', \dots, a_n{}' \rangle$ such that $a_1{}' \leq a_2{}' \leq \cdots \leq a_n{}'$

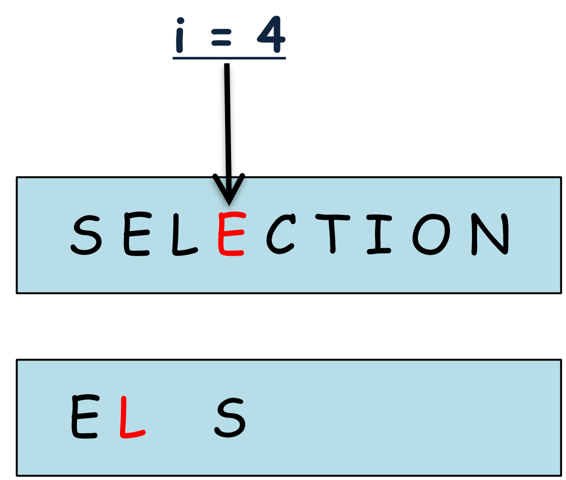
- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \dots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp $\leftarrow a_i$
    j $\leftarrow i - 1$
    **while** $j \geq 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j $\leftarrow$ j-1
    $a_{j+1} \leftarrow$ temp

temp ← E



# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \dots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

```
for i = 2 to n
    temp ← a_i
    j ← i − 1
    while j ≥ 0 and a_j > temp
        a_{j+1} ← a_j
        j ← j-1
    a_{j+1} ← temp
```

i = 4

temp ← E

| S | E | L | E | C | T | I | O | N |
|---|---|---|---|---|---|---|---|---|

| E | | L | S | | | | | |
|---|---|---|---|---|---|---|---|---|

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \le a_2' \le \cdots \le a_n'$

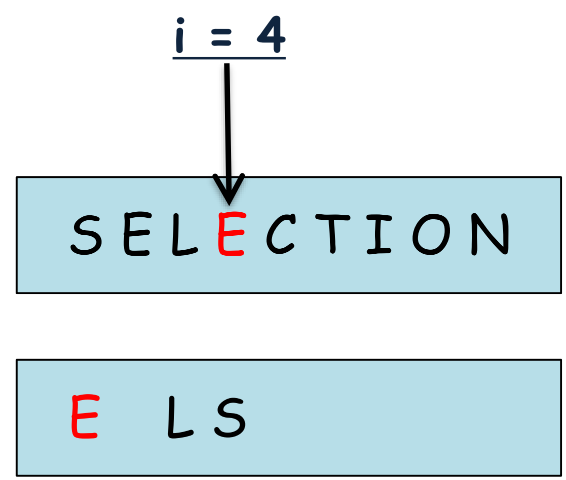
- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort**($\langle a_1, a_2, \dots, a_n \rangle$)

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
  temp $\leftarrow a_i$
  j $\leftarrow i - 1$
  **while** $j \ge 0$ and $a_j >$ temp
    $a_{j+1} \leftarrow a_j$
    j $\leftarrow$ j-1
  $a_{j+1} \leftarrow$ temp

temp $\leftarrow$ E



# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

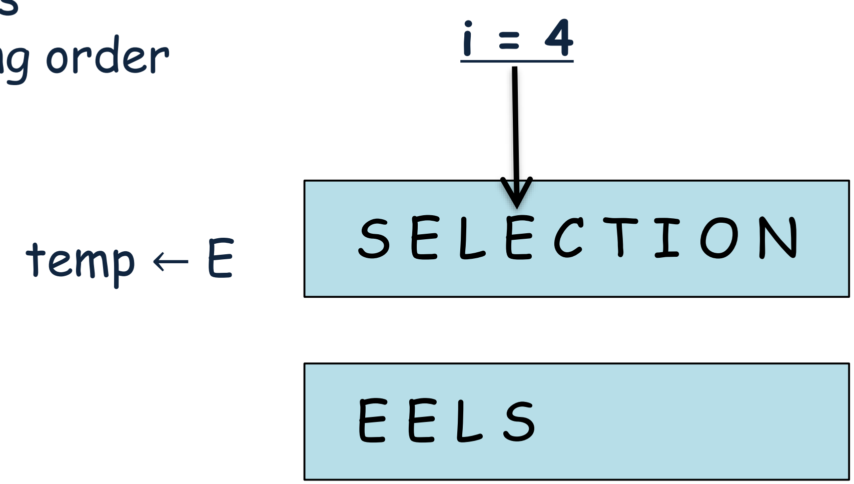
  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \leq a_2' \leq \cdots \leq a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort**($\langle a_1, a_2, \dots, a_n \rangle$)

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp $\leftarrow a_i$
    j $\leftarrow i - 1$
    **while** $j \geq 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j $\leftarrow$ j-1
    $a_{j+1} \leftarrow$ temp



# Decrease-by-a-Constant

## Sorting Problem
(Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \dots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2', \dots, a_n' \rangle$ such that $a_1' \leq a_2' \leq \cdots \leq a_n'$

- How can we use a solution for the sequence $\langle a_1, a_2, \dots, a_{n-1} \rangle$ to obtain a solution for the sequence $\langle a_1, a_2, \dots, a_n \rangle$ ?
  - just find an appropriate position to place $a_n$

**InsertionSort($\langle a_1, a_2, \dots, a_n \rangle$)**

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

```
for i = 2 to n
    temp ← a_i
    j ← i − 1
    while j ≥ 0 and a_j > temp
        a_{j+1} ← a_j
        j ← j-1
    a_{j+1} ← temp
```

i = 9

temp ← N

S E L E C T I O N

C E E I L N O S T

# Decrease-by-a-Constant

## Sorting Problem
## (Decrease-by-One)

- given a sequence of n orderable items $\langle a_1, a_2, \ldots, a_n \rangle$, reorder the items as

  $\langle a_1', a_2' \ldots$ such that $\ldots$

- How c … to obtai …

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=o}^{i-1} 1 = \sum_{i=1}^{n-1} i = \frac{1}{2} n(n-1) \in \Theta(n^2)$$

  - just find an appropriate position to place $a_n$

**InsertionSort**($\langle a_1, a_2, \ldots, a_n \rangle$)

**input** : a sequence of orderable elements
**output**: sorted sequence in nondecreasing order

**for** i = 2 to n
    temp ← $a_i$
    j ← $i - 1$
    **while** $j \geq 0$ and $a_j >$ temp
        $a_{j+1} \leftarrow a_j$
        j ← j-1
    $a_{j+1} \leftarrow$ temp

i = 9

temp ← N

S E L E C T I O N

C E E I L N O S T

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- Directed graph can be used to represent order-dependent tasks

# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

- Directed graph can be used to represent order-dependent tasks


task v can start only after task u finishes

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- Directed graph can be used to represent order-dependent tasks


task v can start only after task u finishes

- Directed Acyclic Graph (DAG) must be used to represent such order-dependent tasks

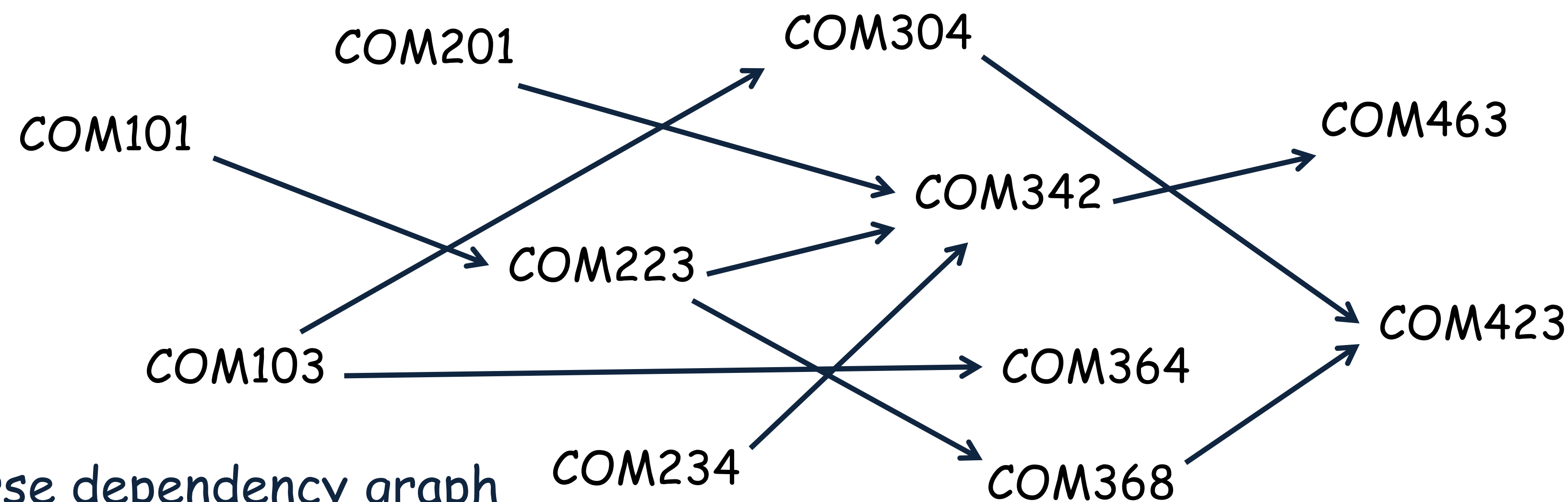
# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- Directed graph can be used to represent order-dependent tasks


task v can start only after task u finishes

- Directed Acyclic Graph (DAG) must be used to represent such order-dependent tasks




course dependency graph

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- a topological sort of a graph is a linear ordering of the vertices of a directed acyclic graph (DAG) such that if (u,v) is an edge in DAG, then u appears before v in this linear ordering

# Decrease-by-a-Constant

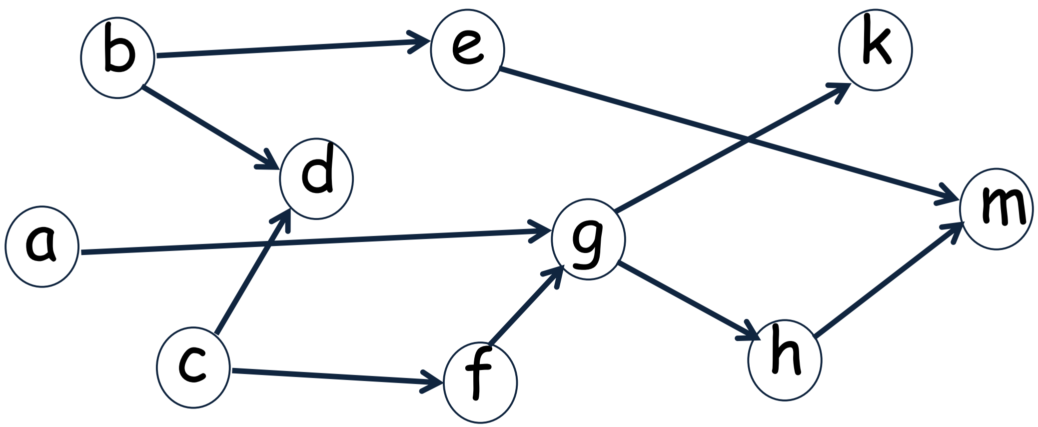
## Topological Sort (Decrease-by-One)

- a topological sort of a graph is a linear ordering of the vertices of a directed acyclic graph (DAG) such that if (u,v) is an edge in DAG, then u appears before v in this linear ordering
- for course dependency graph, a topological order gives in which order the courses should be taken

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

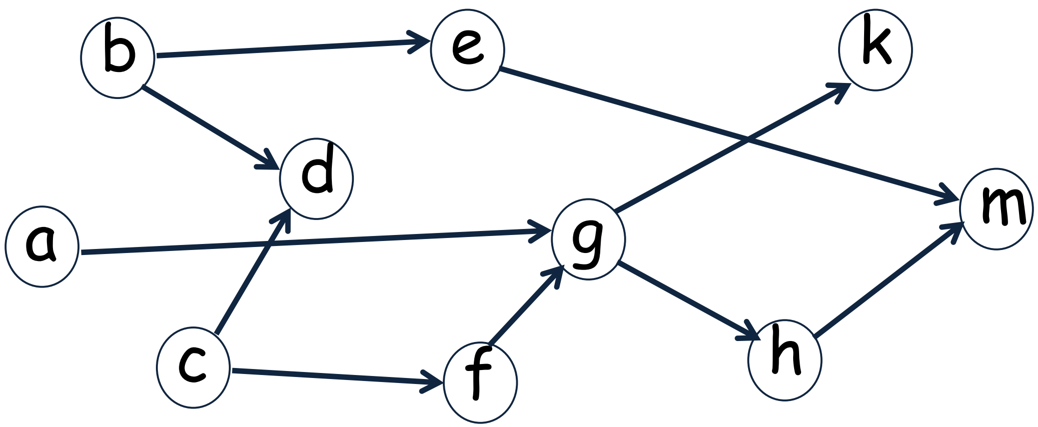
- a topological sort of a graph is a linear ordering of the vertices of a directed acyclic graph (DAG) such that if (u,v) is an edge in DAG, then u appears before v in this linear ordering
- for course dependency graph, a topological order gives in which order the courses should be taken



# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- a topological sort of a graph is a linear ordering of the vertices of a directed acyclic graph (DAG) such that if (u,v) is an edge in DAG, then u appears before v in this linear ordering
- for course dependency graph, a topological order gives in which order the courses should be taken

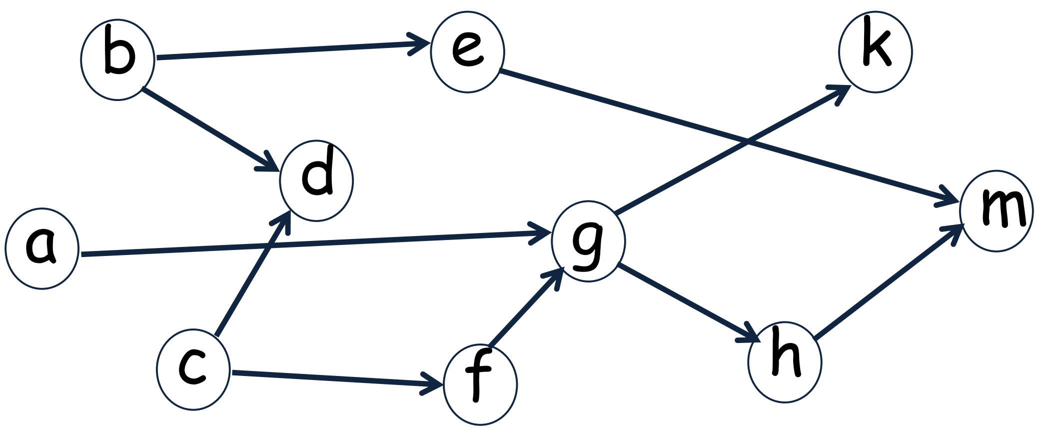



a, b, c, d, e, f, g, h, k, m
b, c, a, e, d, f, g, k, h, m
c, b, d, f, a, g, h, m, k, e

# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

- a topological sort of a graph is a linear ordering of the vertices of a directed acyclic graph (DAG) such that if (u,v) is an edge in DAG, then u appears before v in this linear ordering
- for course dependency graph, a topological order gives in which order the courses should be taken



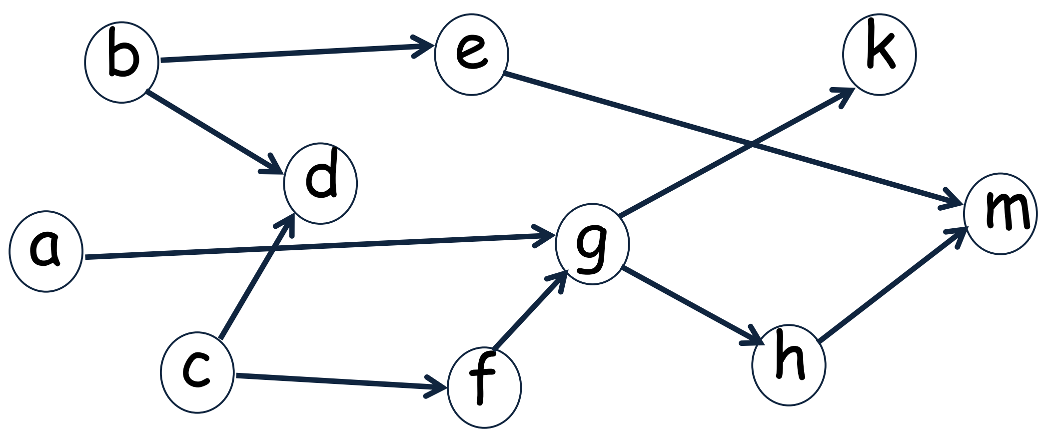

a, b, c, d, e, f, g, h, k, m
b, c, a, e, d, f, g, k, h, m
c, b, d, f, a, g, h, m, k, e

- there is an edge (e,m) in the graph, but e comes after m in the ordering

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- a topological sort of a graph is a linear ordering of the vertices of a directed acyclic graph (DAG) such that if (u,v) is an edge in DAG, then u appears before v in this linear ordering
- for course dependency graph, a topological order gives in which order the courses should be taken




a, b, c, d, e, f, g, h, k, m
b, c, a, e, d, f, g, k, h, m
c, b, d, f, a, g, h, m, k, e

- there is an edge (e,m) in the graph, but e comes after m in the ordering
- swap e and m

c, b, d, f, a, g, h, e, k, m

# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph

# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges

# Decrease-by-a-Constant

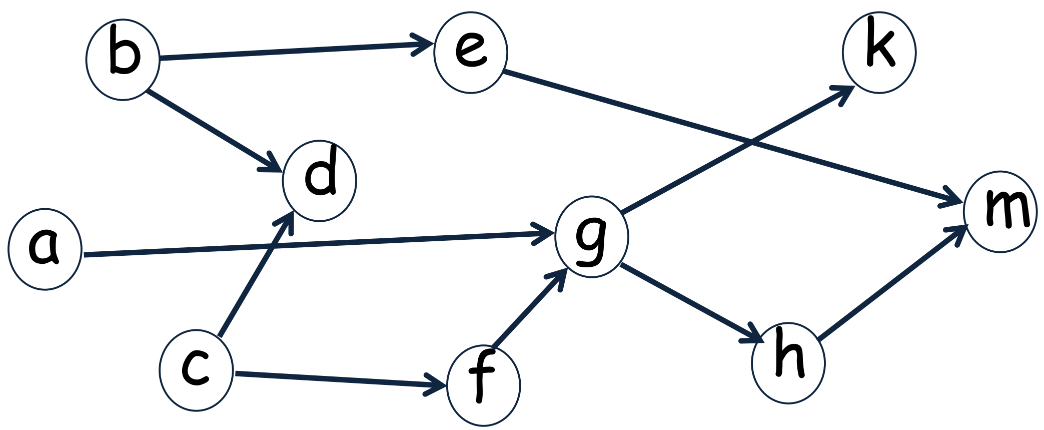
Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

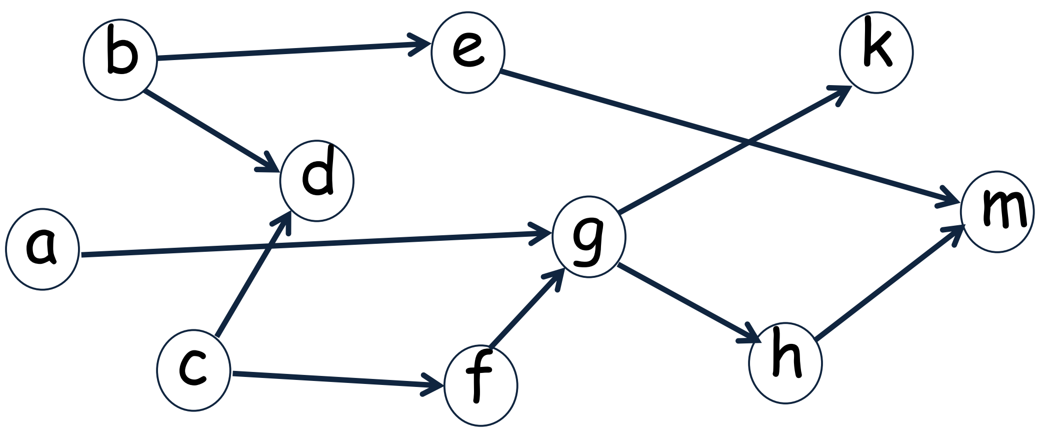
- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph



# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph



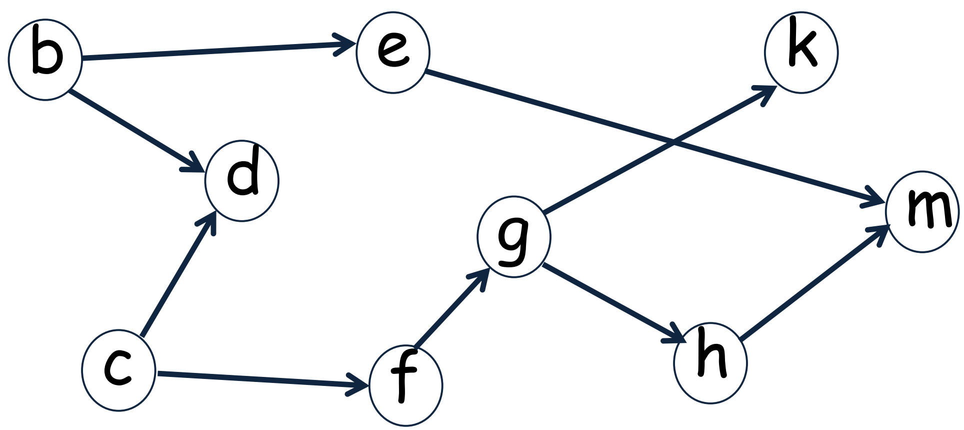

$O(|V| + \Sigma\ deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a



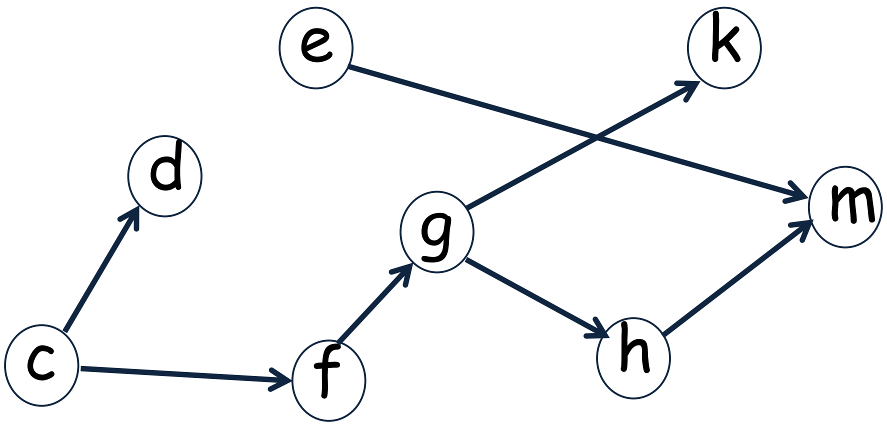

$O(|V| + \Sigma\ deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a, b



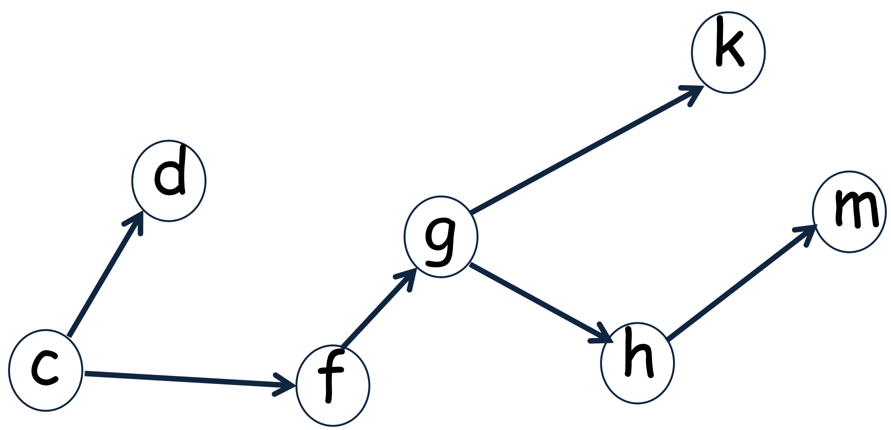

$O(|V| + \Sigma\, deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a, b, e



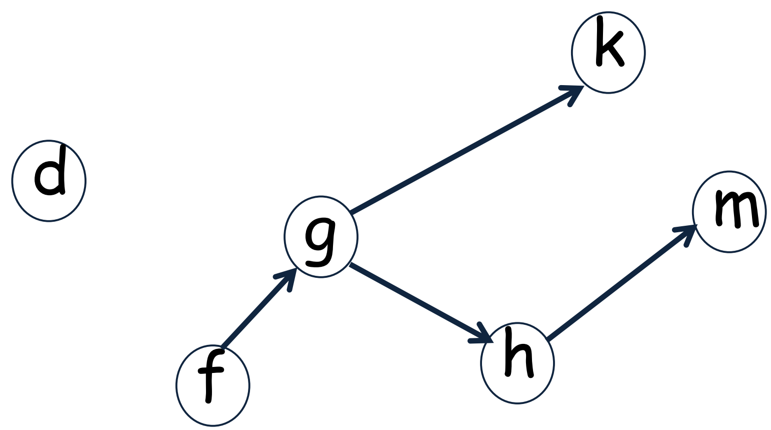

$O(|V| + \Sigma\ deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a, b, e, c



