Алгоритмын шинжилгээ ба зохиомж хичээлийн лабораторийн ажил

(F.CSM301)

Д. Батмөнх

2025 оны 11-р сарын 16

Агуулга

1-р хичээл

1. Өөрийн сонгосон программчлалын 2 хэл дээр өгөгдсөн текст файлыг унших программ бич. Жич бичсэн программ болзошгүй алдаа бүрийг мэдээлдэг байна (error handling).

2. Бичил шалгалт хий (хэрхэн бичихийг хавсралтад оруулсан зааварчилгаас харна уу)

2-р хичээл

Энэ лабораторийн ажлын хүрээнд өмнөх лекцэд үзсэн алгоритмын хийсвэр кодыг өөрийн сонгосон 2 хэл дээр бичиж хэсэгчлэх аргачлалыг (divide and conquer method) бататгаж, туршилтын өгөгдлийг бэлтгэсэн энгийн текст файлаас дуудаж unit test хийнэ:

1. Insertion sort implementation

2. Merge sort implementation

3. Хоёртын хайлтыг өөрийгөө дахин дуудах аргаар бич (recursive binary search)

4. Өгөгдсөн жагсаалтаас хамгийн их утгыг буцаах алгоритм бич (хэсэгчлэх аргачлал ашиглана)

3-р хичээл

Лекцэд үзсэн ажиллах хугацааг тодорхойлох сэдвийн хүрээнд энэхүү лабораторийн ажлыг гүйцэтгэнэ:

1. Хязгаарын тодорхойлолт ашиглаад дараах тэнцэтгэлийг батал:

   - $2^{n+1} = O(2^n)$,
   - $2^{2n} = O(2^n)$

2. Дараах хос функцийг ашиглаад, $f(n)$ функцэд тохирох хязгаарыг ($O(g(n))$, $\Omega(g(n)), \Theta(g(n))$ гурвын аль нэгийг) олоорой:

   - $f(n) = \log n^2;\ g(n) = \log n + 5$
   - $f(n) = 2^n;\ g(n) = 11n^2$
   - $f(n) = n \cdot \log(n) + n;\ g(n) = \log(n)$

3. Дараах алгоритмын ажиллагаанд шинжилгээ (мөр мөрөөр) хийж, зарцуулах хугацааг (time complexity) тооцоол:

   - Зөөх эрэмбэлэлт
   - Нэгтгэх эрэмбэлэлт
   - Хоёртын хайлт

4-р хичээл

Лекцэд үзсэн ажиллах хугацааг тодорхойлох сэдвийн хүрээнд энэхүү лабораторийн ажлыг гүйцэтгэнэ:

1. Лекцэд дурдсан (25), (30)–(32)-р томьёонуудыг батал.
2. $\lg(\Theta(n)) = \Theta(\lg n)$ болохыг батал.
3. Фибоначчийн тоог $F_i = (\phi^i - \hat{\phi}^i)/\sqrt{5}$, индукцийн аргаар батал, үүнд: ϕ нь алтан харьцаа, $\hat{\phi}$ нь түүний хосмог.
4. Дараах функцүүдийг өсөлтөөр нь эрэмбэл: $\lg(\lg^* n)$, $(\lg n)^{\lg n}$, e^n, $(\sqrt{2})^{\lg n}$, $(\lg n)!$, $\lg(n!)$

5-р хичээл

Лекцэд үзсэн хэсэгчлэх сэдвийн хүрээнд энэхүү лабораторийн ажлыг гүйцэтгэнэ:

1. Дараах бодлогыг орлуулах аргаар бод:
   - $T(n) = T(n-1) + n$ тэгшитгэлийн шийд $T(n) = O(n^2)$ болохыг харуул.
   - $T(n) = 2T(n/2 + 17) + n$ тэгшитгэлийн шийд $T(n) = O(n \lg n)$ болохыг харуул.
   - $T(n) = 4T(n/2) + \Theta(n)$ тэгшитгэлийн шийд $T(n) = \Theta(n^2)$ болохыг харуул.
2. Дараах бодлогыг хэсэгчлэх аргаар бод[1]:
   - `https://leetcode.com/problems/longest-nice-substring/`
   - `https://leetcode.com/problems/super-pow/`
   - ⋆ `https://codeforces.com/problemset/problem/1490/D`

[1] ⋆ тэмдэглэгээтэй бодлогууд нь лабораторийн ажлын турш нийт 10 байх бөгөөд эдгээр нь идэвхийн 5 оноонд хамаарна

6-р хичээл

Лекцэд үзсэн хэсэгчлэх сэдвийн хүрээнд энэхүү лабораторийн ажлыг гүйцэтгэнэ:

1. Дараах бодлогыг ерөнхий аргаар бод:
   - $T(n) = 2T(n/4) + \sqrt{n}\lg^2 n$
   - $T(n) = 2T(n/4) + \sqrt{n}$
   - $T(n) = 2T(n/4) + n$
   - $T(n) = 64T(n/4) + n^3$
   - $T(n) = 64T(n/4) + 128$
2. Дараах бодлогыг хэсэгчлэх аргаар бод:
   - `https://leetcode.com/problems/maximum-subarray/`
   - ⋆ `https://leetcode.com/problems/median-of-two-sorted-arrays/`
3. Дараах бодлогыг динамик программчлалын аргаар бод:
   - `leetcode.com/problems/fibonacci-number/`
   - `https://leetcode.com/problems/climbing-stairs/`
   - `https://leetcode.com/problems/min-cost-climbing-stairs/`

7-р хичээл

Дараах бодлогыг динамик программчлалын аргаар бод (одтойгоос 1 бодлого сонгож бодно):

1. `https://leetcode.com/problems/pascals-triangle-ii/`
2. `https://codeforces.com/problemset/problem/550/C`
3. `https://leetcode.com/problems/longest-common-subsequence/`
4. `https://leetcode.com/problems/unique-binary-search-trees/`

⋆ `https://codeforces.com/problemset/problem/431/C`

⋆ `https://leetcode.com/problems/maximum-subarray/`

⋆ `https://leetcode.com/problems/longest-palindromic-subsequence/`

8-р хичээл

Дараах бодлогыг хомхойлох аргаар бод (одтойгоос 1 бодлого сонгон бодож болно):

1. `https://leetcode.com/problems/non-overlapping-intervals/`
2. `https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/`
3. `https://codeforces.com/problemset/problem/700/D`

⋆ `https://leetcode.com/problems/maximum-profit-in-job-scheduling/`

⋆ `https://leetcode.com/problems/two-city-scheduling/`

A Лабораторийн ажлын файлын зохион байгуулалт

```
F.CSM301_lab/
|-- .git/
|-- .gitignore
|-- README.md
|
|-- lab01/
|   |-- README.md
|   |-- assignment.md
|   |
|   |-- java/
|   |   |-- src/
|   |   |   \-- main/java/com/lab01/
|   |   |       \-- Main.java
|   |   |-- src/
|   |   |   \-- test/java/com/lab01/
|   |   |       \-- MainTest.java
|   |   |-- pom.xml
|   |   \-- README_java.md
|   |
|   |-- python/
|   |   |-- src/
|   |   |   \-- main.py
|   |   |-- tests/
|   |   |   \-- test_main.py
|   |   |-- requirements.txt
|   |   \-- README_python.md
|   |
|   |-- cpp/
|   |   |-- src/
|   |   |   \-- main.cpp
|   |   |-- tests/
|   |   |   \-- test_main.cpp
|   |   |-- CMakeLists.txt
|   |   \-- README_cpp.md
|   |
|   |-- data/
```

```
|   |   |-- input/
|   |   |   |-- test1.txt
|   |   |   |-- test2.txt
|   |   |   \-- ...
|   |   \-- expected/
|   |       |-- test1.txt
|   |       |-- test2.txt
|   |       \-- ...
|   |
|   |-- docs/
|   |   |-- lab01_report.md
|   |   |-- setup.md
|   |   \-- F.CSM301_Lab_Work.pdf
|   |
|   \-- scripts/
|       |-- build_all.sh
|       |-- build_java.sh
|       |-- build_python.sh
|       |-- build_cpp.sh
|       |-- run_tests.sh
|       |-- test_java.sh
|       |-- test_python.sh
|       \-- test_cpp.sh
|
|-- ...
...
```

B Linux суулгах

1. GNOME хэрэглэгчийн орчин бүхий Debian системийг татаж авна.

2. Bootable USB бэлтгэхийн тулд Etcher программыг татан суулгаад, ажиллуулна. Улмаар `debian-live-13.1.0-amd64-gnome.iso` файлаа зааж, Bootable USB бэлтгэнэ.

3. Дараа нь компьютероо дахин эхлүүлж, `BIOS/UEFI` тохиргоо уруу ороод (ихэвчлэн `F2`, `F12`, `ESC`, `DEL`), `Boot order` хэсэгт `USB`-г хамгийн түрүүнд ачаалахаар тохируулна. Хэрэв `Fast Boot` асаатай бол энэ тохиргоог унтраана.

4. Ийнхүү бэлтгэсэн USB дискээ ачаалж, Debian систем суулгах үйлдлийг эхлүүлэх бөгөөд суулгах явцад `Partition disk` хэсэгт `Install alongside Windows` гэж сонгоод, `GRUB Bootloader` суулгана.

