UNT
UNIVERSITY OF NORTH TEXAS

CSCE 3110
Data Structures and Algorithms

Introduction and Course Overview

Plan for Today

- Review of Course
  - Organization
  - Content

- What is an algorithm?

Instructor Information

- Self-introduction

Instructor: **Prof. Heng Fan** (please call me Heng)

Ph.D. from Stony Brook University

Email: heng.fan@unt.edu

Office: Discover Park F284

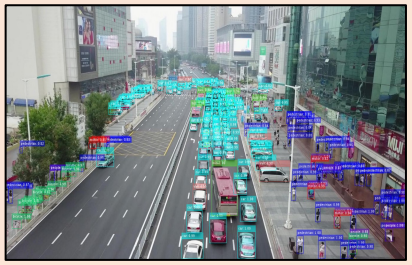
Phone: 940-565-3209

Website: https://hengfan2010.github.io/

Instructor Information

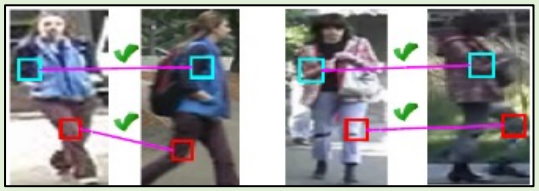
- Research interests
  - **Computer Vision, Artificial Intelligence**


Visual object tracking



Object detection

application



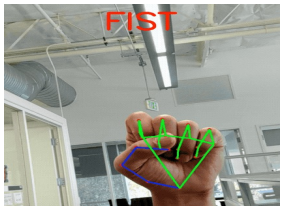
Person re-identification

Image inpainting

Semantic segmentation

Intelligent vehicles

Robotics



Human-machine interaction

Video surveillance

more …

Some pictures are form Internet.

Instructor Information

- Useful resources online
  - Convolutional Neural Networks by DeepLearningAI
    - Instructor: Prof. Andrew Ng at Stanford University, well known for deep learning
    - YouTube: https://www.youtube.com/playlist?list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF (free)
    - Course topics includes
      - Basic concepts in computer vision
      - Deep convolutional neural networks (CNNs)
      - State-of-the-art CNN models
      - Fundamental computer vision tasks, including but not limited to image classification, object detection, semantic segmentation, etc.
      - …

Focusing more on deep learning in computer vision!

Instructor Information

- Useful resources online
  - Neural Networks and Deep Learning by DeepLearningAI
    - Instructor: Prof. Andrew Ng at Stanford University, well known for deep learning
    - YouTube: https://www.youtube.com/playlist?list=PLkDaE6sCZn6Ec-XTbcX1uRg2_u4xOEky0 (free)
    - Course topics includes
      - Basic concepts in deep networks
      - Mathematical computation in deep network, including gradient descent, various regression tasks, forward-pass, back-propagation, activation functions, etc.
      - …

Focusing more on mathematics in deep learning!

Instructor Information

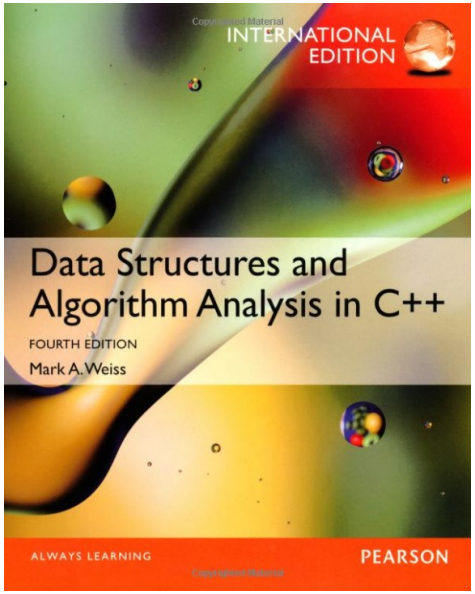
- Useful resources online
  - Implement your ideas using deep learning platforms
    - PyTorch (Facebook), using Python
      - Download: https://pytorch.org/
      - Tutorial: https://pytorch.org/tutorials/
    - TensorFlow (Google), using Python
      - Download: https://www.tensorflow.org/
      - Tutorial: https://www.tensorflow.org/tutorials
    - GPU resource
      - Google Colab
  - Research papers from conferences
    - Computer Vision: CVPR/ICCV/ECCV …
    - Machine Learning: ICLR/ICML/NeurIPS …
    - Robotics: RSS/ICRA/IROS …