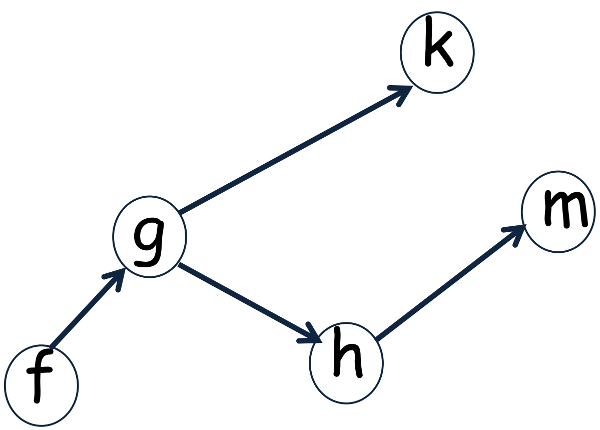

$O(|V| + \Sigma\ deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a, b, e, c, d



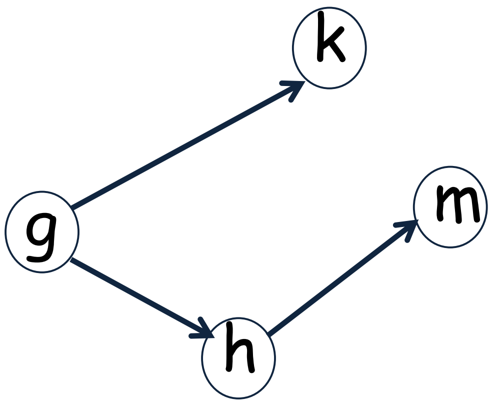

$O(|V| + \Sigma\, deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a, b, e, c, d, f



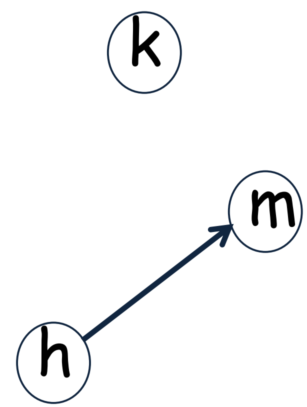

$O(|V| + \Sigma\, deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a, b, e, c, d, f, g



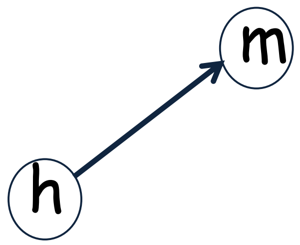

$O(|V| + \Sigma\ deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a, b, e, c, d, f, g, k




$O(|V| + \Sigma\ deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a, b, e, c, d, f, g, k, h

m

$O(|V| + \Sigma\ deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

- output a vertex u with $deg^{in}(u)=0$ from the graph
- remove all outgoing edges
- repeat the procedure until no more vertices in the graph

a, b, e, c, d, f, g, k, h, m

$O(|V| + \Sigma\, deg^{out}(v)) = O(|V|+|E|)$

# Decrease-by-a-Constant

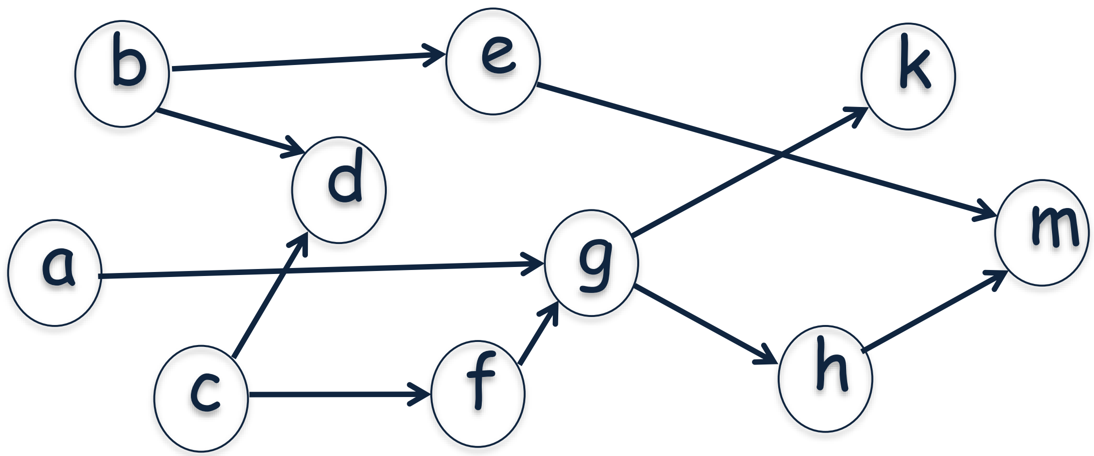
Topological Sort
(Decrease-by-One)

- run DFS on the given graph, and sort the vertices according to their finishing time

# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

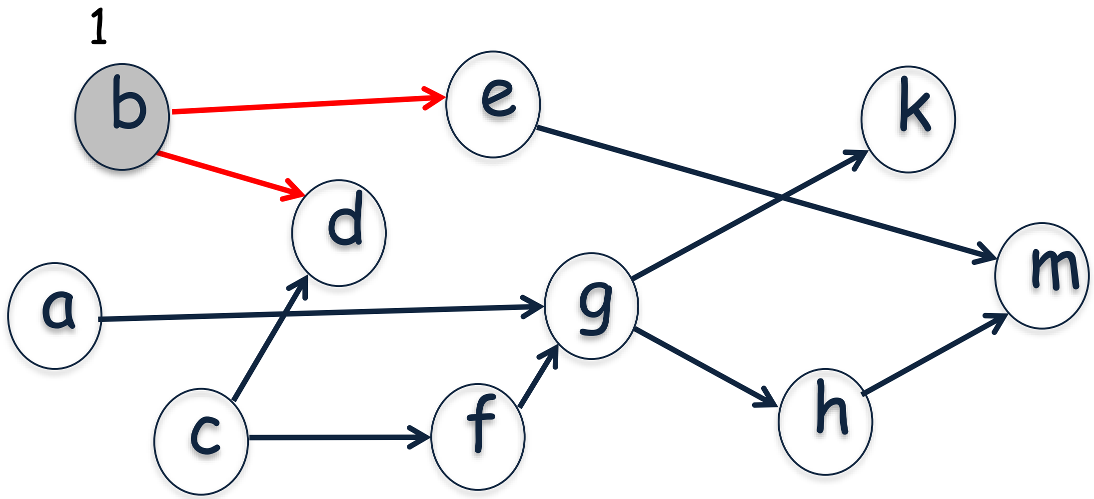
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

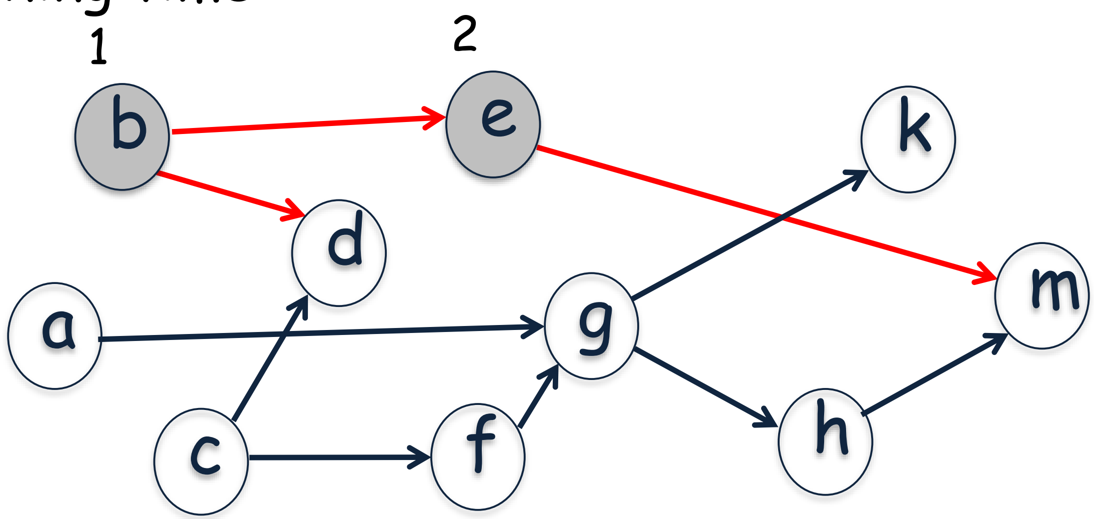
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

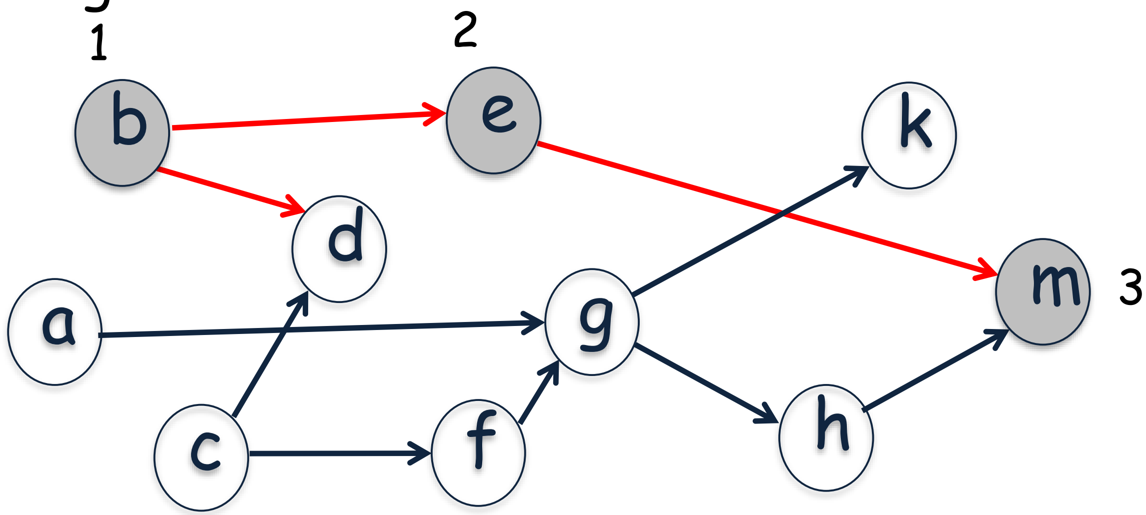
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