C IDE суулгах

```
sudo apt install default-jdk -y
cd /tmp

wget -O jetbrains-toolbox.tar.gz \
"https://data.services.jetbrains.com/products/download?code=TBA&platform=linux"

tar -xvzf jetbrains-toolbox.tar.gz
sudo mv jetbrains-toolbox-*/ /opt/jetbrains-toolbox

/opt/jetbrains-toolbox/jetbrains-toolbox
```

Анх ажиллуулахад `Toolbox update` хийх бөгөөд дараа нь `Application` цэс дотор `shortcut` үүснэ. Түүнийг ашиглаад IntelliJ IDEA суулгана.

D Git ашиглах

SSH key үүсгэхийн тулд `terminal` программаа нээгээд дараах командыг биелүүлнэ:

```
sudo apt update
sudo apt install git -y

git config --global user.name "Нэрээ оруулна"
git config --global user.email "Имэйлээ оруулна"

ssh-keygen -t ed25519 -C "your_email@example.com"
cat ~/.ssh/id_ed25519.pub
```

Улмаар SSH түлхүүрийг GitHub-д нэмнэ. Дараа нь, GitHub-д шинээр repository үүсгээд түүнийгээ компьютертоо дараах байдлаар татаж авна:

```
cd ~/Documents
mkdir Projects
cd Projects
git clone https://github.com/username/my-project.git
```

Git ашиглах заавар

.gitignore

```
# ============================================================================
# OS / Editor junk
# ============================================================================
.DS_Store
Thumbs.db
ehthumbs.db
Icon?
*.swp
*.swo
*~

# VSCode / IntelliJ / Eclipse
.vscode/
.idea/
*.iml
*.ipr
*.iws
.classpath
.project
.settings/
.metadata

# ============================================================================
# Python
# ============================================================================
__pycache__/
*.py[cod]
*$py.class

# Virtual environments
.venv/
venv/
ENV/
env/
env.bak/
venv.bak/
```

```
# Packaging / build
build/
dist/
.eggs/
*.egg-info/
*.egg

# Test / coverage
.pytest_cache/
.tox/
.nox/
.coverage
.coverage.*
htmlcov/
nosetests.xml
coverage.xml
*.cover
.hypothesis/

# ============================================================================
# Java (Maven / Gradle)
# ============================================================================
# Build output
target/
**/build/
!src/**/build/

# Maven
pom.xml.tag
pom.xml.releaseBackup
pom.xml.versionsBackup
pom.xml.next
release.properties
dependency-reduced-pom.xml

# Gradle
.gradle/
!gradle-wrapper.jar
```

```
# Compiled classes / packages
*.class
*.jar
*.war
*.ear
*.nar
*.zip
*.tar.gz

# JVM crash logs
hs_err_pid*
replay_pid*

# ============================================================================
# C++ (CMake / build)
# ============================================================================
# Object / binary files
*.o
*.obj
*.so
*.dylib
*.dll
*.a
*.lib
*.exe
*.out
*.app

# CMake build dirs
CMakeFiles/
CMakeCache.txt
cmake_install.cmake
Makefile
CTestTestfile.cmake
_deps/
cmake-build-*/

# ============================================================================
# Project-specific
# ============================================================================
```

```
# Test results / reports
test_results/
test_reports/
coverage_reports/
*_report.xml
*_results.json

# Generated docs (keep /docs in repo)
docs/generated/
*_generated.pdf

# Ignore local scripts output if any
scripts/*.log
scripts/*.tmp
```

E Unit test хийх жишээ

Шалгалтад суурилсан хөгжүүлэлт буюу Test Driven Development (TDD) зарчмыг баримтлан хэрхэн код бичихийг алхам алхмаар үзүүлье.

Дараах unit testing framework ашиглана:

- C++ бол Catch2
- Java бол JUnit
- Python бол unittest

Бичил шалгалтын код бичнэ, жишээлбэл, Java хэл дээр:

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class CalculatorTest {

    @Test
    public void testAddTwoPositiveNumbers() {

        Calculator calculator = new Calculator();

        int result = calculator.add(3, 4);
        assertEquals(7, result, "3 + 4 нь 7-тэй тэнцүү байх ёстой");
```

```
    }
}
```

Бичил шалгалтын кодыг ажиллуулмагц алдаа заана, учир нь Calculator класс байхгүй. Бүтээгдэхүүний кодод энэ классыг бичнэ:

```
public class Calculator {

    public int add(int a, int b) {
        return 7;
    }
  }
```

Бичил шалгалтыг дахин ажиллуулмагц амжилттай давах болно. Шинэ бичил шалгалт нэмнэ:

```
public class CalculatorTest {

    @Test
    public void testAddTwoPositiveNumbers() {

        Calculator calculator = new Calculator();

        int result = calculator.add(3, 4);
        assertEquals(7, result, "3 + 4 нь 7-тэй тэнцүү байх ёстой");
    }