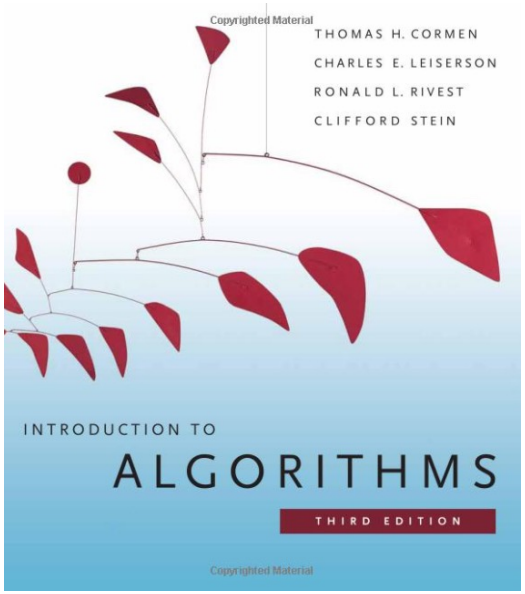
Course Webpage and Schedule

- Course webpage
  - **Canvas or https://hengfan2010.github.io/teaching/22F-3110/index.htm**
  - Course materials, including slides, assignments, quizzes, etc.
- Schedule
  - **Tuesday & Thursday: 1:00 pm – 2:20 pm**
- Office hours
  - **Thursday: 3:30 pm – 5:50 pm (using zoom or in office) or by appointment**
- TA: Xiaoqiong Liu (xiaoqiongliu@my.unt.edu)
- IA: Rahul Reddy Balabhadruni (rahulbalabhadruni@my.unt.edu)

Textbooks

- Recommended textbooks
  - Data Structures and Algorithms Analysis in C++ (4th edition), by Mark Allen Weiss
  - Introduction to Algorithms (3rd edition), by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein





Content

- Topics include:
  - Time and space complexity analysis (Asymptotic notation)
  - Recursion and Recurrence relations
  - Review of Basic Data Structures (lists, stacks, queues, etc.)
  - Tree-based data structures, including heaps, BSTs, union/find data structures and AVL trees
  - Hashing
  - Data structures for storing graphs, elementary graph algorithms (breadth-first search, depth-first search) and their applications
  - Algorithms for solving minimum spanning tree problem (Prim's algorithm and Kruskal's algorithm) and their implementations

ABET Outcomes

- After completing the course satisfactorily, a student is expected to
  - Understand time complexity of algorithms
  - Be able to solve recurrence relations
  - Understand and be able to analyze the performance of data structures for searching, including balanced trees, hash tables, and priority queues
  - Apply graphs in the context of data structures, including different representations, and analyze the usage of different data structures in the implementation of elementary graph algorithms including
    - depth-first search
    - breadth-first search
    - topological ordering
    - Prim's algorithm
    - Kruskal's algorithm
  - Be able to code the above-listed algorithms (using C++)

Prerequisites

- CSCE 2100 Computing Foundations I and CSCE 2110 Computing Foundations II.
  - You should be familiar with algorithms
    - loops, recursivity, C++ function, etc.
    - pseudo-code
  - You should know how to program without help in C++
    - design a program, compile, run experiments, etc.
    - debug a program

