- Course Objectives

- Strengthen knowledge of data structures and algorithms
- Extend data structures and algorithms based on requirements
- Evaluate performance of DS&A empirically
- Learn new data structures and algorithms
- Improve problem solving skills
- Improve programming skills
- Improve communication skills

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas
UT DALLAS

- You will enjoy the class if you ...

- Love programming
- Enjoy problem solving
- Appreciate the importance of the proper use of data structures
- Want to strengthen your knowledge of DS&A
- Want to improve your interviewing skills

- Knowledge you should already have

Standard data structures, such as, lists, stacks, queues, trees, priority queues, search trees, hashing, graphs

Common algorithms for sorting and searching

Design paradigms: divide and conquer, dynamic programming

Fundamental graph algorithms: DFS, BFS, Minimum spanning trees, Shortest paths, Flows

Programming skills in Java, C++, or Python; OO concepts

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas
UT DALLAS

- Your grade will depend on:

- Scores in projects, lots of them
- Excellence credits in long projects
- Performance in exams (quizzes, final exam)
- Participation in class and elearning forum

- Projects

Many programming projects will be assigned (Implementation of DS&A, Empirical study of performance of DS&A)

Study project description carefully, especially input/output specifications. Each project may have a starting code base and acceptable reference resources.

Projects are classified into short and long projects. Short projects are assigned weekly. Long projects have 2 deadlines, and they allow you to earn excellence credits. Submit before the 1st deadline to be eligible for excellence credits.

Long projects will be assigned with sample inputs and outputs. But you should create test cases on your own, trying to cover all cases.

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas
UT DALLAS

- Short Projects

Several projects will be assigned each week. They are due in 1 week. Solve as many as you can.

Short projects do not carry excellence credits.

No late submissions will be graded (without prior approval). You can submit revised or additional solutions before the deadline.

Only the final submission will be graded. If submitting revised code, include code for all problems that you solved that week.

- # Long projects

Challenging tasks, with longer deadlines (usually 2-4 weeks).

Algorithms needed for the projects are taught in class (or sometimes, you may have to learn from external resources).

Each long project has 2 deadlines.

1st deadline: only one submission, no revision is possible; First 25% of projects submitted that pass test cases, can earn up to 10 excellent credits, based on code quality, correctness, and running times. Submissions that fail test cases get no excellence credits.

2nd deadline: submissions can be revised before the deadline; can earn 100% of the points, but will not earn excellence credits.

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas
UT DALLAS

- # On code quality

1. Give meaningful names to all identifiers (names of classes, methods, variables). Organize code in each class: fields, constructors, interface methods, nested classes, helper methods, miscellaneous.
2. Format code to be easily readable. Limit no. of lines in a method to 1-2 screens.
3. Document your code with comments: e.g., input, output of functions, loop invariants, assertions, global variables, preconditions, exceptions. Follow Javadoc standard: https://en.wikipedia.org/wiki/Javadoc
4. Avoid dead code, debugging code, unneeded or unused variables, unnecessary output, "// TO DO" comments left unfilled.
5. Learn proper use of static and nonstatic methods.
6. Learn usage of public, protected, private.

- On code quality (continued)

7. Avoid code repetition (possibly due to cut-and-paste). Code duplication may indicate that a new function needs to be created to perform this common task.
8. Minimize usage of global variables. It is good to have well-named constants named globally, for all methods to use. Avoid passing return values of functions using global variables (i.e., avoid "side effects"). Follow functional programming.
9. Create test inputs (unit testing) for all methods and classes.
10. Conform to input/output specifications.
11. Use standard solutions whenever possible (such as using data structures or encryption functions from libraries, rather than implementing them on your own).
12. Use simplest data structures with the desired running times.

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas
UT DALLAS

- On code quality (continued)

13. Avoid "clever" encoding schemes: "12#25#498" to encode three integers {12, 25, 498} as return value of a function.
14. Avoid strings and hashing, except where really needed.
15. Learn proper use of generics; avoid @SuppressWarnings.
16. Develop code that is type-safe, that will not generate type errors at run-time.
17. Do not catch exceptions without writing code to handle it. It is better to declare your function as "throws SomeException" than to write "catch (SomeException e) { e.printStackTrace(); }".
18. Include names of all members of group, date of creation and changes, purpose, and other release notes in all source files. Cite sources of code that you used from outside sources.

- Groups

Projects can be done in groups of 4 students. If you have formed your own group, submit the group registration form, with names of all students in the group.

Other students will be organized into groups by the instructor.

Each group will work together on all projects, until the end of the semester. Any requests to change groups have to approved by the instructor. Seek help from instructor if you are unable to resolve problems within a group.

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas

- Grading of projects

Projects will be evaluated by (manual) code review and by executing them on (large) test cases.

Students are expected to follow good software engineering practices, and write high-quality code.

Each project should also be accompanied by a report (txt, pdf, doc) that summarizes results, cites sources.

Projects may be penalized for poorly written code.

At the discretion of the instructor or TA, you may be asked to come and show a demo of the project.

- Making an "A"

Earn at least 30 excellence credits.

Score an average of 85% or more in exams: quizzes, final.

Score 300 or more in short projects.

Score 450 or more in long projects.

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas

- Attendance

You are encouraged to attend all classes and actively participate in all discussions (in class and in the online forum on elearning).

Quizzes will be held in some classes (unannounced). No makeup quiz given to anyone who misses a quiz without prior permission.

UTD/CS department policy on attendance: one grade reduction for missing 3 classes, and a grade of "F" for missing 4 classes, without proper excuse.

- # Project submissions

All projects must be submitted on elearning. Do not submit large test data files.

Deadlines for projects will usually be at 1:59 AM on Mondays.

Each project should be submitted as a single zip or rar file. Do not use other formats like 7z. Include a readme file. Projects submitted as individual files or in other formats will not be graded.

Small projects and 2nd deadline submissions of of long projects can be revised (more than once, if needed) before their deadlines. Only the final submission before the deadline will be graded. For 1st deadline of long projects, only one submission is possible.

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas

- # Do's and don'ts

You may interact and learn from any of these sources:
- Instructor and class notes
- Students in our class and the class forum on elearning
- Any textbook on data structures and algorithms
- Lecture notes made available by instructors (world-wide)
- Wikipedia and other web sources

Don't ask for help in writing or debugging your code.

Don't share your code until after the semester is over. If you use github to store your code, set it up as a private project. Otherwise, your code is accessible to others.

Don't use code from internet (or other) sources that is not explicitly approved in the project description.

- # Honor code

All students shall maintain the highest level of academic integrity and honor.

All sources and collaborations must be acknowledged in your project reports.

Code found to be plagiarized (from other students or from web sources) will receive zero credit, and referred to the Dean of students for disciplinary action.

For more information, see the URL:
https://www.utdallas.edu/conduct/integrity/

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas
UT DALLAS

- # Cheating

Cheating in classes makes no sense. Would you pay for a new car and ask the dealer to give you an old one, instead? Would you play a game in which you have 0.01% chance of winning $1, and if you lose, you pay $1M?

Grades that you get, will have small consequences now, and no consequences in the long term. Knowledge you obtain & habits you develop, will serve you for life.

Things that are worthy in life: knowledge, good quality work, accomplishments, reputation, family, friends. A good reputation takes a lifetime to build, and one act of dishonesty to ruin it.

Don't cheat. Don't help friends cheat.