Chapter 7

Ch.7 Problem Solving and Algorithms
QUIZ: Match the steps in Polyga’s method to the ones in the computer method for problem solving

Devise a plan  Analysis and Specification
Look back  Implementation
Understand  Algorithm Development
Carry out the plan  Maintenance
7.2 Algorithms with simple variables

Variable = a means of storing intermediate results from one task to the next.

At the hardware level, a simple variable is just one or several adjacent Bytes in the computer memory.

Q: How many Bytes does a simple variable have in PEP/8?

3
QUIZ: In Ch.6 we distinguished between loops that can execute 0 times and loops that execute at least 1 time. Which type is this?
Extra-credit:

Implement algorithm Calculate square root in Python, using the while command.

Check the result using math.sqrt().

Due next time (Mon) at the beginning of class.

The full pseudocode is on pp.208-9 of our text.
Modify this pseudocode to print the values after initializing them.

```plaintext
integer data[20]
Write “How many values?”
Read length
Set index to 0
WHILE (index < length)
    Read data[index]
    Set index to index + 1
```
QUIZ: When the loop exits, what do we need to do?

Set position to 0
Set found to FALSE
WHILE (position < length AND NOT found )
    IF (numbers[position] equals searchItem)
        Set found to TRUE
    ELSE
        Set position to position + 1
QUIZ:
End-of-chapter question 66
**QUIZ Binary Search**

**Searching for cat**

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>Middle</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>5</td>
<td>cat &lt; dog</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>2</td>
<td>cat &lt; chicken</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>cat &lt; ant</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>cat = cat</td>
</tr>
</tbody>
</table>

Return: true

**Searching for deer**

<table>
<thead>
<tr>
<th>[0]</th>
<th>ant</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>cat</td>
</tr>
<tr>
<td>[2]</td>
<td>chicken</td>
</tr>
<tr>
<td>[3]</td>
<td>cow</td>
</tr>
<tr>
<td>[4]</td>
<td>deer</td>
</tr>
<tr>
<td>[5]</td>
<td>dog</td>
</tr>
<tr>
<td>[6]</td>
<td>fish</td>
</tr>
<tr>
<td>[7]</td>
<td>goat</td>
</tr>
<tr>
<td>[8]</td>
<td>horse</td>
</tr>
<tr>
<td>[9]</td>
<td>camel rat</td>
</tr>
<tr>
<td>[10]</td>
<td>snake</td>
</tr>
</tbody>
</table>

*Figure 7.10  Trace of the binary search*
Sorting

Arranging items in a collection so that there is an ordering on one (or more) of the fields in the items

Sort Key

The field (or fields) on which the ordering is based

Sorting algorithms

Algorithms that order the items in the collection based on the sort key

Why is sorting important?
QUIZ Selection Sort

*Show the swapped elements with arrows.*
*Show the sorted elements with shading.*

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. What are the 4 fundamental types of algorithms used to manipulate arrays?

2. What control structure is normally used to access the elements of an array?

3. Which is faster, sequential search or binary search?
   - How much faster? (use “Big-Oh” notation)

4. What is the downside of binary search?
def sum(a, b):
    return a + b

Write a Python function that multiplies three numbers
QUIZ

\[ N! = N \times (N - 1)! \]

Base case
\[ \text{Facto}(0) = 1 \quad \text{(0! is 1)} \]

General Case
\[ \text{Fact}(N) = N \times \text{Fact}(N-1) \quad \text{(for \( N \geq 1 \))} \]

Calculate:
\[ 0! = \]
\[ 1! = \]
\[ 2! = \]
\[ 5! = \]
Chapter review questions

- Describe the computer problem-solving process and relate it to *Polya’s How to Solve It* list
- Distinguish between a simple type and a composite type
- Distinguish between a void subprogram and a value-returning subprogram
- Recognize a recursive problem and write a recursive algorithm to solve it
- Distinguish between an unsorted array and a sorted array
- Describe the Quicksort algorithm
- Apply the linear search, binary search, selection sort and Quicksort to an array of items by hand
Chapter 8

Abstract Data Structures
Describe how a **word processor** can be seen from the three views:

- user/application
- logical/abstract
- implementation (in Python)
Describe how a thermostat can be seen from the three views:

• user/application
• logical/abstract
• implementation (in Python)
A stack is initially empty. Draw the stack after each of these operations:

• push(42)
• push(15)
• push(10)
• push(10)
• pop()
• pop()
• pop()
A queue is initially empty. Draw the queue after each of these operations:

- enqueue(42)
- enqueue(15)
- enqueue(10)
- dequeue()
- dequeue()
- enqueue(21)
- dequeue()
- dequeue()
QUIZ: Show the list and the outputs (if any) after each of the following operations:

moveNext()
moveNext()
getCrt()
getNext()
remove()
insert(42)
moreItems()
moveNext()
getNext()
Algorithm for Creating and Printing Items in a List

WHILE (more data)
   Read value
   Insert(myList, value)
   Reset(myList)
   Write "Items in the list are "
WHILE (moreItems(myList))
   GetNext(myList, nextItem)
   Write nextItem, '

Trick question:
Which implementation is being used (array or linked)?

