Lecture 1:
Basic concepts for data structures
About the course

• This course will help students understand the basic data structures such as matrices, stacks, queues, linked lists, etc.

• BBM 203 Programming Laboratory: The students will gain hand-on experience via a set of programming assignments supplied as complementary.

• Requirements: You must know basic programming (i.e. BBM101).
References


• Data Structures Notes, Mustafa Ege.
The course web page will be updated regularly throughout the semester with lecture notes, programming assignments, announcements and important deadlines.

http://web.cs.hacettepe.edu.tr/~bbm201
Getting Help

• Office hours
  See the web page for details

• BBM 203 Programming Laboratory
  Course related recitations, practice with example codes, etc.

• Communication
  Announcements and course related discussions through
  BBM 201: https://piazza.com/hacettepe.edu.tr/fall2018/bbm201
  BBM 203: https://piazza.com/hacettepe.edu.tr/fall2018/bbm203
Course Work and Grading

• 2 midterm exams (52%)
  • Closed book and notes

• Final exam (40%)
  • Closed book
  • To be scheduled by the registrar

• Quizzes & Homeworks (8%)
  • Attempting to create false attendance (e.g., signing in the attendance list on behalf of someone else) will be punished.
  • Attendance is mandatory – students who fail to attend more than %30 of the lectures will fail from the course (≈if you do not attend 4 lectures, you will fail).
# Course Overview

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Oct 11</td>
<td>Introduction to Data Structures</td>
</tr>
<tr>
<td>2</td>
<td>Oct 18</td>
<td>Performance Analysis</td>
</tr>
<tr>
<td>3</td>
<td>Oct 25</td>
<td>Representation of Multidimensional Arrays</td>
</tr>
<tr>
<td>4</td>
<td>Nov 1</td>
<td>Band, Sparse and Triangular Arrays</td>
</tr>
<tr>
<td>5</td>
<td>Nov 8</td>
<td>Stacks and Queues</td>
</tr>
<tr>
<td>6</td>
<td>Nov 15</td>
<td>Evaluation of Expressions</td>
</tr>
<tr>
<td>7</td>
<td>Nov 22</td>
<td>Array-based Linked Lists</td>
</tr>
<tr>
<td>8</td>
<td>Nov 29</td>
<td>Midterm Exam 1</td>
</tr>
<tr>
<td>9</td>
<td>Dec 6</td>
<td>Linked Lists Applications (Stacks, Queue, Hashtables)</td>
</tr>
<tr>
<td>10</td>
<td>Dec 13</td>
<td>Doubly Linked Lists</td>
</tr>
<tr>
<td>11</td>
<td>Dec 20</td>
<td>Binary Trees, Tries</td>
</tr>
<tr>
<td>12</td>
<td>Dec 27</td>
<td>Graph Representation</td>
</tr>
<tr>
<td>13</td>
<td>Jan 3</td>
<td>Midterm Exam 2</td>
</tr>
<tr>
<td>14</td>
<td>Jan 10</td>
<td>Examples</td>
</tr>
</tbody>
</table>
• Programming assignments (PAs)
  • Four assignments throughout the semester.
  • Each assignment has a well-defined goal such as solving a specific problem.
  • You **must work alone** on all assignments stated unless otherwise.

• Important Dates
  • Programming Assignment 1  22 October 2018
  • Programming Assignment 2  5 November 2018
  • Programming Assignment 3  26 November 2018
  • Programming Assignment 4  17 December 2018
Policies

• **Work groups**
  • You must work alone on all assignments stated unless otherwise

• **Submission**
  • Assignments due at 23:59 (no extensions!)
  • Electronic submissions (no exceptions!)

• **Lateness penalties**
  • No late submission is accepted

• **Attendance**
  • Attendance is mandatory – If you do not attend 4 recitation classed, you will fail F1 grade.
  • If you do not get more than 25 points for 3 out of 4 assignments will fail with F1 grade.
Cheating

• What is cheating?
  • Sharing code: by copying, retyping, looking at, or supplying a file
  • Coaching: helping your friend to write a programming assignment, line by line
  • Copying code from previous course or from elsewhere on WWW

• What is NOT cheating?
  • Explaining how to use systems or tools
  • Helping others with high-level design issues
Cheating

• **Penalty for cheating:**
  • Suspension from school for 6 months (minimum)

• **Detection of cheating:**
  • We do check: Our tools for doing this are much better than most cheaters think!
BASIC CONCEPTS FOR DATA STRUCTURES
Digital Data

Movies

Music

Photos

Protein Shapes

DNA:
gatcttttta tttaaagcgt ctctttatta gatcttttat taggatcatg atctctctgtg
gataagtagc tatctcctag gcagatcata taattaagga gatctgttgg tggtgatcga
cgggtgtacg tattgcttat aagcgtggat cttaatggca tggattgcaac agtcactcgg
cagaatcaag gttgtatatg ggtatcttac tggttttacc tggattttaa gcaatgttat
acacattcgt tccgctcgctc ttctgctaaa tttagtaaa ttaatccaat ctttgaccca

Maps

0010101001010101010010010010101000010010010100....
Data structures

Primitive
- integer
- float
- characters
- pointers

Non-primitive
- Array
- Lists
- File

Linear
- Stack
- Queue

Non-linear
- Tree
- Graph
Digital Data Must Be ... 