    @Test
    public void testAddPositiveAndNegativeNumbers() {

      Calculator calculator = new Calculator();
      int result = calculator.add(3, -4);
      assertEquals(-1, result, "3 + (-4) нь -1-тэй тэнцүү байх ёстой");
    }
}
```

Бичил шалгалтыг дахин ажиллуулмагц унах болно, учир нь бүтээгдэхүүний кодод тогтмол тоо 7 буцаасан байгаа. Бичил шалгалтын кодоо сайжруулна (refactoring):

```
public class Calculator {
```

```
    public int add(int a, int b) {
        return a + b;
    }
}
```

Бичил шалгалтыг дахин ажиллуулмагц давах болно. Шинэ бичил шалгалтын код нэмье:

```
public class CalculatorTest {

    @Test
    public void testAddTwoPositiveNumbers() {

        Calculator calculator = new Calculator();

        int result = calculator.add(3, 4);
        assertEquals(7, result, "3 + 4 нь 7-тэй тэнцүү байх ёстой");
    }

    @Test
    public void testAddPositiveAndNegativeNumbers() {

      Calculator calculator = new Calculator();
      int result = calculator.add(3, -4);
      assertEquals(-1, result, "3 + (-4) нь -1-тэй тэнцүү байх ёстой");
    }

    @Test
    public void testMultiplyTwoPositiveNumbers() {
      Calculator calculator = new Calculator();

      int result = calculator.multiply(3, 4);
      assertEquals(12, result, "3 * 4 нь 12-той тэнцүү байх ёстой");
    }
}
```

Бичил шалгалтыг дахин ажиллуулмагц унах болно. Шинэ бичил шалгалтын код нэмье:

```
public class Calculator {
```

```
    public int add(int a, int b) {
        return a + b;
    }

    public int multiply(int a, int b) {
        return a * b;
    }
}
```

Бичил шалгалтыг дахин ажиллуулмагц амжилттай давах болно.

F Бичил шалгалтын бусад жишээ код

Python хэл дээр:

```
import unittest


def insertion_sort(A, n):
    for i in range(1, n):
        key = A[i]
        j = i - 1
        while j >= 0 and A[j] > key:
            A[j + 1] = A[j]
            j = j - 1
        A[j + 1] = key


class TestInsertionSort(unittest.TestCase):
    def test_insertion_sort(self):
        lst = [12, 3, 7, 9, 14, 6, 11, 2]
        insertion_sort(lst, len(l))
        self.assertEqual([2, 3, 6, 7, 9, 11, 12, 14], lst)


if __name__ == "__main__":
    unittest.main()
```

Java хэл дээр:

```
// InsertionSort.java
public class InsertionSort {
    public void insertionSort(int[] A, int n) {
        for (int i = 1; i < n; ++i) {
            int key = A[i];
            int j = i - 1;
            while (j >= 0 && A[j] > key) {
                A[j + 1] = A[j];
                j--;
            }
            A[j + 1] = key;
        }
    }
}

// InsertionSortTest.java
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class InsertionSortTest {
    @Test
    public void sortArray() {
        int[] array = {12, 3, 7, 9, 14, 6, 11, 2};
        InsertionSort insertionSortTests = new InsertionSort();
        insertionSortTests.insertionSort(array, array.length);
        Assertions.assertArrayEquals(new int[]{2, 3, 6, 7, 9, 11, 12, 14}, array);
    }
}
```

C++ хэл дээр шалгахдаа

```
// CMakeLists.txt
cmake_minimum_required(VERSION 3.25)
project(InsertionSort)

Include(FetchContent)

FetchContent_Declare(
        Catch2
        GIT_REPOSITORY https://github.com/catchorg/Catch2.git
        GIT_TAG        v3.4.0 # or a later release
)

FetchContent_MakeAvailable(Catch2)

add_executable(InsertionSort main.cpp)
target_link_libraries(InsertionSort PRIVATE Catch2::Catch2WithMain)

set(CMAKE_CXX_STANDARD 20)
```

бэлтгээд

```
cmake .
make
```

командаар төсөлд шаардлагатай файлуудаа үүсгэнэ.

```
// main.cpp
#include <catch2/catch_test_macros.hpp>
#include <catch2/matchers/catch_matchers_vector.hpp>
#include <vector>

void insertionSort(std::vector<int> A, int n) {
    for (int i = 1; i < n; ++i) {
        int key = A[i];
        int j = i - 1;
        while (j >= 0 && A[j] > key) {
            A[j + 1] = A[j];
            j = j - 1;
        }
        A[j + 1] = key;
    }
}


TEST_CASE (
"Array sort"
)
 {
    std::vector<int> array = {12, 3, 7};
    insertionSort(array, sizeof(array));
    REQUIRE_THAT(array, Catch::Matchers::UnorderedEquals(std::vector<int>{3, 7, 12}));
}
```