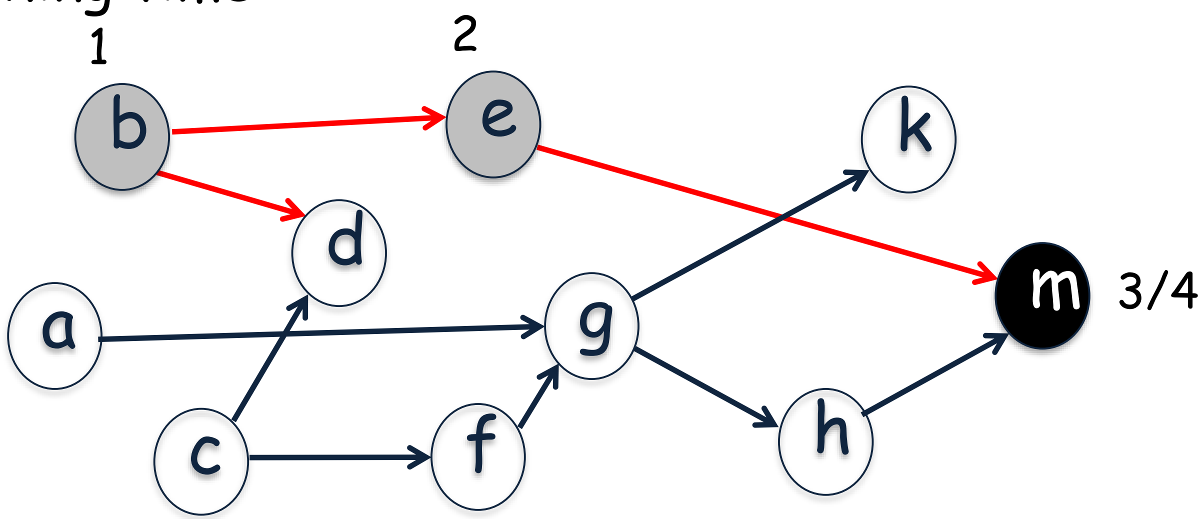
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

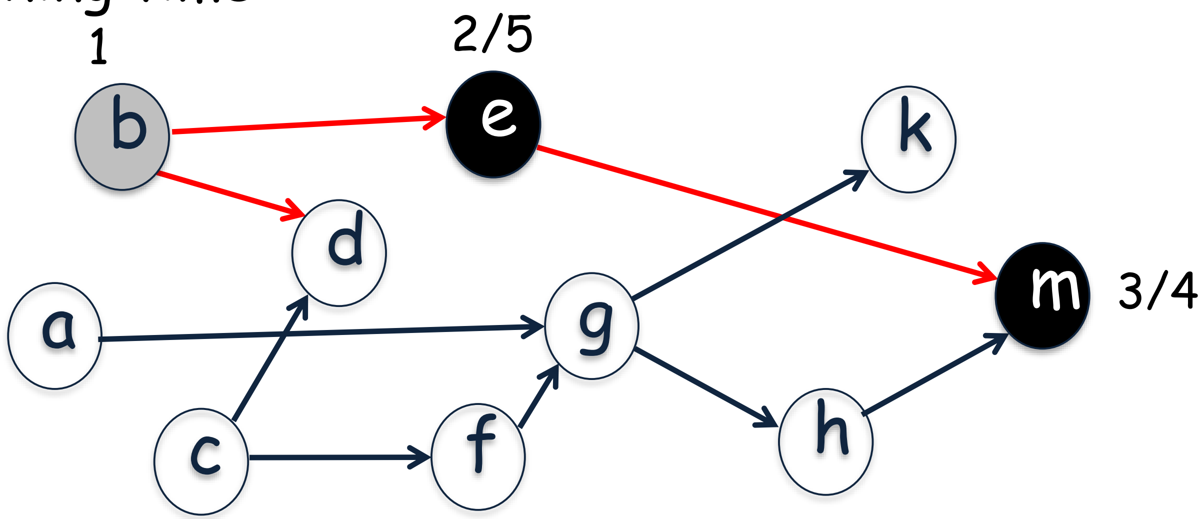
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

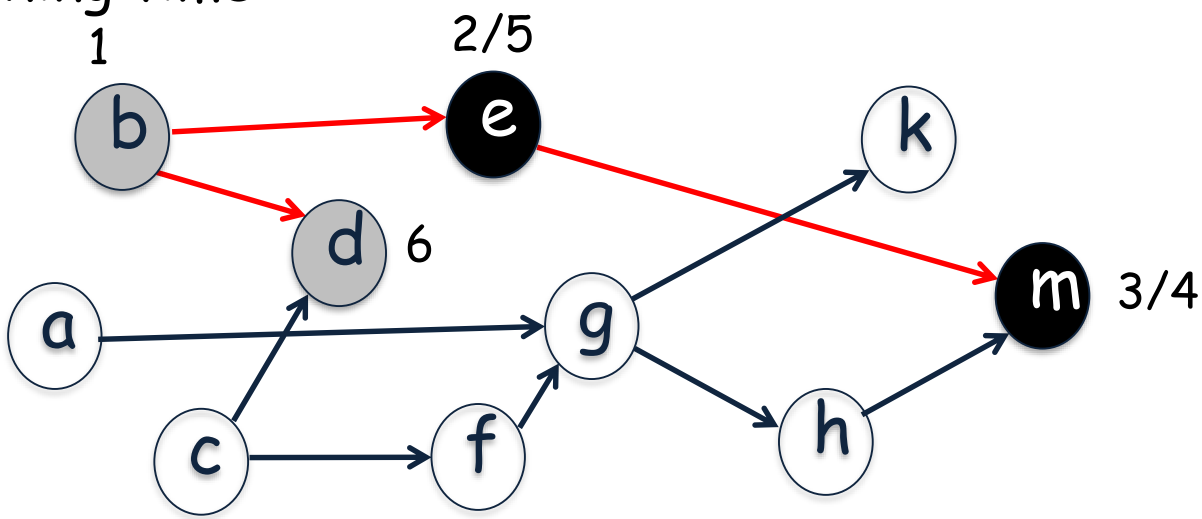
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

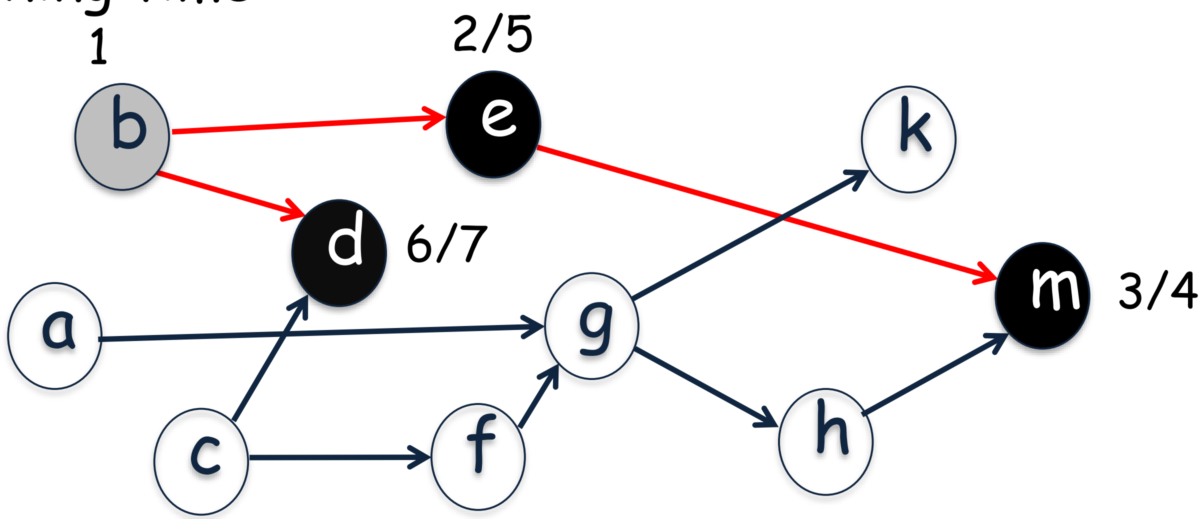
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort
## (Decrease-by-One)

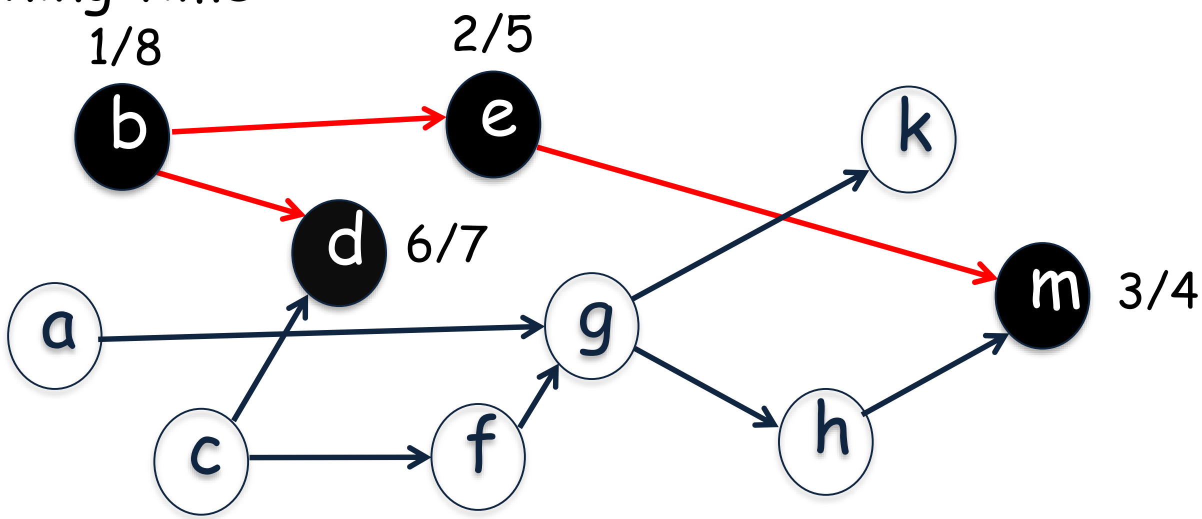
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort
## (Decrease-by-One)

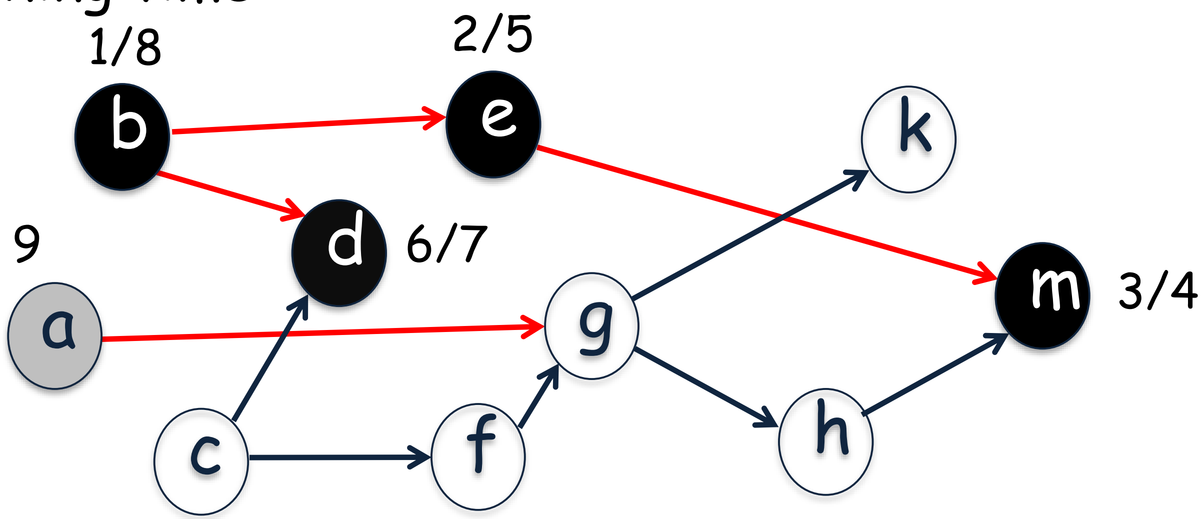
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

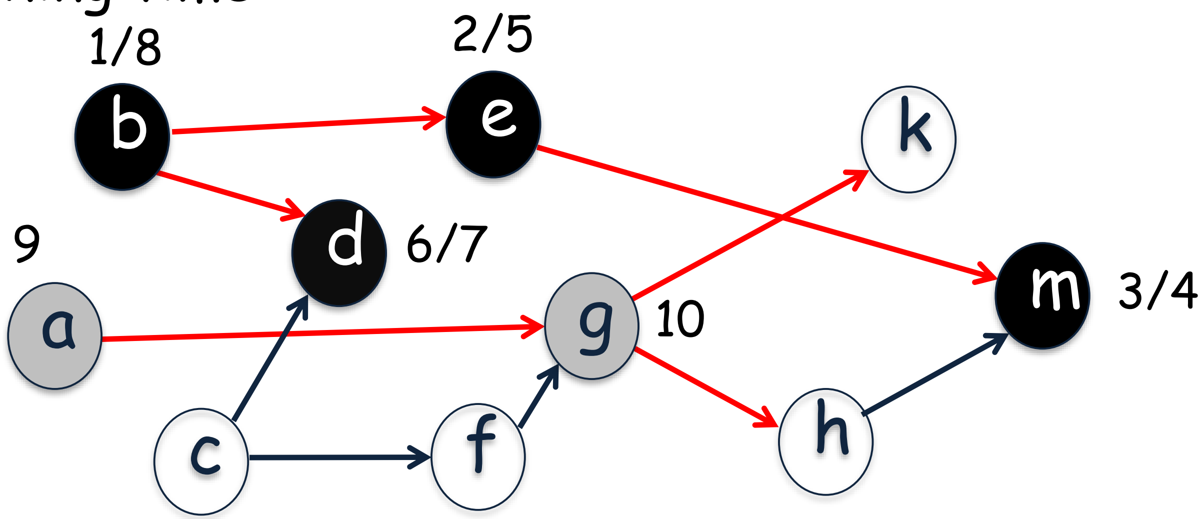
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