- **Encoded** (e.g. 01001001 <-> 

- **Arranged**
  - Stored in an orderly way in memory / disk

- **Accessed**
  - Insert new data
  - Remove old data
  - Find data matching some condition

- **Processed**
  - Algorithms: shortest path, minimum cut, FFT, ...
Data Structures → Data StructurING

How do we organize information so that we can find, update, add, and delete portions of it efficiently?
Data Structure Example Applications

• How does Google quickly find web pages that contain a search term?

• What’s the fastest way to broadcast a message to a network of computers?

• How can a subsequence of DNA be quickly found within the genome?

• How does your operating system track which memory (disk or RAM) is free?

• In the game Half-Life, how can the computer determine which parts of the scene are visible?
Suppose You’re Google Maps…

• You want to store data about cities (location, elevation, population)…

What kind of operations should your data structure(s) support?
Operations to support the following scenario…

Finding addresses on map?
• *Lookup city by name*…

Mobile user?
• *Find nearest point to me*…

Car GPS system?
• *Calculate shortest-path between cities*…
• *Show cities within a given window*…

Political revolution?
• *Insert, delete, rename cities*
How will you count user views on YouTube?

• Lets write a userViewCount() function

```c
int userViewCount(int current_count)
{
    int new_count;
    new_count = current_count + 1;
    return new_count;
}
```

Will this implementation work all the time?
How will you count user views on YouTube?

%99.9 times yes.
How will you count user views on YouTube?

YouTube's counter previously used a 32-bit integer

YouTube said the video - its most watched ever - has been viewed more than 2,147,483,647 times.

It has now changed the maximum view limit to 9,223,372,036,854,775,808, or more than nine quintillion.
How bad can it be?

- June 4, 1996
- Ariane 5 rocket launched by the European Space Agency
- After a decade of development costing $7 Billion (~21 Billion in Turkish Liras, just for comparison Istanbul’s third bridge cost estimates are 4.5 Billion TL)
- Exploded just 40 seconds after its lift-off
- The destroyed rocket and its cargo were valued at $500 million
- Reason?
How bad can it be?

- Reason?
- Inertial reference system error: specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer.
- The number was larger than 32,767, the largest integer storable in a 16 bit signed integer, and thus the conversion failed.
- $500 Million rocket/cargo
- Time and effort
Floating Point Representation

Format of Floating points
IEEE754

64bit = double, double precision
1  11bit  52bit

32bit = float, single precision
1  8bit   23bit

16bit = half, half precision
1  5bit   10bit
## Floating Point Representation

<table>
<thead>
<tr>
<th>Nvidia Tesla Workstation GPU Performance Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
</tr>
<tr>
<td>Double Precision (FP64)</td>
</tr>
<tr>
<td>Double Precision (FP64)</td>
</tr>
<tr>
<td>Single Precision (FP32)</td>
</tr>
<tr>
<td>Single Precision (FP32)</td>
</tr>
<tr>
<td>Half Precision (FP16)</td>
</tr>
<tr>
<td>Half Precision (FP16)</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
</tr>
<tr>
<td>Memory Size</td>
</tr>
<tr>
<td>Memory Size</td>
</tr>
<tr>
<td>Release Date</td>
</tr>
<tr>
<td>Release Date</td>
</tr>
</tbody>
</table>
Goals

“I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships.”

Linus Torvalds, 2006
A data structure is a way to store and organize data in computer, so that it can be used efficiently.

Some of the more commonly used data structures include lists, arrays, stacks, queues, heaps, trees, and graphs.
What are data structures?

• Data structures are software artifacts that allow data to be stored, organized and accessed.

• Ultimately data structures have two core functions: put stuff in and take stuff out.
Why so many?

• Space efficiency
• Time efficiency:
  • Store
  • Search
  • Retrieve
  • Remove
  • Clone etc.
Choosing Data Structures

Queue vs Binary Tree

---Which one to use for what task?
Why So Many Data Structures?

- Ideal data structure:
  - “fast”, “elegant”, memory efficient

- Generates tensions:
  - time vs. space
  - performance vs. elegance
  - generality vs. simplicity
  - one operation’s performance vs. another’s

The study of data structures is the study of tradeoffs. That’s why we have so many of them!