CS1101S Discussion Group Week 7: *Data Structure Design & Mid-term Review*

Niu Yunpeng

niuyunpeng@u.nus.edu

October 3, 2017

Overview

Data Structure Design

Three steps to implement a program

In order to solve a problem using a program, you need:

- Think of an appropriate algorithm;
- Design a suitable data structure;
- Do the coding (with good coding style).

Data Structure Design

Thus...

The first three CS modules are:

- CS1010/CS1101S Programming Methodology
- CS2030 Programming Methodology II
- CS2040 Data Structures and Algorithms

Data Structure Design

Data structure

- In computer science, a data structure is a particular way of organizing data in a computer so that it can be used efficiently.

Algorithm

- In computer science, an algorithm is a self-contained sequence of actions to be performed.

Data Structure Design

Data & information

- Data is the storage of information.
- Two kinds of information: **states** & **procedures**.

Data structure & algorithm

- *To store states efficiently:* use **data structure**;
- *To perform procedures efficiently:* use **algorithm**.

Data Structure Design

Design principle of data structure

- Understand the requirement before doing the actual design;
- Separate the interface from the implementation;
- Compare the advantage and tradeoff;
- Principle of last commitment.

Data Structure Design

Examples of data structure so far...

- Coin change
- Symbolic differentiation
- Rational number
- Complex number
- Pair/list/tree
- Set
- ...

Data Structure Design

Common pattern of these examples

- Constructor
- Accessor
- Predicate
- Printer
- ...

Overview

1. Data structure design
   - Design principle
   - Examples
2. Sorting
3. Mid-term review
   - What we have learned
   - To prepare for the mid-term test

Sorting

Sorting algorithms so far...

- Insertion sort
- Selection sort
- Merge sort
- Quick sort

Sorting

Insertion sort

```
function insertion_sort(xs) {
    if (is_empty_list(xs)) {
        return xs;
    } else {
        return insert(head(xs), insertion_sort(tail(xs)));
    }
}
function insert(x, xs) {
    if (is_empty_list(xs)) {
        return list(x);
    } else if (x <= head(xs)) {
        return pair(x, xs);
    } else {
        return pair(head(xs), insert(x, tail(xs)));
    }
}
```

Sorting

Selection sort

```
function selection_sort(xs) {
    if (is_empty_list(xs)) {
        return xs;
    } else {
        var s = smallest(xs);
        return pair(s, selection_sort(remove(s, xs)));
    }
}
```

Sorting

Selection sort

```
function smallest(xs) {
    function sm(x, ys) {
        if (is_empty_list(ys)) {
            return x;
        } else if (x < head(ys)) {
            return sm(x, tail(ys));
        } else {
            return sm(head(ys), tail(ys));
        }
    }

    return sm(head(xs), tail(xs));
}
```

Sorting

Merge sort

```
function merge_sort(xs) {
    if (is_empty_list(xs) || is_empty_list(tail(xs))) {
        return xs;
    } else {
        var mid = middle(length(xs));
        return merge(merge_sort(take(xs, mid)),
                     merge_sort(drop(xs, mid)));
    }
}

function middle(n) {
    return math_floor(n / 2);
}
```

Sorting

Merge sort

```
function merge(xs, ys) {
    if (is_empty_list(xs)) {
        return ys;
    } else if (is_empty_list(ys)) {
        return xs;
    } else {
        var x = head(xs);
        var y = head(ys);

        if (x < y) {
            return pair(x, merge(tail(xs), ys));
        } else {
            return pair(y, merge(xs, tail(ys)));
        }
    }
}
```

Sorting

Merge sort

```
function take(xs, n) {
    if (n === 0) {
        return [];
    } else {
        return pair(head(xs), take(tail(xs), n - 1));
    }
}

function drop(xs, n) {
    if (n === 0) {
        return xs;
    } else {
        return drop(tail(xs), n - 1);
    }
}
```