Brings crt back to the head of the list
Logical Level

The algorithm that uses the list does not need to know how the data in the list is stored (array or linked), or how the various operations (Insert(), Reset(), moreItems()) are implemented!

We have written algorithms using a stack, a queue, and a list without ever knowing the internal workings, i.e. the implementation of these containers.
Write the (unique) path from the root to the node containing:
- 7
- 8
- 6
List all the nodes having:

- 0 children
- 1 child
- 2 children
QUIZ BST:
Search for item 18

**IsThere***(tree, item)***

IF (tree is null)
  RETURN FALSE
ELSE
  IF (item equals info(tree))
    RETURN TRUE
  ELSE
    IF (item < info(tree))
      **IsThere**(left(tree), item)
    ELSE
      **IsThere**(right(tree), item)
Extra-credit QUIZ:
Can you spot a similarity between this “tree search” and binary search?

```
IsThere(tree, item)
IF (tree is null)
    RETURN FALSE
ELSE
    IF (item equals info(tree))
        RETURN TRUE
    ELSE
        IF (item < info(tree))
            IsThere(left(tree), item)
        ELSE
            IsThere(right(tree), item)
```
Inserting an item in a BST: first we have to search for it!

Search for Kyrsten in this tree …
Inserting an item in a BST: first we have to search for it!

Search for Kyrsten in this tree …

… and insert the item where the search ended!
QUIZ on BST insertion

Problem 47/280:
The following elements are inserted in an initially empty BST:
50, 72, 96, 107, 26, 12, 11, 9, 2, 10, 25, 51, 16, 17, 95
Find the final tree.
Graphs: The non-uniqueness of paths generates cycles and unconnected parts

(a) Vertices: People
Edges: Siblings
QUIZ: Find 5 cycles in this graph
QUIZ: In each case, decide if the data structure is a tree or just a graph.
Graphs: directed and undirected

(b) Vertices: Cities
Edges: Direct Flights
QUIZ: Draw the adjacency matrix for this graph
Extra-credit QUIZ:
If in a graph all edges are bi-directional, what property does the adjacency matrix have?
Chapter Review Questions

• Distinguish between an array-based implementation and a linked implementation
• Distinguish between an array and a list
• Distinguish between and a unsorted list and a sorted list
• Distinguish between the behavior of a stack and a queue
• Distinguish between the binary tree and a binary search tree
Chapter Review Questions

• Draw the binary search tree that is built from inserting a series of items
• Understand the difference between a tree and a graph
• Explain the concept of subprograms and parameters and distinguish between value and reference parameters
Chapter 9

Object-Oriented Design and Programming
Do not confuse the “responsibilities” of real-life people with the “responsibilities” of the objects representing those people!

Example: the zoo-keeper

- In real-life, (s)he has the responsibility to feed the animals, clean the cages, etc.

- In an OO program: `zooKeeper.getSSN()`, `zooKeeper.getListOfAnimalsCaredFor()`, etc.
Can you think of any other useful responsibilities?  
*Hint: think of Python strings.*
Can you think on other useful responsibilities? Hint: Think about the list algorithms from Ch.8
QUIZ Translation Process (9.2)

• Explain the difference between a compiler and an interpreter.
• Is Bytecode compiled or interpreted?
• What is JVM?
• How is JVM similar to PEP/8?
Prolog – another example
(not in text)

%recursive relations
ancestor( X, Z) :- % Rule ancl: X is an ancestor of Z
parent( X, Z).

ancestor( X, Z) :- % Rule anc2: X is an ancestor of Z
parent( X, Y),
ancestor( Y, Z).

%indiv. work: define descendant

?-ancestor(tom, jim).
yes
%parent relation
parent(pam, bob).
parent(tom, bob).
parent(pam, liz).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).
parent(liz, john).
parent(liz, james).
parent(dean, john).
parent(dean, james).
parent(dorian, jim).
parent(pam, tom_jr).
parent(tom, tom_jr).
parent(bob, dan).
parent(liz, alexia).
parent(dean, alexia).