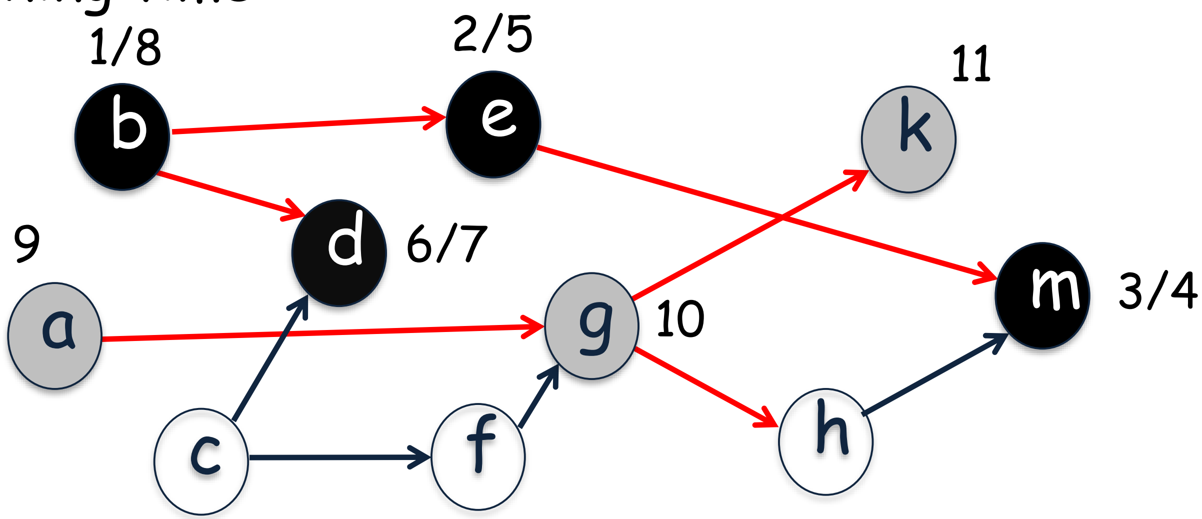
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

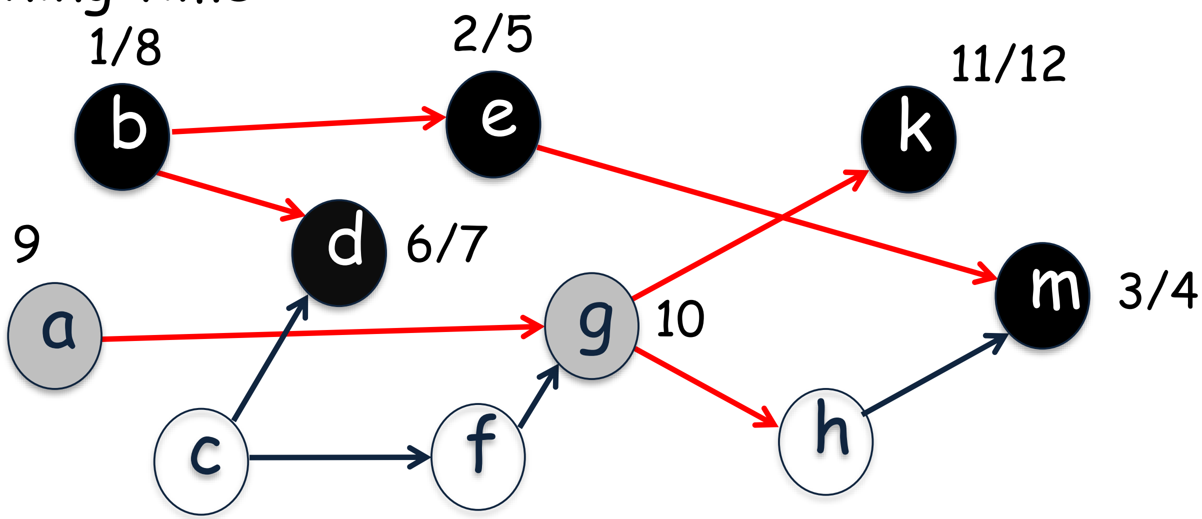
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

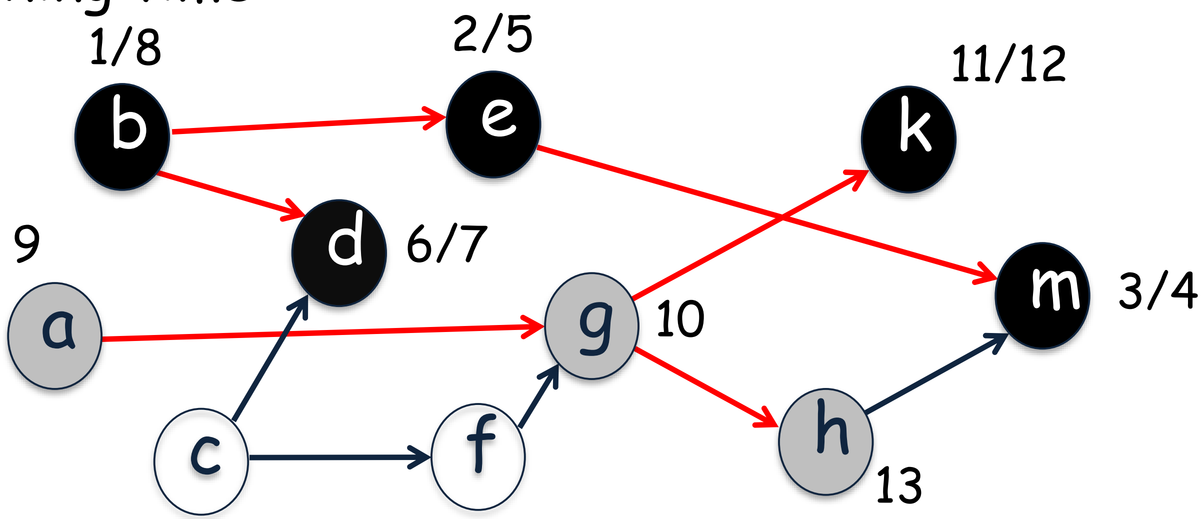
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

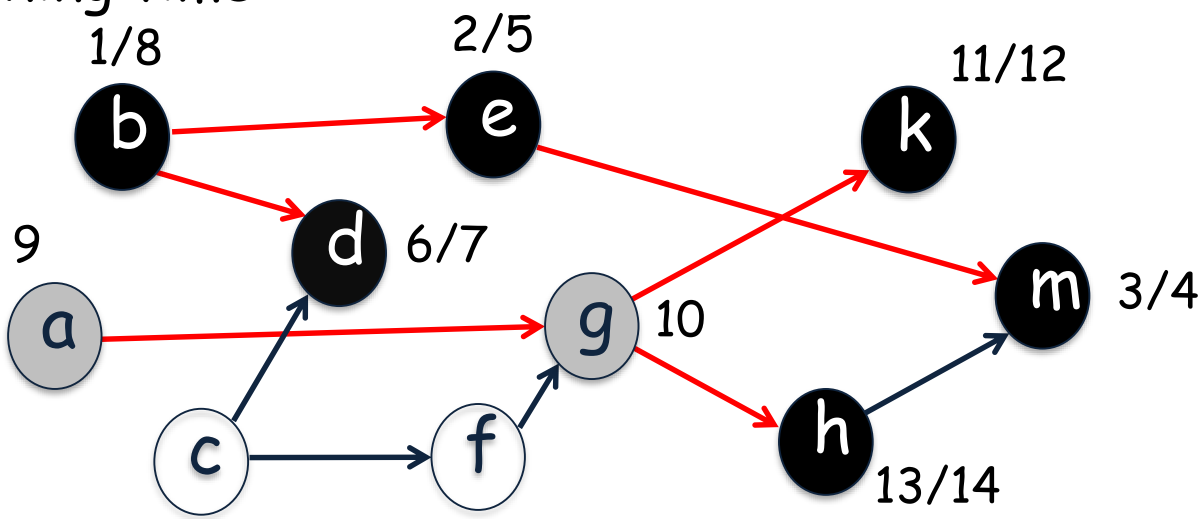
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort
## (Decrease-by-One)

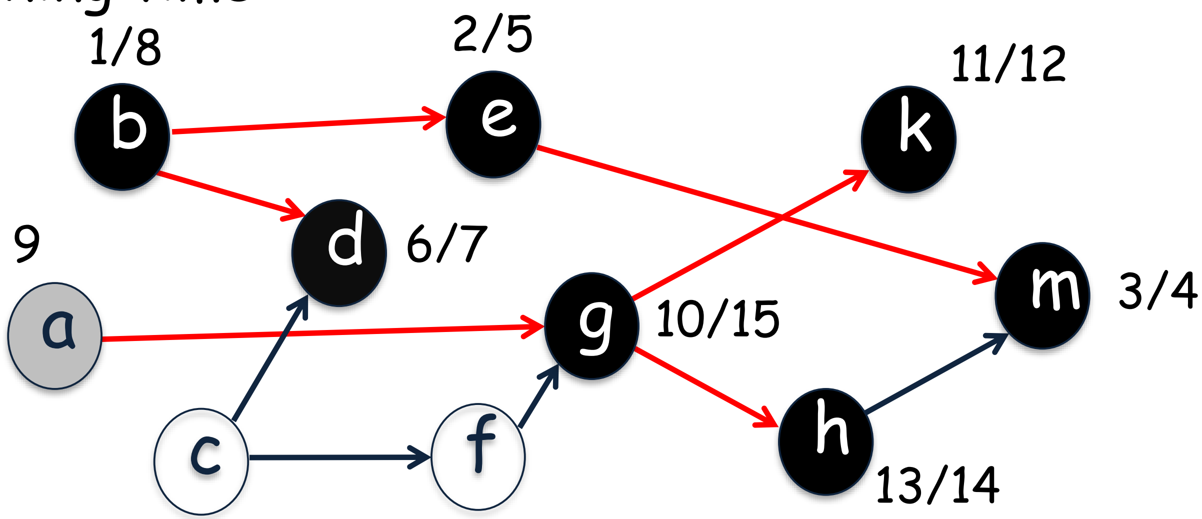
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

Topological Sort
(Decrease-by-One)

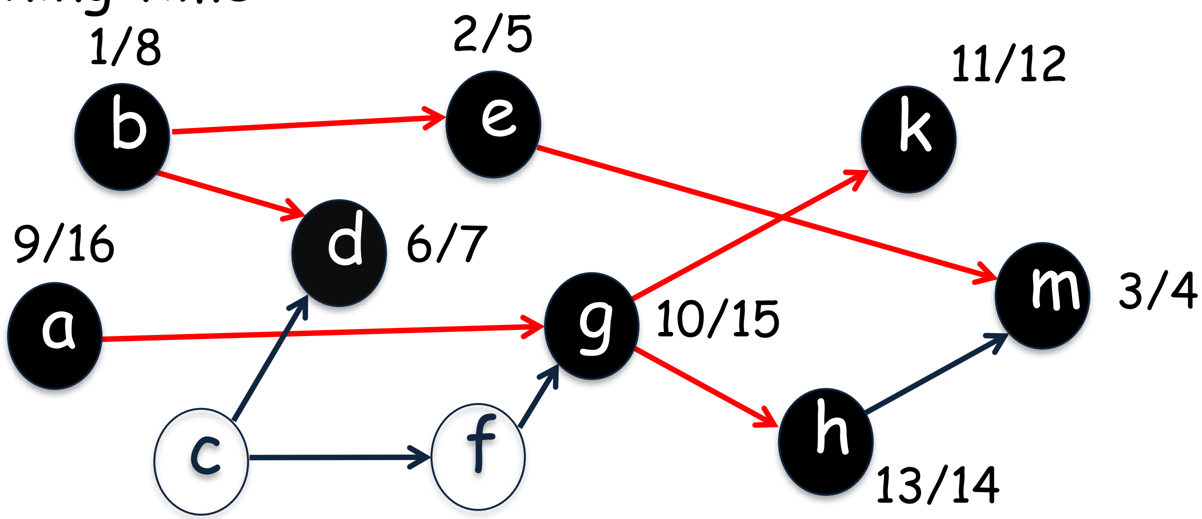
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