Sorting

Quick sort

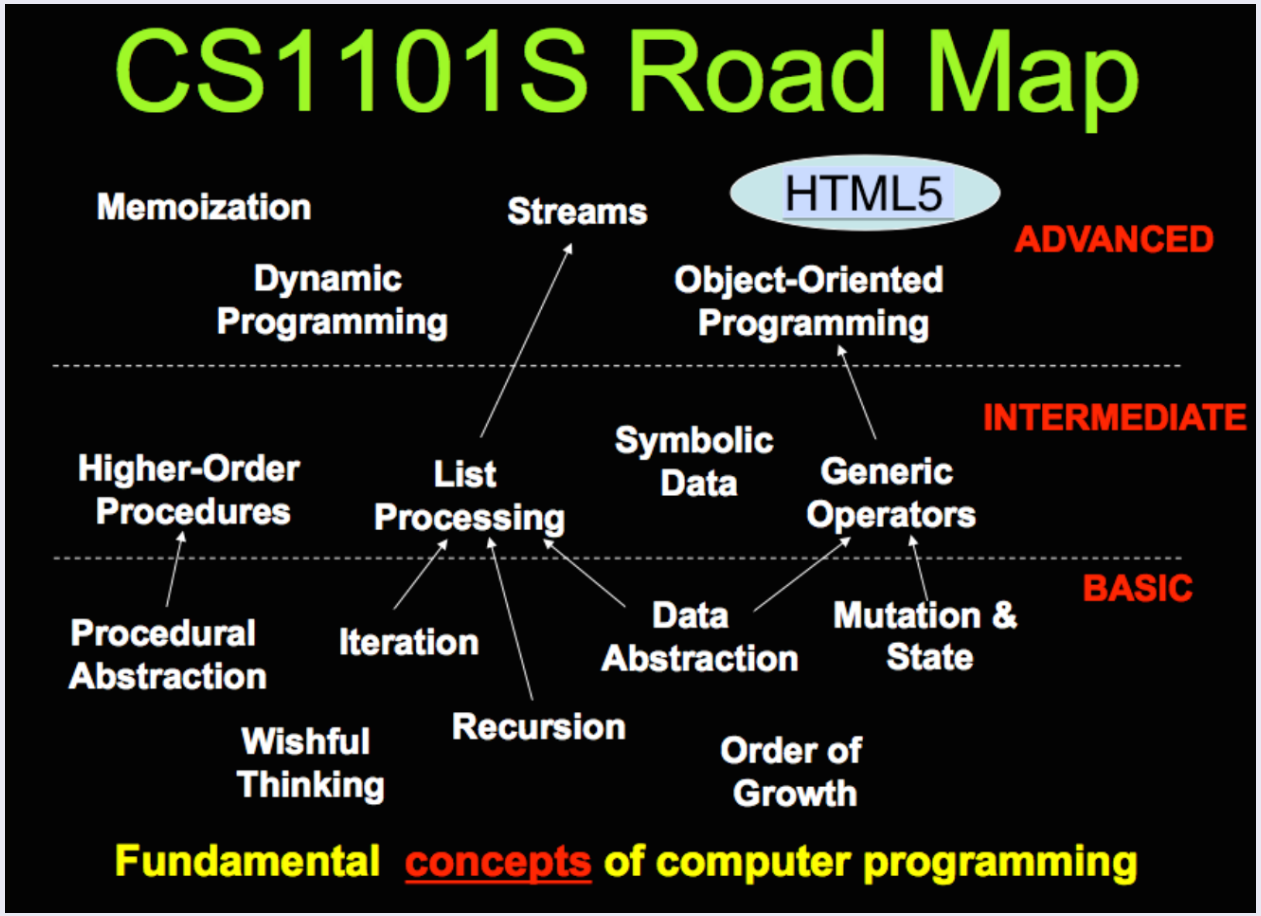
```
function quicksort(xs) {
    // Implementation
}

function partition(xs, p) {
    // Implementation
}
```

Overview

1. Data structure design
   - Design principle
   - Examples
2. Sorting
3. Mid-term review
   - What we have learned
   - To prepare for the mid-term test

Revisit the CS1101S roadmap



Mid-term Review

Things we have covered so far...

