Course Registration System

Steve Nketiah and Kate Rober

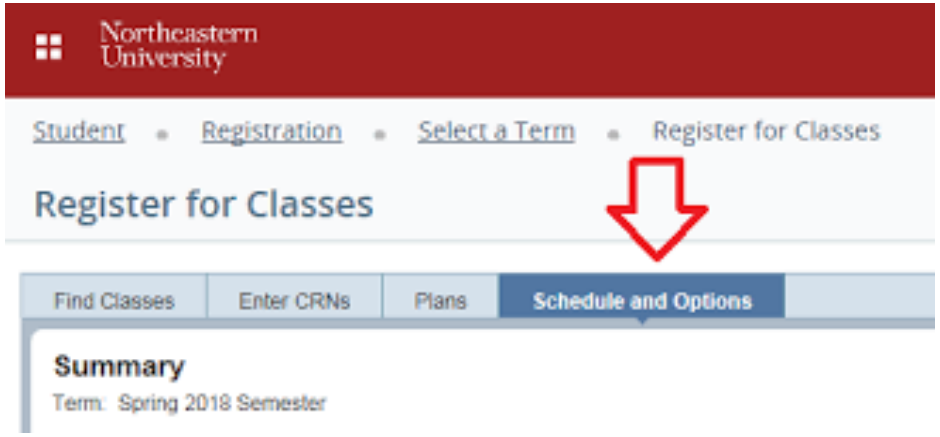
EECE 2560: Fundamentals of Engineering Algorithms

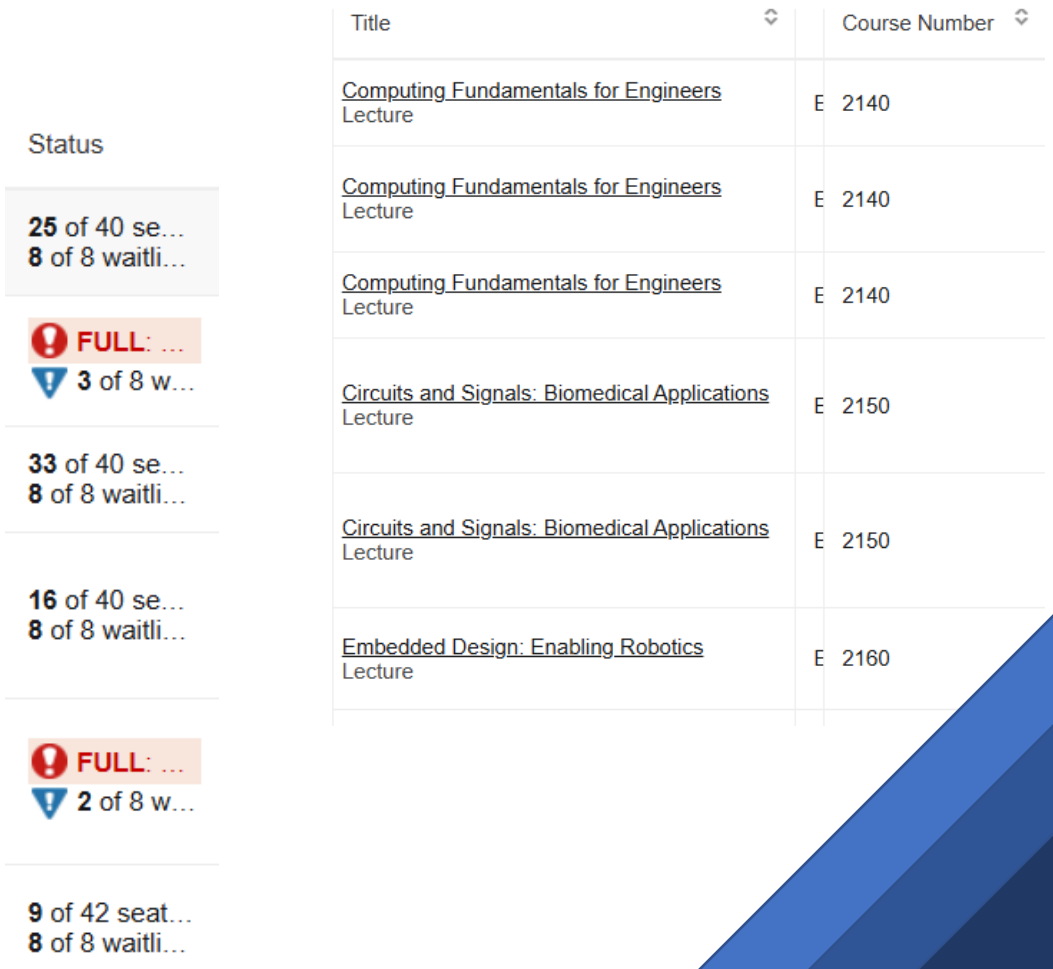
Introduction

- **Project Scope**
  - Design and develop a course registration system where the user can view a list of available courses, register for courses, join a waitlist, drop courses, display students enrolled in a course, and keep track of courses they register for
- **Objectives and Goals**
  - Allow user to efficiently register for courses
  - Have an easy-to-use menu interface
  - Create a process for randomly filling seats for user simulation
  - Record student and course information

Literature Review

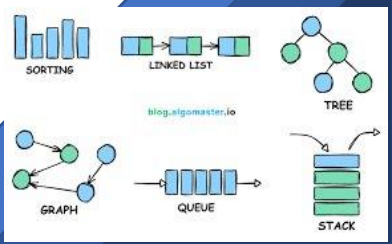
- **Northeastern course registration system**
  - Time tickets
  - Choosing from course catalogue/list
  - Course title and number
  - Course status (available seat and waitlist)





