

A Compendium of Basic Algorithms

Melvin Cabatuan
 Electronics and Computer Engineering Department
 De La Salle University
 Manila, Philippines
 Email: melvin.cabatuan@dlsu.edu.ph

I. INTRODUCTION

This lecture notes includes a collection of commonly used algorithms in an introductory course for a computational background. An algorithm is a finite set of precise instructions for performing a computation or for solving a problem. An algorithm has input values from a specified set. From each set of input values, an algorithm produces output values. An algorithm should yield the correct output values for each set of input values. Furthermore, the steps of an algorithm must be defined precisely and the desired output must be produced after a finite number of steps. Finally, the procedure should be applicable for all problems of the desired form, not just for a particular set of input.

An algorithm is often described using a pseudocode which is a high-level description of an algorithm that uses the structural conventions of a standard programming language, but is intended for human reading. The following sections describe an algorithm in terms of a pseudocodes.

II. PSEUDOCODES

Finding the Maximum

Algorithm 1 Calculate *maximum* element in a finite integer set

Require: $\{a_1, a_2, \dots, a_i, \dots, a_n\} \in \mathbb{Z}$
Ensure: $result = \max(a_1, a_2, \dots, a_i, \dots, a_n)$
 $result \leftarrow a_1$
for $i = 2$ **to** n **do**
 if $result < a_i$ **then**
 $result \leftarrow a_i$
 end if
end for

Finding the Minimum

Algorithm 2 Calculate *minimum* element in a finite integer set

Require: $\{a_1, a_2, \dots, a_i, \dots, a_n\} \in \mathbb{Z}$
Ensure: $result = \min(a_1, a_2, \dots, a_i, \dots, a_n)$
 $result \leftarrow a_1$
for $i = 2$ **to** n **do**
 if $result > a_i$ **then**
 $result \leftarrow a_i$
 end if
end for

Linear/Sequential Search (Using a For-Loop)

Algorithm 3 Locate an element x in a list of distinct values or determine that it is not in the list.

Require: $\{a_1, a_2, \dots, a_i, \dots, a_n\} \neq \emptyset; x \in \mathbb{Z}$
Ensure: $result = k$, where $(a_k = x)$ and $k \in \{1, \dots, n\}$ if the element is found; otherwise $k = -1$
 $result \leftarrow -1$
for $i = 1$ **to** n **do**
 if $result == a_i$ **then**
 $result \leftarrow i$
 end if
end for

Linear/Sequential Search (Using a While-Loop)

Algorithm 4 Locate an element x in a list of distinct values or determine that it is not in the list.

Require: $\{a_1, a_2, \dots, a_i, \dots, a_n\} \neq \emptyset; x \in \mathbb{Z}$
Ensure: $result = k$, where $(a_k = x)$ and $k \in \{1, \dots, n\}$ if the element is found; otherwise $k = -1$
 $i \leftarrow 1$
while $(i \leq n) \wedge (x \neq a_i)$ **do**
 $i \leftarrow i + 1$
end while
if $i \leq n$ **then**
 $result \leftarrow i$
else
 $result \leftarrow -1$
end if

Binary Search

Algorithm 5 Locate an element x in a list of distinct and sorted values or determine that it is not in the list.

Require: $\{a_1, a_2, \dots, a_i, \dots, a_n\} \neq \in \mathbb{Z}$, where $a_1 < a_2 < \dots < a_n$; $x \in \mathbb{Z}$

Ensure: $result = k$, where $(a_k = x)$ and $k \in \{1, \dots, n\}$ if the element is found; otherwise $k = -1$

```

 $i \leftarrow 1$ 
 $j \leftarrow n$ 
while  $i < j$  do
   $mid \leftarrow \left\lfloor \frac{i + j}{2} \right\rfloor$ 
  if  $x > a_{mid}$  then
     $i \leftarrow mid + 1$ 
  else
     $j \leftarrow mid$ 
  end if
end while
if  $x == a_i$  then
   $result \leftarrow i$ 
else
   $result \leftarrow -1$ 
end if

```

Bubble Sort

Algorithm 6 Given a list of elements of a set, sort in increasing order.

Require: $\{a_1, a_2, \dots, a_i, \dots, a_n\} \in \mathbb{R}$, $n \geq 2$

Ensure: $\{a_1, a_2, \dots, a_n\}$ where $a_1 < a_2 < \dots < a_n$

```

for  $i = 1$  to  $n - 1$  do
  for  $j = 1$  to  $n - i$  do
    if  $a_j > a_{j+1}$  then
       $temp \leftarrow a_{j+1}$ 
       $a_{j+1} \leftarrow a_j$ 
       $a_j \leftarrow temp$ 
    end if
  end for
end for

```

Insert Sort

Algorithm 7 Given a list of elements of a set, sort in increasing order.