- Components of programming language
- Wishful thinking/abstraction
- Recursion/iteration
- Higher-order programming
- Pair/list/tree processing
- Data structure design
- ...

Mid-term Review

Components of programming language

- Primitives:
  The smallest constituent unit of a programming language.
- Combination:
  Ways to put primitives together.
- Abstraction:
  The method to simplify the messy combinations.
  - To abstract data: use naming;
  - To abstract procedures: use functions.
  - Sometimes, naming and functions are combined together.

Wishful thinking/abstraction

To make a good abstraction:

- Modularity:
  Separate multiple steps (and sub-steps).
- Readability:
  Easy for others to read and understand.
- Reusability:
  Provide a generic interface to be used commonly.
- Maintainability:
  Convenient to debug, refactor and deploy.

Mid-term Review

Recursion/iteration

- *Iteration*: the buttom-up approach;
- *Recursion*: the top-down approach.

How to understand recursion?

- Use ***substitution model***.
- Repeatedly replace a function call by its function body, in which the formal parameters are replaced by the respective actual arguments.

Mid-term Review

Recursive function

- Any function that calls itself (directly or indirectly) is called a recursive function.

To write recursive functions correctly

- Base case(s)
- Scale
- Sub-problem(s)

Mid-term Review

Deferred operation

- The operations that have to be suspended because they need to wait for some other operations to finish first.
- In order to suspend them, we need to remember them in the memory, which is a waste of space.

Recursive & iterative process

- Execution of a recursive function may give rise to either a recursive or iterative process.
- Recursive process: those with deferred operations.
- Iterative process: those without deferred operations.

Mid-term Review

Classical examples of recursion

- Factorial
- Square root
- Power function
- Fibonacci
- Greatest common divisor (GCD)
- Least common multiple (LCM)
- Hanoi tower
- Coin change
- Permutation / combination
- ...

Higher-order programming

Why we can do higher-order programming:

- Functions are also variables.
- They are not special.
- They just behave like normal variables.

To use higher-order programming:

- Variables can be functions.
- Parameters can be functions.
- Return values can be functions.

Mid-term Review

Pair/list/tree processing

Up to now, the list library supports different kinds of functions:

- List builder: `list`, `build_list`, `enum_list`;
- List getter: `head`, `tail`, `list_ref`, `member`, `is_member`;
- List information: `is_list`, `is_empty_list`, `length`;
- List modifier: `append`, `reverse`, `remove`, `remove_all`, `filter`, `map`, `for_each`;
- List converter: `accumulate`, `list_to_string`.

Mid-term Review

Data structure design

You should follow these principles:

- Understand the requirement before doing the actual design;
- Separate the interface from the implementation;
- Compare the advantage and tradeoff;
- Principle of last commitment.

Mid-term Review

Two types of study

- Subject-oriented: to learn the really useful stuff;
- Examination-oriented: to help you get good grades.

Consequence

- Subject-oriented: good for you (long-term goal);
- Examination-oriented: good for your CAP (short-term goal).

Mid-term Review

How to choose between two types of study

- During recess week and reading week: *examination-oriented*;
- Else: *subject-oriented*.

Suggestion

- CAP is important that it should be part of your life.
- However, it should not become all of your life.

Mid-term Review

To prepare for an examination effectively

- Read all the materials again;
- Do as many PYPs (past year papers) as possible;
- Summarize what you have learned;
- Be relaxed.

Mid-term Review

To prepare for CS1101S mid-term test

- Do all the available PYPS carefully;
- Read all lecture notes, recitation notes, discussion group notes again;
- Do all discussion group problems again;
- Be familiar with the latest Source language library;
- If you still have time, read the textbook *SICP*.