Key Methods and Techniques

- **OOP:**
  - Program utilizes classes and objects
  - Classes: Student, Node, LinkedList, Queue, and Course
- **Data Structures:**
  - Vectors, linked lists, queues, and sets were used for different features such as storing students, managing waitlists, and more
- **Dynamic Memory:**
  - Operators such as new and delete were used to dynamically allocate memory for objects and nodes
- **Menu-Driven:**
  - Provided different options to interact with the system
  - Made sure the input was valid by flagging errors and repeatedly prompting user
- **Randomization:**
  - rand() function was used to randomly enroll students in courses for better user simulation
  - Generate random student data such as names, grade level, and ID



Data Preprocessing Steps

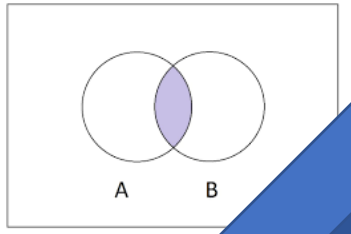
- C++ cin object is used to collect data from the user
- The data is then assigned to the appropriate variable
- Depending on the action selected, the appropriate action is performed

Data Structures

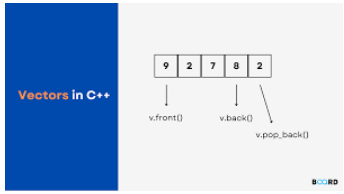
- **Linked List**
  - Manages students enrolled each course
  - Each node contains the students information (ID, Name etc)

- **Queues**
  - Manages the waitlist for courses that are full

- **Sets**
  - Ensures that elements such as IDs and course registrations are unique



- **Vectors**
  - Store the course objects created
  - Stores the names of available courses



Pseudocode - Enroll

Algorithm 1 Enroll Student in Course

```
procedure EnrollStudent(Student student)
    if enrolledCount < maxCapacity then
        enrolledStudents.insert(student)
        enrolledCount ← enrolledCount +1
        print "Enrollment successful for " student.firstName, student.lastName
    else if waitlist.size() < maxWaitlistCapacity then
        waitlist.enqueue(student)
        print "Course is full. " student.firstName, student.lastName, " added to waitlist."
    else
        print "Course & Waitlist is full"
    end if
end procedure
```

Frequency Count

- If statement: 1 operation
- insert operation: $\mathcal{O}(n)$
- enrolled count increment: 1 operations
- Print statement: 1 operation
- Enqueue function: 1 operation
- Print statement: 2 operations

Total count $= \mathcal{O}(n) + 6$
Time complexity $= \mathcal{O}(n)$

Pseudocode – Drop Student

Algorithm 3 Drop Student from Course

```
procedure DropStudent(string student_ID)
    if enrolledStudents.remove(student_ID) = true then
        print "Student with ID " student_ID, " has been unenrolled."
        enrolledCount ← enrolledCount −1
        if waitlist.size() > 0 then
            EnrollStudent(waitlist.front())
            waitlist.dequeue()
        end if
    else
        print "Student with ID " student_ID, " is not enrolled."
    end if
end procedure
```

Frequency Count

- If statement: 1 operation
- Delete operation: $\mathcal{O}(n)$
- Print statement: 1 operation
- enrolled count decrement: 1 operations
- Dequeue function: 1 operation
- Print statement: 1 operations

Total count $= \mathcal{O}(n) + 5$
Time complexity $= \mathcal{O}(n)$

Pseudocode - Display

Algorithm 3 Display All Students in the List

```
procedure DISPLAY
    current ← head
    while current ≠ null do
        print Student in the list.
        current ← current.next
    end while
end procedure
```

Frequency Count

- New node assignment: 1 operations
- While Loop comparisons: n operations
- Print statement: n operations
- current node assignment: n operations
- Return statement: 1 operations

Total count $= 3n + 2$
Time complexity $= \mathcal{O}(n)$

LIVE DEMONSTRATION

Key Findings

- Program simulates student course registration
- Courses were populated with students randomly while considering grade level for more realistic registration
- Information tracking helped manage the user's registered courses
- Program handles errors in user inputs, which ensures no bugs and ease of use for the user

Implication of Findings

- **Effective Course registration process**
  - The system can streamline the process of course registration similar to actual registration systems
- **Efficient Waitlist implementation**
  - The waitlist operations are very efficient because they are all O(1)
- **Security Concerns**
  - Brought insights to security concerns when handling student data such as full names, ID numbers, and other personal information

Project Limitations

- **Limited Dataset size**
  - This project is focused on small to medium-sized datasets
- **Lack of Concurrent User Simulation**
  - The system assumes that registration and waitlist processing occur sequentially
- **User Interface**
  - This system focuses purely on the backend logic of the registration system

Time spent

- Hours Spent : 3-5 hours per week, 12 – 20 hours per month

Conclusion

- **Basic Functionality of the system**
  - We successfully met our primary goal of using linked lists and queues for course registrations.
- **Waitlist Management**
  - The waitlist implementations were very efficient due to the use of queues.
- **User friendly interface**
  - User-friendly interface was designed that made system understandable and easy to use

Future work

- **Scalability**
  - Optimize the system's data structures and performance for a larger scale by implementing techniques such as indexing or caching
- **Support for multiple users**
  - Eventually incorporate a server for multiple users to register at the same time
- **Details**
  - Add more details to the course list such as meeting times, professor information, and locations
- **User interface and User Experience**
  - The system can be expanded to include a user-friendly GUI

References

- Linked Lists Canvas Module: Lecture 7 and Lecture 8
- https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/
- https://www.geeksforgeeks.org/c-classes-and-objects/
- https://www.geeksforgeeks.org/vector-in-cpp-stl/