Require: $\{a_1, a_2, \dots, a_i, \dots, a_n\} \in \mathbb{R}$, $n \geq 2$

Ensure: $\{a_1, a_2, \dots, a_n\}$ where $a_1 < a_2 < \dots < a_n$

```

for  $j = 2$  to  $n$  do
   $i \leftarrow 1$ 
  while  $a_j > a_i$  do
     $i \leftarrow i + 1$ 
  end while
   $temp \leftarrow a_j$ 
  for  $k = 0$  to  $j - i - 1$  do
     $a_{j-k} = a_{j-k-1}$ 
  end for
   $a_i = temp$ 
end for

```

Greedy Change-Making

Algorithm 8 Make x cents change with quarters, dimes, nickels, and pennies, and using the least total number of coins.

Require: $\{c_1, c_2, \dots, c_i, \dots, c_n\} \in \text{Coin Denomination}$

where $c_1 > c_2 > \dots > c_n$; x - amount of money in cents

Ensure: $result$ = minimum number of coins

```

 $result \leftarrow 0$ 
for  $i = 1$  to  $n$  do
  while  $x \geq c_i$  do
     $x \leftarrow x - c_i$ 
     $result \leftarrow result + 1$ 
  end while
end for

```

Greedy Change-Making per Denomination

Algorithm 9 Make x cents change with quarters, dimes, nickels, and pennies, and using the least total number of coins.

Require: $\{c_1, c_2, \dots, c_i, \dots, c_n\} \in \text{Coin Denomination}$

where $c_1 > c_2 > \dots > c_n$; x - amount of money in cents

Ensure: $result[n]$ = minimum number of coins per denomination

```

for  $i = 1$  to  $n$  do
   $result_i \leftarrow 0$ 
  while  $x \geq c_i$  do
     $x \leftarrow x - c_i$ 
     $result_i \leftarrow result_i + 1$ 
  end while
end for

```

Greedy Algorithm for Scheduling Talks

Algorithm 10 Suppose we have a group of proposed talks with preset start and end times. Schedule as many of these talks as possible in a lecture hall, under the assumptions that once a talk starts, it continues until it ends, no two talks can proceed at the same time, and a talk can begin at the same time another one ends.

Require: schedule $\{(s_1, f_1), (s_2, f_2), \dots, (s_n, f_n)\}$ where

s_i - start times of talks, and f_i - finish times of talks

Ensure: $result$ = set of scheduled talks

```

 $result \leftarrow \emptyset$ 
 $sort(\{(s_1, f_1), (s_2, f_2), \dots, (s_n, f_n)\})$  w.r.t. finish times
for  $i = 1$  to  $n$  do
  if  $(s_i, f_i)$  is compatible with  $result$  then
     $result \leftarrow result \cup (s_i, f_i)$ 
  end if
end for

```

Matrix Multiplication

Algorithm 11 Multiply an $m \times k$ matrix by a $k \times n$ matrix.

Require: $m \times p$ matrix $A = [a_{ij}]$; $p \times n$ matrix $B = [b_{ij}]$

Ensure: $result = m \times n$ matrix $C = [c_{ij}]$

```

for  $i = 1$  to  $m$  do
  for  $i = 1$  to  $n$  do
     $c_{ij} \leftarrow 0$ 
    for  $k = 1$  to  $p$  do
       $c_{ij} \leftarrow c_{ij} + a_{ik}b_{kj}$ 
    end for
  end for
end for

```

Boolean Product of Zero-One Matrices

Algorithm 12 Calculate the Boolean product of an $m \times k$ zero-one matrix by a $k \times n$ zero-one matrix.

Require: $m \times p$ matrix $A = [a_{ij}]$; $p \times n$ matrix $B = [b_{ij}]$

Ensure: $result = m \times n$ matrix $C = [c_{ij}]$

```

for  $i = 1$  to  $m$  do
  for  $i = 1$  to  $n$  do
     $c_{ij} \leftarrow 0$ 
    for  $k = 1$  to  $p$  do
       $c_{ij} \leftarrow c_{ij} \vee (a_{ik} \wedge b_{kj})$ 
    end for
  end for
end for

```

Closest Pair

Algorithm 13 Find the closest pair of points by computing the distances between all pairs of the n points and determining the smallest distance.

Require: $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ where $x_k, y_k \in \mathbb{R}$

Ensure: $result = \text{minimum}(\text{distance}((x_i, y_i), (x_j, y_j)))$

```

 $min \leftarrow \infty$ 
for  $i = 2$  to  $n$  do
  for  $j = 1$  to  $i - 1$  do
    if  $(x_j - x_i)^2 + (y_j - y_i)^2 < min$  then
       $min \leftarrow (x_j - x_i)^2 + (y_j - y_i)^2$ 
       $result \leftarrow ((x_i, y_i), (x_j, y_j))$ 
    end if
  end for
end for

```

REFERENCES

- [1] K. Rosen, *Discrete mathematics and its applications*, 7th ed. New York, NY: McGraw-Hill, 2012.