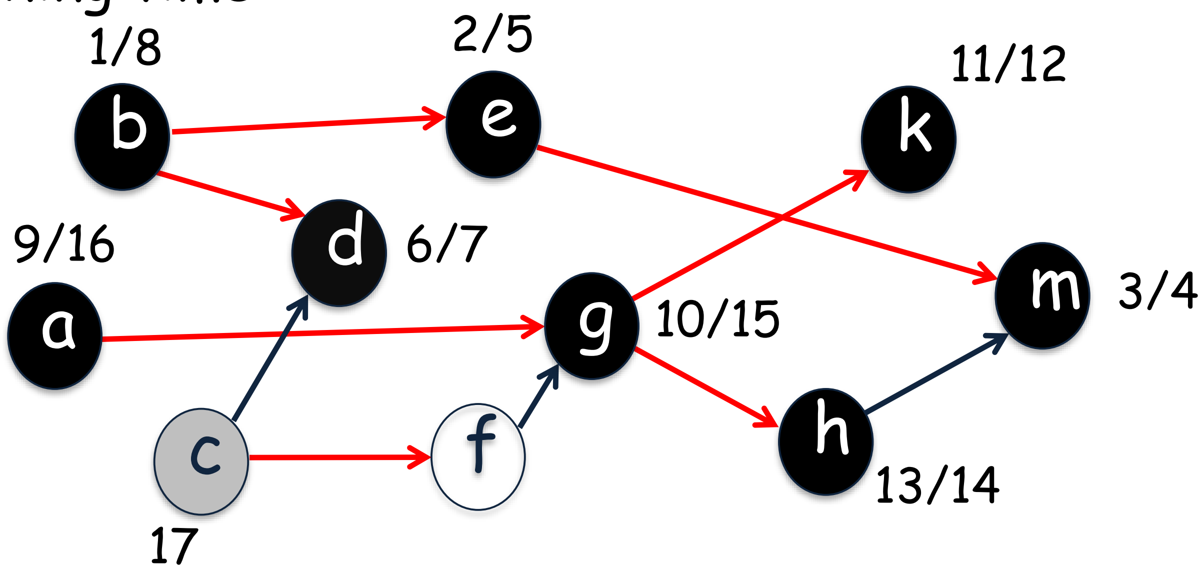
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort
(Decrease-by-One)

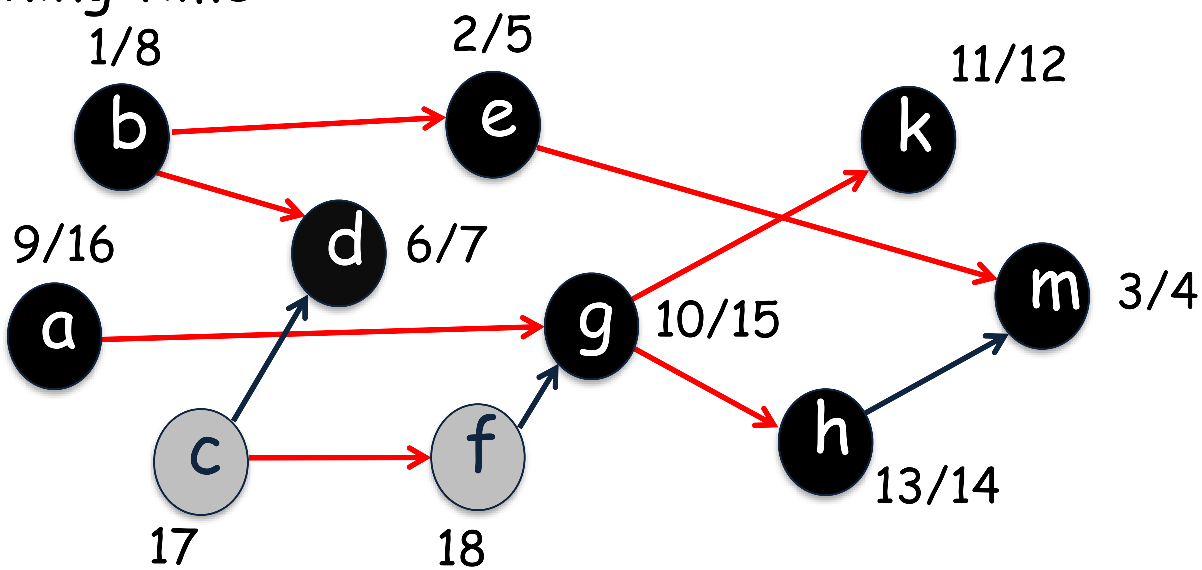
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

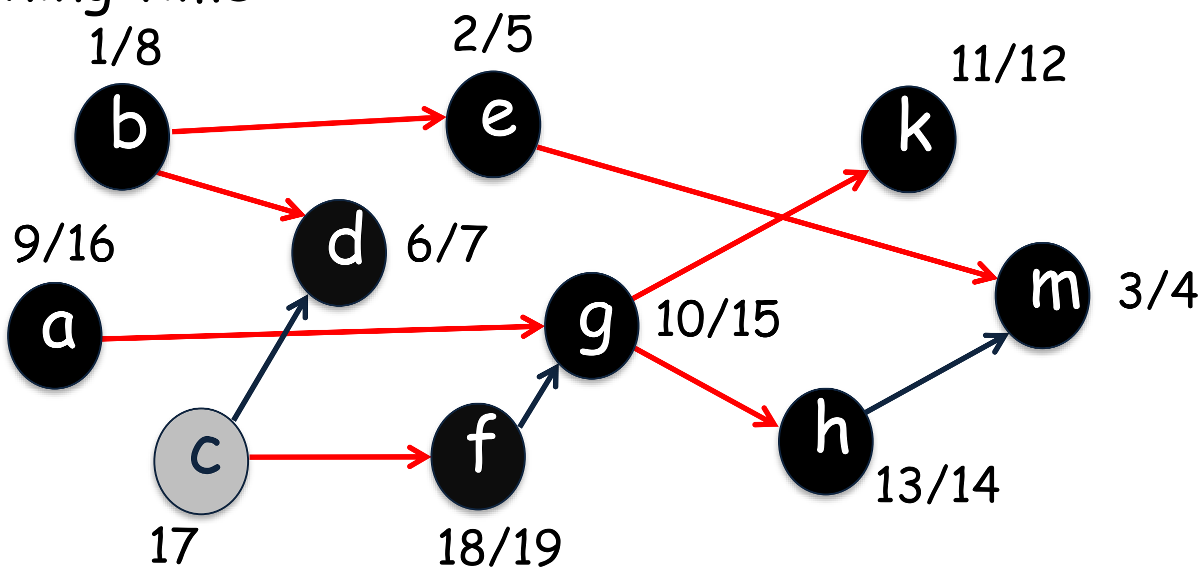
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

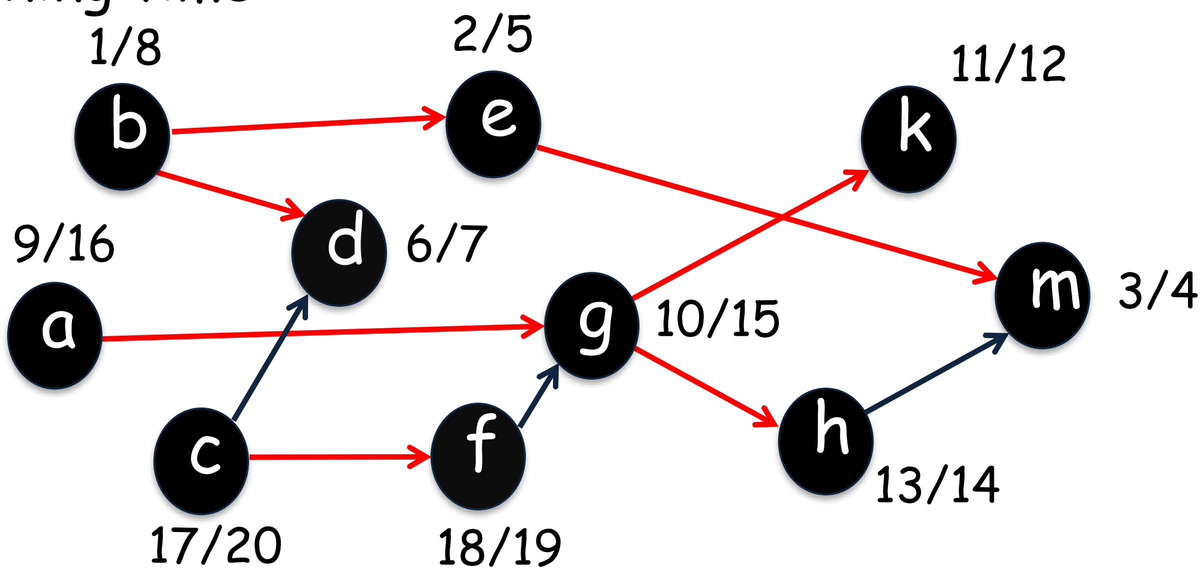
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

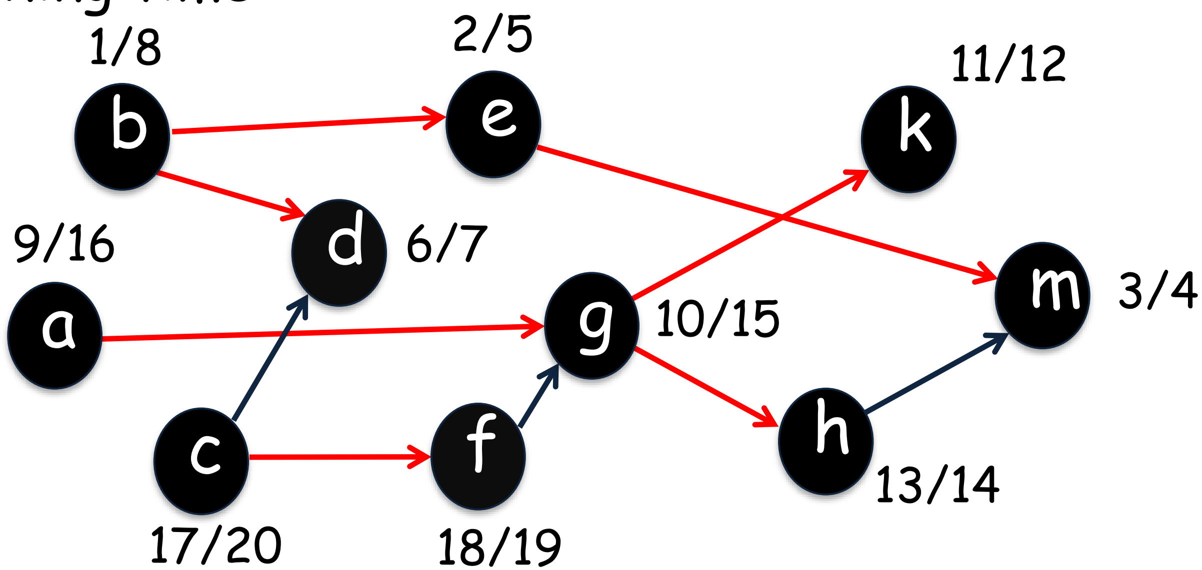
- run DFS on the given graph, and sort the vertices according to their finishing time



# Decrease-by-a-Constant

## Topological Sort (Decrease-by-One)

- run DFS on the given graph, and sort the vertices according to their finishing time




c, f, a, g, h, k, b, d, e, m

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \ldots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \ldots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \ldots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
    **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
        **return** 'yes'
    **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
        BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
    **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \ldots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \ldots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

$x = 70$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

```
while i ≤ j
    if  x = a_⌊(i+j)/2⌋
        return 'yes'
    elseif  x < a_⌊(i+j)/2⌋
        BinarySearch(X,i, ⌊(i+j)/2⌋-1;x)
    else BinarySearch(X,⌊(i+j)/2⌋+1,j;x)
```