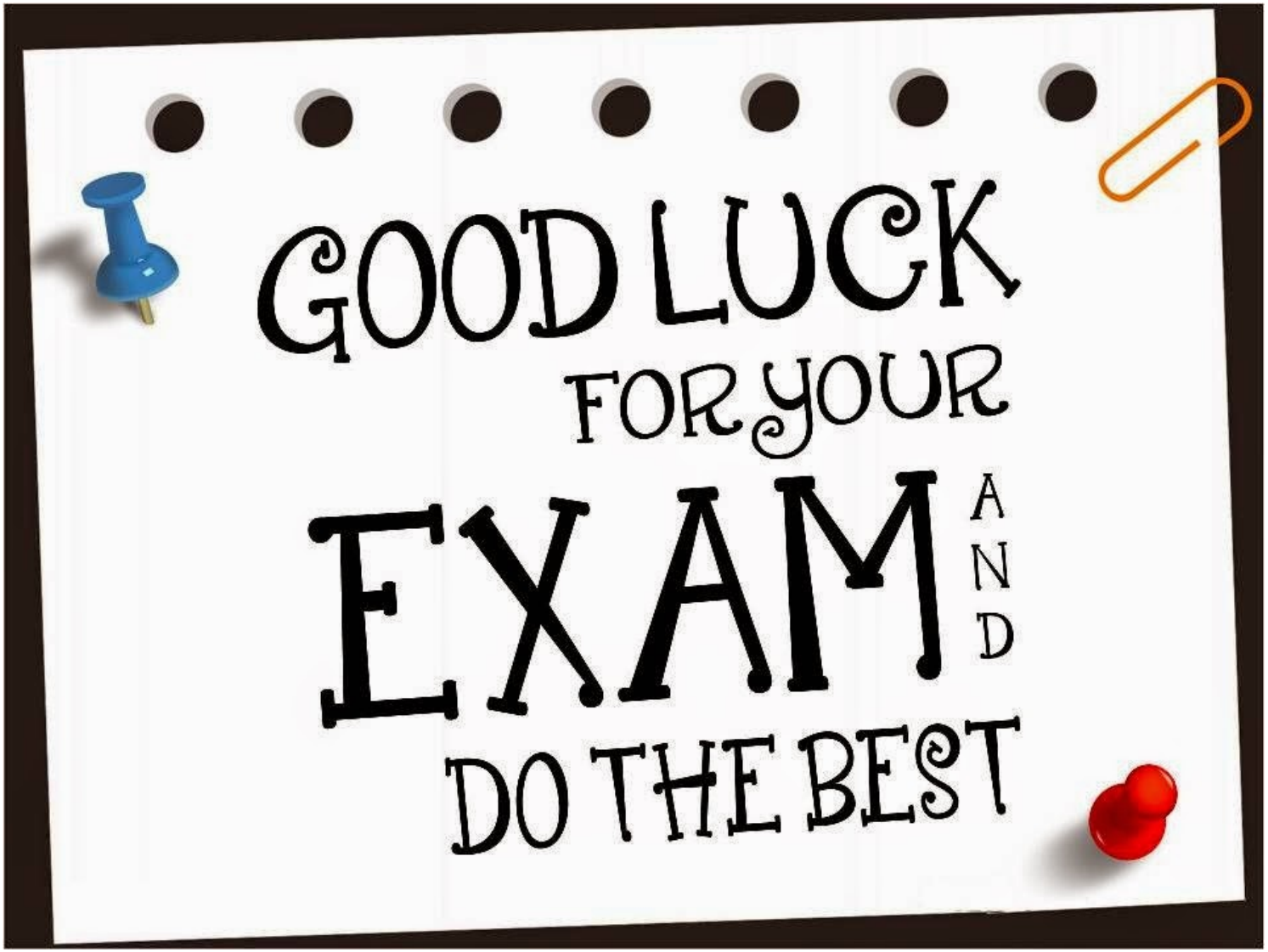
After these steps

- Don't worry anymore, you are ready for the midterm!

Good Luck

All the best for your midterm test!

Good Luck



The End