%gender facts
female(pam).
female(liz).
female(ann).
female(pat).
female(alexia).
male(tom).
male(bob).
male(tom_jr).
male(jim).
male(john).
male(james).
male(dean).
male(dorian).
male(dan).

sister( X, Y) :-
parent( Z, X),
parent( Z, Y),
female( X),
not( X = Y).

?-sister(liz, Name)
?-sister(pat, Name)
Chapter review questions

- Distinguish between functional design and object-oriented design
- Describe the stages of the object-oriented design process
- Apply the object-oriented design process
- Name, describe, and give examples of the three essential ingredients of an object-oriented language
- Understand how the constructs of top-down and object-oriented design are implemented in programming languages
Chapter 10

Operating Systems
Software Categories

**Application software**
Software written to address specific needs—to solve problems in the real world

**System software**
Software that manages a computer system at a fundamental level

*Can you name examples of each?*
Roles of an Operating System

Figure 10.1
An operating system interacts with many aspects of a computer system.

What operating systems have you used?
When a program is compiled, where is it assumed that the program will be loaded into memory? That is, where are logical addresses assumed to begin?
QUIZ: Question 48

When a program is compiled, where is it assumed that the program will be loaded into memory? That is, where are logical addresses assumed to begin?

At location 0.
QUIZ: Question 49

If, in a single contiguous memory management system, the program is loaded at address 30215, compute the physical addresses (in decimal) that correspond to the following logical addresses:

a. 9223
b. 2302
c. 7044
QUIZ : Question 49

If, in a single contiguous memory management system, the program is loaded at address 30215, compute the physical addresses (in decimal) that correspond to the following logical addresses:

a. 9223
39438
b. 2302
32517
c. 7044
37259
Partition Selection Algorithms

Which of several empty partitions should the OS allocate to a new program?

- **First fit** Allocate program to the first partition big enough to hold it
- **Best fit** Allocate program to the smallest partition big enough to hold it
- **Worst fit** Allocate program to the largest partition big enough to hold it

Why would anyone want to do this?

*Hint: Think of dynamic partitions*
WHAT BLOCK WILL BE ASSIGNED TO EACH REQUEST IF THE

- First-fit algorithm is used?
- Best-fit algorithm is used?
- Worst-fit algorithm is used?

(Treat each request as an independent event)

Requests come in for blocks of the following sizes: 1000, 25, 780, 1600, and 325
QUIZ: Paged MM

The logical address is mapped into a modified logical address of the form \(<\text{page number}, \text{offset}>\)

**Page number** = Address divided by the page size

- Integer division comes to the rescue again!

**Offset** = Remainder of the logical address divided by the page size (% operator in Python!)

The initial logical address is 42,349

The page size is 2048 (=2^{11}).

What is the modified logical address?
Paged MM

This new logical address is mapped to a physical address with the help of the page-map table (PMT)

Every program has a PMT that shows into which frame each page of the program is stored

If page size is 1024, what is the physical address of \(<2, 518>\) for:

- P1?
- P2?
The life-cycle of a Process

Process States

What can cause a process to move to the Waiting state?

Figure 10.8 The process life cycle
What is the average turn-around time?

In all examples, the 5 processes arrive in close succession at $t=0$.

This is called a Gantt chart – learn more at http://en.wikipedia.org/wiki/Gantt_chart
Shortest Job Next

<table>
<thead>
<tr>
<th>Process</th>
<th>Service time</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>140</td>
</tr>
<tr>
<td>p2</td>
<td>75</td>
</tr>
<tr>
<td>p3</td>
<td>320</td>
</tr>
<tr>
<td>p4</td>
<td>280</td>
</tr>
<tr>
<td>p5</td>
<td>125</td>
</tr>
</tbody>
</table>

What is the average turn-around time?
Round Robin

The time slice is 50 (microsec.)

What is the average turn-around time?
Are these scheduling algorithms preemptive or non-preemptive? Explain

First-Come, First-Served?

Shortest Job Next?

Round Robin?
Chapter review questions

• Describe the two main responsibilities of an operating system
• Define memory and process management
• Explain how timesharing creates the virtual machine illusion
• Explain the relationship between logical and physical addresses
• Compare and contrast the 4 memory management techniques presented
Chapter review questions

• Distinguish between fixed and dynamic partitions
• Define and apply partition selection algorithms
• Explain how demand paging creates the virtual memory illusion
• Explain the stages and transitions of the process life cycle
• Explain the operation of the 3 CPU scheduling algorithms presented
Review questions
Answer in notebook!

- 58, 59, 60
- 69, 70, 71

*Review in Wednesday’s class and lab*