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \dots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \dots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
  **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
    **return** 'yes'
  **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
    BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
  **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

$\boxed{x = 70}$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 1$  $j = 13$

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \dots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \dots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

```
while i ≤ j
    if  x = a_⌊(i+j)/2⌋
        return 'yes'
    elseif  x < a_⌊(i+j)/2⌋
        BinarySearch(X,i, ⌊(i+j)/2⌋-1;x)
    else BinarySearch(X,⌊(i+j)/2⌋+1,j;x)
```

$x = 70$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 1$ $j = 13$

$\lfloor (i + j)/2 \rfloor = 7$

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \dots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \dots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
  **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
    **return** 'yes'
  **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
    BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
  **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

$\boxed{x = 70}$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 1$  $j = 13$

$\lfloor (i+j)/2 \rfloor = 7$

$\boldsymbol{x > a_7 = 55}$

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \dots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \dots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
  **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
    **return** 'yes'
  **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
    BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
  **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

$\boxed{x = 70}$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 8$ $j = 13$

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \dots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \dots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
  **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
    **return** 'yes'
  **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
    BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
  **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

$\boxed{x = 70}$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 8$  $j = 13$

$\lfloor (i+j)/2 \rfloor = 10$

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \dots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \ldots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
  **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
    **return** 'yes'
  **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
    BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
  **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

$\boxed{x = 70}$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 8$  $j = 13$

$\lfloor (i+j)/2 \rfloor = 10$

$\boldsymbol{x < a_{10} = 81}$

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \dots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \dots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
  **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
    **return** 'yes'
  **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
    BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
  **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

$\boxed{x = 70}$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 8$ $j = 9$

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \dots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \dots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
  **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
    **return** 'yes'
  **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
    BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
  **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

$\boxed{x = 70}$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 8$

$j = 9$

$\lfloor (i+j)/2 \rfloor = 8$

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items $[a_1, a_2, \ldots, a_n]$ and a search key K, determine whether the sorted sequence contains the key or not

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \ldots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
  **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
    **return** 'yes'
  **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
    BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
  **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

$\boxed{x = 70}$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 8$  $j = 9$

$\lfloor (i+j)/2 \rfloor = 8$

$\boldsymbol{x = a_8 = 70}$

# Decrease-by-a-Constant-Factor

## Binary Search

- Given a sorted sequence of n items [...] and a search key K, determine whether

$$T(n) = T(n/2) + 1, \text{ and } T(n) = 1 \text{ for n} = 1$$

$$T(n) = \log n + 1 \in \Theta(n^2)$$

**BinarySearch(X,i,j;x)**

**input** : $\{X = \{a_1, a_2, \ldots, a_n\}; x\}$
**output**: 'yes' if $x \in X$, 'no' otherwise

**while** i ≤ j
  **if** $x = a_{\lfloor (i+j)/2 \rfloor}$
    **return** 'yes'
  **elseif** $x < a_{\lfloor (i+j)/2 \rfloor}$
    BinarySearch(X,i, $\lfloor (i+j)/2 \rfloor$-1;x)
  **else** BinarySearch(X,$\lfloor (i+j)/2 \rfloor$+1,j;x)

$x = 70$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 14 | 27 | 31 | 39 | 42 | 55 | 70 | 74 | 81 | 85 | 93 | 98 |

$i = 8$   $j = 9$

$\lfloor (i+j)/2 \rfloor = 8$

$$\boldsymbol{x = a_8 = 70}$$

# Decrease-by-Variable-Size

## Selection Problem

- Given a sequence of n numbers $[a_1, a_2, \ldots, a_n]$, determine the k-th smallest element of the sequence

# Decrease-by-Variable-Size

## Selection Problem

- Given a sequence of n numbers $[a_1, a_2, \ldots, a_n]$, determine the k-th smallest element of the sequence

- for k = 1 or k = n, find the smallest or largest element by scanning the sequence

# Decrease-by-Variable-Size

## Selection Problem

- Given a sequence of n numbers $[a_1, a_2, \dots, a_n]$, determine the k-th smallest element of the sequence
- for k = 1 or k = n, find the smallest or largest element by scanning the sequence
- for $k = \lceil n/2 \rceil$, it's finding the median (the middle value) of the sequence

# Decrease-by-Variable-Size

## Selection Problem

- Given a sequence of n numbers $[a_1, a_2, \ldots, a_n]$, determine the k-th smallest element of the sequence
- for k = 1 or k = n, find the smallest or largest element by scanning the sequence
- for $k = \lceil n/2 \rceil$, it's finding the median (the middle value) of the sequence
- **Brute-force approach**; first sort the given sequence, then output the k-th element of the sorted sequence

# Decrease-by-Variable-Size

## Selection Problem

- Given a sequence of n numbers $[a_1, a_2, \dots, a_n]$, determine the k-th smallest element of the sequence

- for k = 1 or k = n, find the smallest or largest element by scanning the sequence

- for $k = \lceil n/2 \rceil$, it's finding the median (the middle value) of the sequence

- **Brute-force approach**; first sort the given sequence, then output the k-th element of the sorted sequence

  since the problem is to just find the k-th smallest element, sorting the entire sequence would be unnecessary

# Decrease-by-Variable-Size

## Selection Problem

- partition the given sequence around some value p (pivot), that is the first element

# Decrease-by-Variable-Size

## Selection Problem

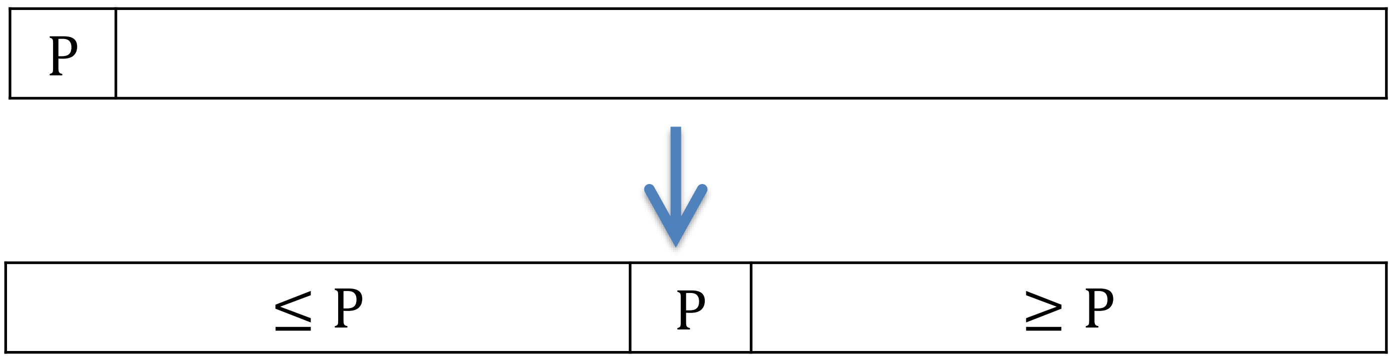
- partition the given sequence around some value p (pivot), that is the first element

| | |
|---|---|
| P | |

# Decrease-by-Variable-Size

## Selection Problem

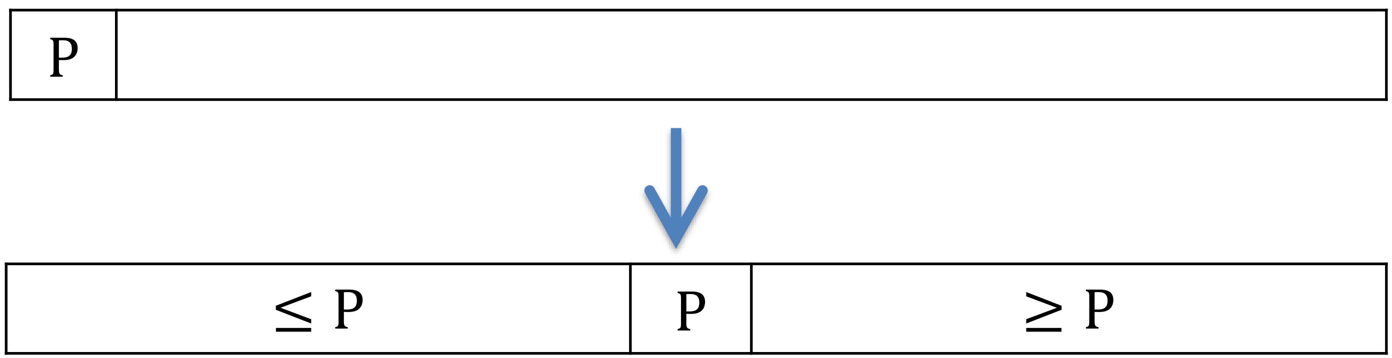
- partition the given sequence around some value p (pivot), that is the first element



# Decrease-by-Variable-Size

## Selection Problem

- partition the given sequence around some value p (pivot), that is the first element

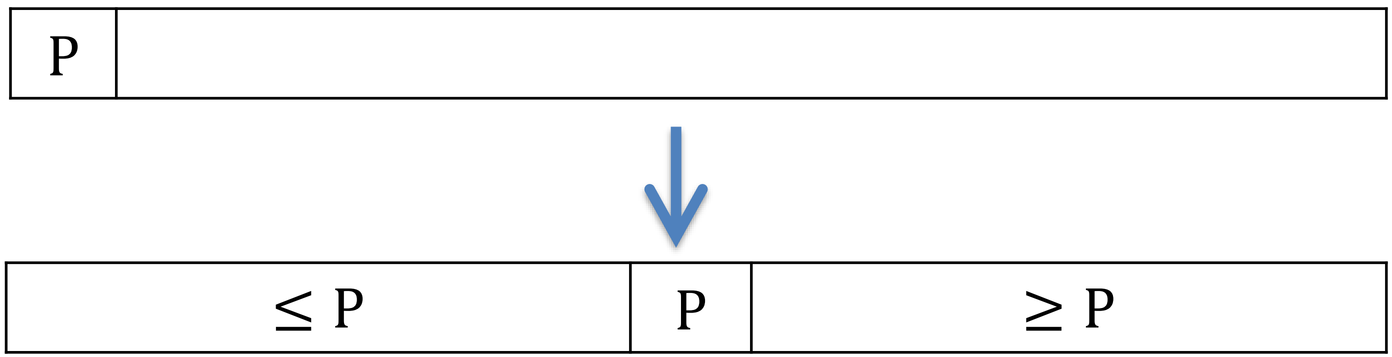



- assume s be the index of the pivot

# Decrease-by-Variable-Size

## Selection Problem

- partition the given sequence around some value p (pivot), that is the first element

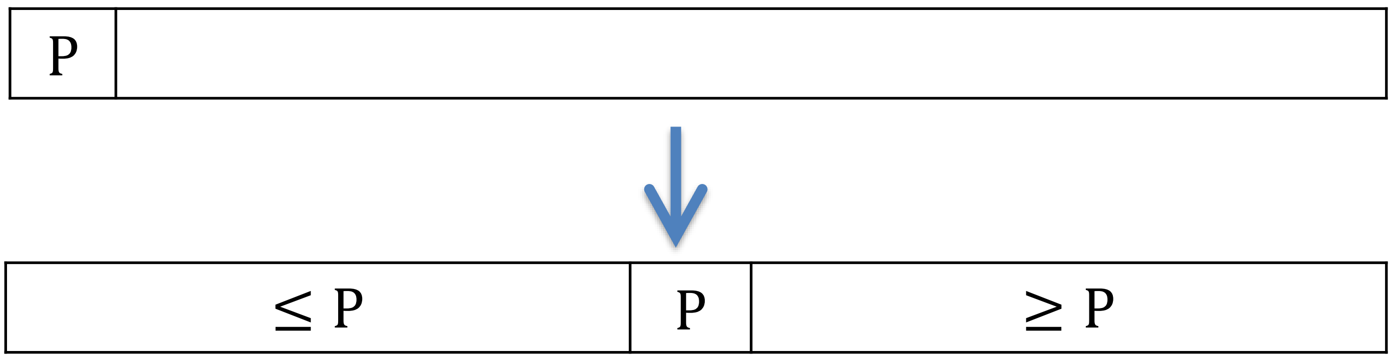



- assume s be the index of the pivot
  - if s = k, then the pivot is the k-th smallest element

# Decrease-by-Variable-Size

## Selection Problem

- partition the given sequence around some value p (pivot), that is the first element

