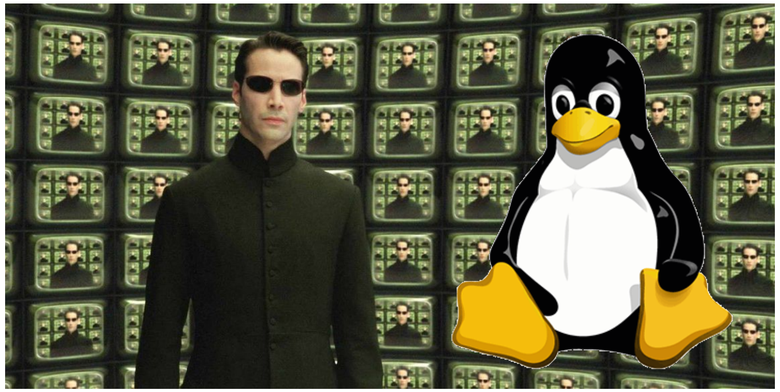
27.07.20 10:47

CSCI 125
Introduction to Computer Science and Programming II
Lecture 7: Data Structure I



Jetic Gū

2020 Summer Semester (S2)

Overview

- Focus: Data Structures
- Architecture: Linux/Unix OS
- Core Ideas:
    1. Introduction to Data Structure
    2. Lists

Why is Data Structure

Summary

What is data structure

- The way data is stored, utilised
- Data structures we've encountered so far
  - Array
  - String
  - MyList
- The way our data is stored has direct impact on how our data can be used

Concept

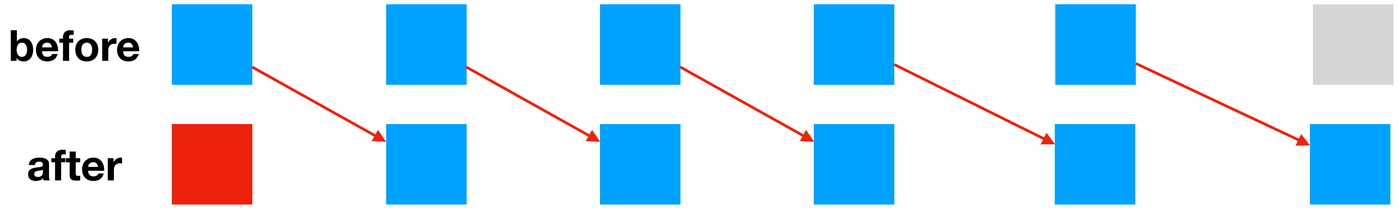
Why does it matter?

- Consider **Accessing** the k-th entry in an array or linked list
  - **Array**
    - access using index `array[k]`
    - Single step access: `array + k` -> **fast**
  - **Linked list**
    - must step through the first `k - 1` nodes in a linked list
    - `k - 1` steps access -> **slow**

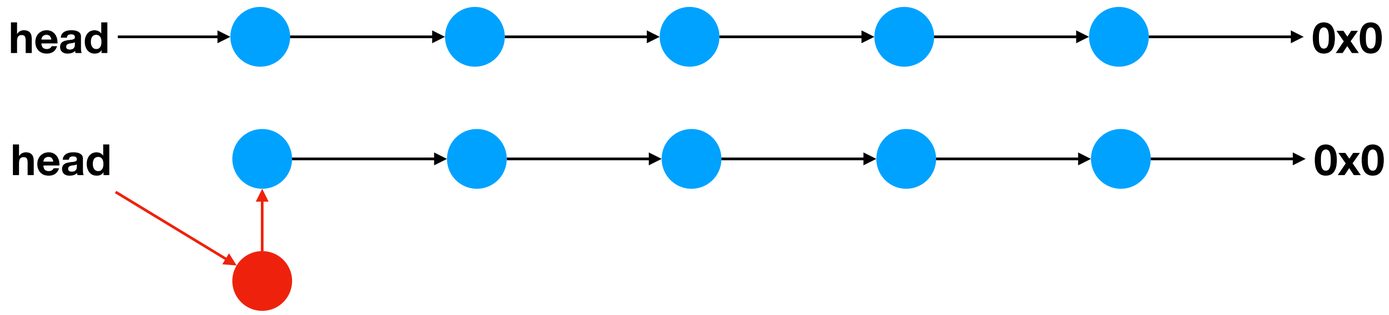
Example

Why does it matter?

