VNUHCM - University of Science

fit@hcmus

CSC10004 – Data Structures and Algorithms

Session 00
Course Introduction

Instructors:
Dr. Lê Thanh Tùng

Content

1. Course information
2. Grading
3. Course requirements
4. Textbooks
5. Course topics
6. Revision

Course Information

- Place: Room I91
- Time: 7h30 – 11h30 every Thursday

- Instructor: Dr. Lê Thanh Tùng (lttung@fit.hcmus.edu.vn)
- Teaching Assistant: Mr. Nguyễn Thanh Tình (nttinh@fit.hcmus.edu.vn)
- Lab teachers:
  - MSc. Nguyễn Trần Duy Minh (ntdminh@fit.hcmus.edu.vn)
  - Mr. Nguyễn Thanh Tình (nttinh@fit.hcmus.edu.vn)

Course Information

- Moodle: https://courses.ctda.hcmus.edu.vn
  - Mobile app

- This course website is used for:
  - Questions and Answers
  - Announcement
  - Course materials
  - Work submission

Grading

- Theoretical Part:
  - Class-work (exercises on theory sessions, quiz, etc): **10%**
  - Midterm Exam: **20%**
  - Final Exam: **40%**
- Practical Part:
  - The lab exams (midterm and final) will be taken on computer (programming tasks): **30%**
- Bonus: ***10%***
- **Cheating** (copies during the course): getting 0 for **the final result**.

Class Requirements | Theory Sessions

- To be on time and actively participate in class activities.
- There are some quizzes during the course.
- Prepare and use your own notebook for the course.
- Use your laptop only for the course-related purposes.
- Keep your phone in silent mode.

Class Requirements | Lab Sessions

- Follow the guidance of the teachers.
- Do not be hesitate to ask questions.
- Try your best to get as much experience as you can.
- Language: C++.
- IDE: optional. (Dev C++, g++, Visual Studio are OK)

Other notes

- **Email**: [24C10][DSA] <Your Subject>
  - Use official email always
  - If the subject's format is not suitable, your email may be ignored
- Read text-books more than the requirements.
- Get the knowledge from the videos suggested by the instructors
- **Contact:** Zalo Group (follow on Moodle page)
- **Challenge:** via Overleaf

Textbooks

- Frank M. Carrano, Timothy Henry (2013), **Data Abstraction and Problem Solving with C++: Walls and Mirrors** (Sixth Edition)
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein (2001), **Introduction to Algorithms** (Second Edition)
- Steven S. Skiena (2008), **The Algorithm Design Manual** (Second Edition)

Syllabus

- **Recursion**
- **Algorithm Efficiency**
- **Sorting Algorithms**
  - $O(n^2)$
    - Selection Sort
    - Bubble Sort
  - O(nlogn)
    - Heap Sort
    - Quick Sort
    - *Merge Sort*
  - O(n): Radix Sort
- **Priority Queue**
- **Tree structures**
  - General tree
  - Binary tree, Binary search tree
  - Balanced tree: AVL Tree, *B-Tree*
- **Graph structure**
  - Traversal
  - Shortest path
  - *Spanning tree/Minimum spanning tree*
- **Hash table**

Q&A Sessions

Question and Answer

Revision

Linked List

Question: Present the difference between the singly linked list with and without tail pointer

Linked List

Question: Present the difference between the singly linked list with and without tail pointer

Tail Pointer	Without Tail Pointer
Insertion at the End: directly	Insertion at the End: sequentially traverse
Memory Usage: Slightly higher due to the additional pointer	More memory-efficient
Use Cases: Better suited for scenarios such as queue implementations	Use Cases: Suitable for applications where end-insertions are rare or where the list is not expected to grow significantly

Question: Advantages and Disadvantages of Linked List vs Array

	Linked List	Array
Advantage	- Dynamic allocation - Flexible size - Insert Head: less - Remove Head: less	- Dynamic allocation (with new) + fixed size - Random access - Insert tail: less - Remove tail: less - Traversal: iteration
Disadvantage	Sequential traversal Insert Tail: too much, yet can improve with tail pointer Remove Tail: too much	- fixed size - Insert head: too much - Remove head: too much

Linked List

Question: Write a C++ function to find the middle node in the singly linked list (without tail pointer) in two ways

Linked List

- **Method: Two-Pointer Technique (Tortoise and Hare)**
- **Method: Count and Iterate Method**

Linked List

Method: Two-Pointer Technique (Tortoise and Hare)

```
Node* findMiddleFastSlow(Node* head) {
    if (head == nullptr) return nullptr;

    Node* slow = head;
    Node* fast = head;
    while (fast != nullptr && fast->next != nullptr) {
        slow = slow->next;
        fast = fast->next->next;
    }
    return slow;
}
```

Linked List

Method: Count and Iterate Method

```
Node* findMiddleCount(Node* head) {
    Node* current = head;
    int count = getLength(head);
    int midIndex = count / 2;
    current = head;
    for (int i = 0; i < midIndex; i++) {
        current = current->next;
    }
    return current;
}
```

Linked List

Question: Given a string s containing positive integers of any length, commas, and the characters [and] representing nested arrays, write a function that flattens this string into a one-dimensional array of integers. The prototype of this required function is

```
void flattenNestedArray(char* s, int* arr, int& size);
```

Input	Output

`s = "[12,2,[3],4]"`	`arr = [12, 2, 3, 4];`
	`size = 4`

`s = "[2,[3,[4,5]],4]"`	`arr = [2, 3, 4, 5, 4];`
	`size = 5`

Linked List

Question: What does each function F1(), F2() do? Explain your answer briefly

```
int F1(int n){
    int S = 1, i, j;
    for(i = 1; i <= n; i++){
        S = S + i*i;
    }
    return S;
}
```

Linked List

Question: What does each function F1(), F2() do? Explain your answer briefly

```
int F2(int n){
    int S = 0;
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= i; j++){
            S = S + i;
        }
    }
    return S;
}
```

Linked List

Question: Write a C++ function to implement an optimization for the function F2() so that it achieves the same functionality using only one loop

```
int F2(int n){
    int S = 0;
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= i; j++){
            S = S + i;
        }
    }
    return S;
}
```

THANK YOU

for YOUR ATTENTION