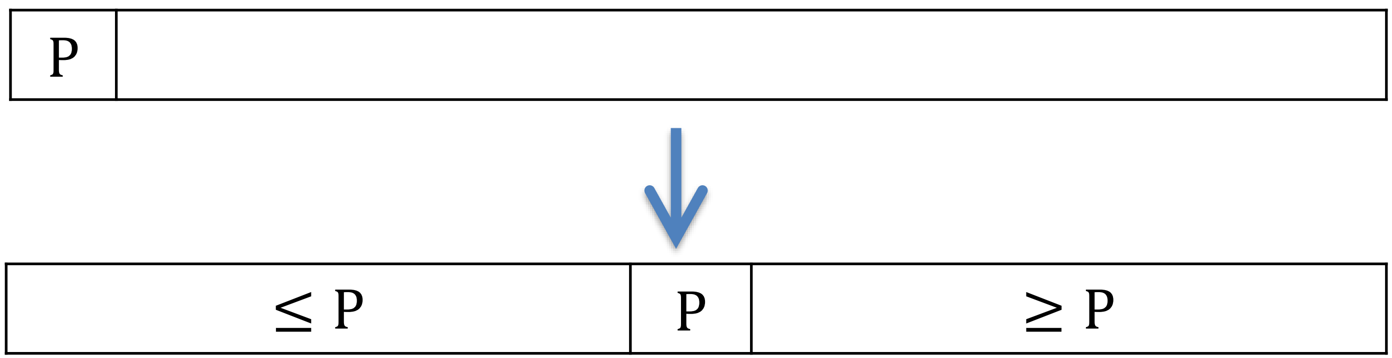



- assume s be the index of the pivot
  - if s = k, then the pivot is the k-th smallest element
  - if s > k, then the k-th smallest element will be the k-th smallest of the left

# Decrease-by-Variable-Size

## Selection Problem

- partition the given sequence around some value p (pivot), that is the first element




- assume s be the index of the pivot
  - if $s = k$, then the pivot is the k-th smallest element
  - if $s > k$, then the k-th smallest element will be the k-th smallest of the left
  - if $s < k$, then the k-th smallest element will be the $(k - s)$-th smallest element of the right

# Decrease-by-Variable-Size

## Selection Probler

- partition the gi hat is the first element

P

**QuickSelect( i, j, k )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$ and an integer k
**output**: the k-th smallest of the sequence X

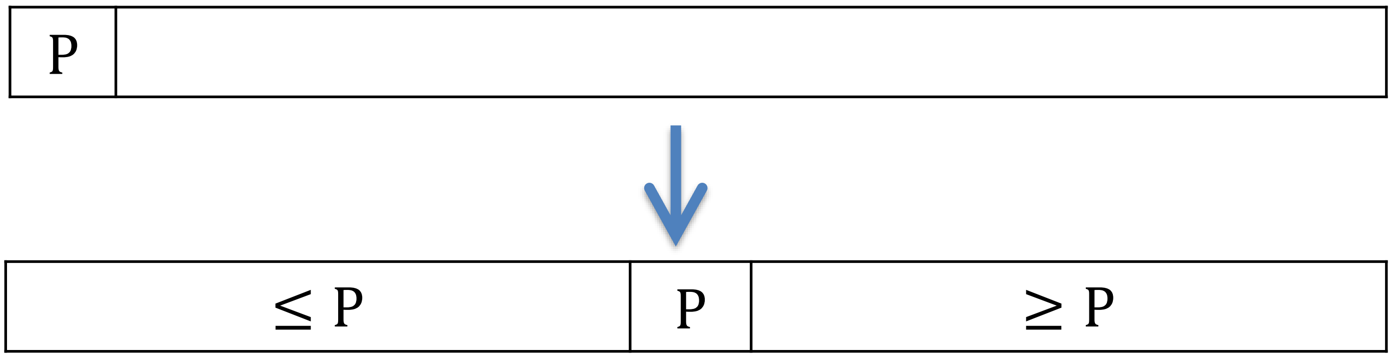
```
s ← Partition({a_i, a_{i+1}, ..., a_j})
if s = k
    return a_s
elseif s > i + k
    QuickSelect( i, s - 1, k )
else QuickSelect( s + 1, j, k - s )
```

- assume s be the index of the pivot
  - if s = k, then the pivot is the k-th smallest element
  - if s > k, then the k-th smallest element will be the k-th smallest of the left
  - if s < k, then the k-th smallest element will be the (k – s)-th smallest element of the right

# Decrease-by-Variable-Size

## Selection Problem

- partition the given sequence around some value p (pivot), that is the first element




- assume s
  - if s =
  - if s > … smallest of th
  - if s < … k – s)-th smalle

**LomutoPartition( i, j )**

**input** :$X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output**: the partition of X and the new position of the pivot

```
p ← a_i ; s ← i
for k = i + 1 to j
    if a_k < p
        s ← s + 1 ; swap(a_s, a_k)
swap(a_i, a_s)
return s
```

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output**: the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

| | | | | | |
|---|---|---|---|---|---|
| 13 | 21 | 5 | 14 | 8 | 10 |

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output**: the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1 j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 21 | 5 | 14 | 8 | 10 |

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \dots, a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1 j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 21 | 5 | 14 | 8 | 10 |

p = 13 s = 1

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 21 | 5 | 14 | 8 | 10 |

p = 13    s = 1    k = 2

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output**: the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1 j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 21 | 5 | 14 | 8 | 10 |

p = 13 s = 1 k = 2

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** :$X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 21 | 5 | 14 | 8 | 10 |

p = 13    s = 1    k = 3

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, ..., a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 21 | 5 | 14 | 8 | 10 |

p = 13    s = 1    k = 3

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** :$X = \{a_i, a_{i+1}, ..., a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
  **if** $a_k$ < p
    s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1 j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 21 | 5 | 14 | 8 | 10 |

p = 13 s = 2 k = 3

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output**: the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 21 | 14 | 8 | 10 |

p = 13    s = 2    k = 3

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output**: the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 21 | 14 | 8 | 10 |

p = 13    s = 2    k = 4

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, ..., a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 21 | 14 | 8 | 10 |

p = 13    s = 2    k = 4

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output**: the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1 j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 21 | 14 | 8 | 10 |

p = 13 s = 2 k = 5

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1  j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 21 | 14 | 8 | 10 |

p = 13  s = 2  k = 5

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
  **if** $a_k$ < p
    s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1 j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 21 | 14 | 8 | 10 |

p = 13 s = 3 k = 5

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 8 | 14 | 21 | 10 |

p = 13    s = 3    k = 5

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, ..., a_j\}$
**output**: the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 8 | 14 | 21 | 10 |

p = 13    s = 3    k = 6

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1 j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 8 | 14 | 21 | 10 |

p = 13 s = 3 k = 6

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 8 | 14 | 21 | 10 |

p = 13    s = 4    k = 6

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output**: the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return** s

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 13 | 5 | 8 | 10 | 21 | 14 |

p = 13    s = 4    k = 6

# Decrease-by-Variable-Size

## Selection Problem

**LomutoPartition( i, j )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$
**output:** the partition of X and the new position of the pivot

p ← $a_i$ ; s ← i
**for** k = i + 1 to j
    **if** $a_k$ < p
        s ← s + 1 ; swap($a_s$, $a_k$)
swap($a_i$, $a_s$)
**return s**

i = 1    j = 6

| | | | | | |
|---|---|---|---|---|---|
| 10 | 5 | 8 | 13 | 21 | 14 |

p = 13    s = 4    k = 6

# Decrease-by-Variable-Size

## Selection Problem

**QuickSelect( i, j, k )**

**input** : $X = \{a_i, a_{i+1}, ..., a_j\}$ and an integer k
**output:** the k-th smallest of the sequence X

s ← Partition($\{a_i, a_{i+1}, ..., a_j\}$)
**if** s = k
    **return** $a_s$
**elseif** s > i + k
    QuickSelect( i, s - 1, k )
**else** QuickSelect( s + 1, j, k - s )

i = 1    k = 5    j = 9

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 10 | 8 | 7 | 12 | 9 | 2 | 15 |

# Decrease-by-Variable-Size

## Selection Problem

**QuickSelect( i, j, k )**

**input** : $X = \{a_i, a_{i+1}, ..., a_j\}$ and an integer k
**output**: the k-th smallest of the sequence X

s ← Partition($\{a_i, a_{i+1}, ..., a_j\}$)
**if** s = k
    **return** $a_s$
**elseif** s > i + k
    QuickSelect( i, s - 1, k )
**else** QuickSelect( s + 1, j, k - s )

i = 1    k = 5    j = 9

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 10 | 8 | 7 | 12 | 9 | 2 | 15 |

s = 3

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 4 | 8 | 7 | 12 | 9 | 10 | 15 |

# Decrease-by-Variable-Size

## Selection Problem

**QuickSelect( i, j, k )**

**input** :$X = \{a_i, a_{i+1}, ..., a_j\}$ and an integer k
**output**: the k-th smallest of the sequence X

s ← Partition($\{a_i, a_{i+1}, ..., a_j\}$)
**if** s = k
    **return** $a_s$
**elseif** s > i + k
    QuickSelect( i, s - 1, k )
**else** QuickSelect( s + 1, j, k - s )

i = 1 k = 5 j = 9

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 10 | 8 | 7 | 12 | 9 | 2 | 15 |

s = 3

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 4 | 8 | 7 | 12 | 9 | 10 | 15 |

i = 4 j = 9

| | | | | | |
|---|---|---|---|---|---|
| 8 | 7 | 12 | 9 | 10 | 15 |

# Decrease-by-Variable-Size

## Selection Problem

**QuickSelect( i, j, k )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$ and an integer k
**output**: the k-th smallest of the sequence X

s ← Partition($\{a_i, a_{i+1}, \ldots, a_j\}$)
**if** s = k
  **return** $a_s$
**elseif** s > i + k
  QuickSelect( i, s - 1, k )
**else** QuickSelect( s + 1, j, k - s )

i = 1  k = 5  j = 9

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 10 | 8 | 7 | 12 | 9 | 2 | 15 |

s = 3

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 4 | 8 | 7 | 12 | 9 | 10 | 15 |

i = 4  j = 9

| | | | | | |
|---|---|---|---|---|---|
| 8 | 7 | 12 | 9 | 10 | 15 |

s = 5

| | | | | | |
|---|---|---|---|---|---|
| 7 | 8 | 12 | 9 | 10 | 15 |

# Decrease-by-Variable-Size

## Selection Problem

**QuickSelect( i, j, k )**

**input** : $X = \{a_i, a_{i+1}, \dots, a_j\}$ and an integer k
**output**: the k-th smallest of the sequence X

s ← Partition($\{a_i, a_{i+1}, \dots, a_j\}$)
**if** s = k
    **return** $a_s$
**elseif** s > i + k
    QuickSelect( i, s - 1, k )
**else** QuickSelect( s + 1, j, k - s )

i = 1    k = 5    j = 9

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 10 | 8 | 7 | 12 | 9 | 2 | 15 |

s = 3

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 4 | 8 | 7 | 12 | 9 | 10 | 15 |

i = 4    j = 9

| | | | | | |
|---|---|---|---|---|---|
| 8 | 7 | 12 | 9 | 10 | 15 |

s = 5

| | | | | | |
|---|---|---|---|---|---|
| 7 | 8 | 12 | 9 | 10 | 15 |

# Decrease-by-Variable-Size

## Selection Problem

**QuickSelect( i, j, k )**

**input** : $X = \{a_i, a_{i+1}, \ldots, a_j\}$ and an integer k
**output**: the k-th smallest of the sequence X

s ← Partition($\{a_i, a_{i+1}, \ldots, a_j\}$)
**if** s = k
  **return** $a_s$
**elseif** s > i + k
  QuickSelect( i, s - 1, k )
**else** QuickSelect( s + 1, j, k - s )

i = 1 k = 5 j = 9

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 10 | 8 | 7 | 12 | 9 | 2 | 15 |

s = 3

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 4 | 8 | 7 | 12 | 9 | 10 | 15 |

j = 9

| | |
|---|---|
| 0 | 15 |

$$T(n) = (n-1) + (n-2) + \cdots + 1 = n(n-1)/2 \in \Theta(n^2)$$

s = 5

| | | | | | |
|---|---|---|---|---|---|
| 7 | 8 | 12 | 9 | 10 | 15 |