- Consider **Inserting** a new entry in an array or linked list
  - **Array**: requires up to `n` copies, very **slow**



  - **Linked list**: 1 step execution, very **fast**



Example

Idea

- There is no perfect one-that-does all data structure
  - Use the one that suits your need
- Courses to take
  - CSCI 225: Data Structures and Programming
  - SFU: CMPT 307, CMPT 405, advanced Data Structure and Algorithms

Idea

- This course
  - List, Queue, Stack (This week)
  - Basic algorithm analysis techniques and implementation (Next week)

Concept

List

Linked List, Array

Summary

Definition

- an **Abstract List**
  Linearly **ordered** data entries
- Common Types
  - Array
  - Linked list

Concept

Operations

- Operations at the k-th entry of the list include:
  - Access to the object
  - Erasing an object
  - Insertion of a new object
  - Replacement of the object
  - Access to entry $k-1$ and $k+1$

Array Operations

- Operations at the k-th entry of a standard **Array** `a` include:
  - **Access** to the object, e.g.: `cout << a[k];`
    - Single step instruction, **fast**
  - **Replacement** of the object, e.g.: `a[k] = 10;`
    - Single step instruction, **fast**
  - **Access** to entry $k-1$ and $k+1$
    - Single step instruction, **fast**

Array Operations

- Operations at the k-th entry of a standard **Array** `a` include:

  - **Erasing** an object

```
for (int i=k; i<n; i++)
    a[i] = a[i+1];
```

    - Up to n steps, **slow**

  - **Insertion** of a new object

```
for (int i=n; i>k; i--)
    a[i] = a[i-1];
a[k] = newValue;
```

    - Up to n steps, **slow**

Concept

Array Operations

- **Erasing** an object

```
for (int i=k; i<n; i++)
    a[i] = a[i+1];
```

  - Up to n steps, **slow**

- **Insertion** of a new object

```
for (int i=n; i>k; i--)
    a[i] = a[i-1];
a[k] = newValue;
```

  - Up to n steps, **slow**

- How many steps will it take if: $k = 0; k = n; k \approx \frac{n}{2}$?

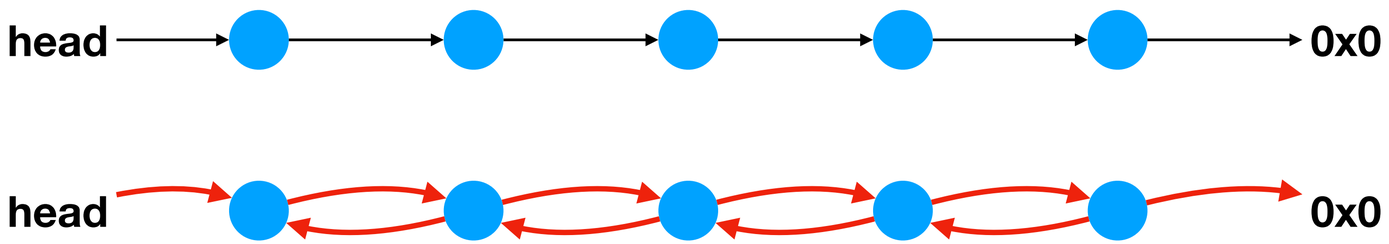
Concept

Linked lists

- The linked list we've seen in class is singly linked

```
class MyList {
    MyList *next;
    MyList *last;
};
```

  - Singly linked lists
  - Doubly linked lists



Concept

Linked lists Operations

- Operations at the k-th entry of a standard **Array** `a` include:
  - **Access** to the object,
    - `k` steps instruction, **slower than array**
  - **Replacement** of the object
    - `k` steps instruction, **slower than array**
  - **Access** to entry $k-1$ and $k+1$
    - Single step instruction, **fast**

Concept

Linked lists Operations

- **Erasing** an object
  - `k` steps to find the object, **slow**
  - 1 step to delete, **fast**
- **Insertion** of a new object
  - `k` steps to find the position, **slow**
  - 1 step to insert, **fast**

- How many steps will it take if: $k = 0; k = n; k \approx \frac{n}{2}$?

Concept

Linked lists vs Array

	Accessing			Erase			Replace			Insert		
	k=0	k=n	k=n/2	k=0	k=n	k=n/2	k=0	k=n	k=n/2	k=0	k=n	k=n/2
Array	1 step fast	1 step fast	1 step fast	n step slow	1 step fast	n/2 step slow	1 step fast	1 step fast	1 step fast	n step slow	1 step fast	n/2 step slow
List (doubly linked)	1 step fast	1 step fast	n/2 step slow	1 step fast	1 step fast	n/2 step slow	1 step fast	1 step fast	n/2 step slow	1 step fast	1 step fast	n/2 step slow

Other operations

- Concatenating two lists:
  - Linked list: can be done in 1 step
  - Array: n steps
- Memory allocation
  - Linked list: n steps
  - Array: 1 step (allocation of an entire chunk of memory is very fast)

Concept

How to choose a DS?

- Application dependent
  - e.g. always appending and prepending: linked list
  - e.g. loads of random indexed access: array
- Interviewing for a job
  - "Design an algorithm to do this..." = "Choose a data structure to do this..."

Concept

P2
List

Review: List

- Basic Operations
- Linked list (singly linked; doubly linked), and Array
- Operations and Runtime
- Choosing DS for your application is important!

Future