Tentative Schedule

Week	Lecture topic	Reading
1-2	Asymptotic Notation	Weiss, Chap. 1-2
3	Recurrence Relations	
4	Abstract Data Types, Elementary Data Structures	Weiss, Chap. 3
5-6	Trees	Weiss, Chap. 4
7-8	Priority Queue	Weiss, Chap. 6
9	Hashing (Midterm Exam)	Weiss, Chap. 5
10	Sorting	Weiss, Chap. 7
11-12	Graph Data Structures	Weiss, Chap. 9
13-14	Graph Algorithms	
15	Dynamic Programming (optional)	Weiss, Chap. 10
16	Final Exam	

This schedule may be updated in the future. Check Canvas or course webpage!

Grading

- **Quizzes:** 20%
- **Assignments** 40%
- **Midterm exam (closed book):** 15%
- **Final exam (closed book):** 20%
- **Attendance:** 5%
- **Course project (optional):** 5% (Bonus)

Grading

- (In-class) Quizzes (20%)
  - During class on TBD
  - Every two or three weeks
  - 20-30 minutes
  - May discuss solutions in office hours with instructor or TA
  - At least 5 quizzes throughout the semester

Grading

- Assignments (40%)
  - There will be five homework assignments
  - Assignments is due at the end of the day (11:59 pm)
  - Written and programing exercises (C++)
  - All programming will be in C/C++ and must compile on a Unix/Linux machine. No credit will be given for programs that do not compile

- Important notes
  - All assignments must be turned in electrically using Canvas
  - A late penalty of 10% will be applied to all late assignments for up to 3 calendar days. Assignments that are not turned in 3 days after the due date will not be accepted
  - All holidays and weekends will be counted as calendar days

Grading

- Exams (35%)
  - Midterm exam (15%)
    - during class on TBD
    - closed book
  - Final exam (20%)
    - comprehensive
    - date and time: TBD
    - closed book

Grading

- Attendance (5%)
  - Each student is allowed three absences from class for the entire semester without direct penalty to his or her grade (this does not include penalties that may result from missing in-class exams)
  - For each absence over three, a student's final grade will be reduced by 1 point, but no more than 5 points accumulated

Grading

- Course project (5%)
  - Optional
  - Individual work
  - Coding task
  - Release time TBD

Grading Scale

- Tentative grading scale (based on 100 points)

A	90-100
B	80-89
C	70-79
D	60-69
F	below 60

- No absolute grading scale; appropriate letter grade cutoffs set by instructor at the end of semester.

Academic Integrity

- Academic Integrity

  Academic Integrity is defined in the UNT Policy on Student Standards for Academic Integrity. Any suspected case of Academic Dishonesty will be handled in accordance with the University Policy and procedures. Possible academic penalties range from a verbal or written admonition to a grade of F in the course. Further sanctions may apply to incidents involving major violations. You will find the policy and procedures at: https://vpaa.unt.edu/ss/integrity.

  - Each topic discussed in class will have associated assignment
  - Students may discuss assignment problems and approaches with each other but must write their solutions individually
  - Students may not copy assignment from any source (e.g., other students, the Internet)
  - No collaboration is allowed in quizzes and exams

Do NOT Cheat!

Questions?

Course Overview

- This course provides a fun, fast-speed, modern introduction to data structures and algorithms
- We will …
  - Practice using different algorithm design techniques
  - Improve analytical skills while learning mathematical tools for algorithm analysis
  - Learn fundamental algorithms and data structures
  - Learn the inner workings of some of the algorithms that currently have the most impact in practice
  - Improve the coding skills

Why Learn Algorithms?

- Algorithms are the heart and sole of computing

- Algorithmic computing now plays a key role in nearly everything proficiency opens many doors

A Good Algorithm (or Data Structure)

- Always terminates and produces correct output
  - A "close enough" answer is sometimes fine
  - Some types of randomized algorithms can fail, but only with miniscule probability
- Makes efficient use of computational resources
  - Minimize running time, memory usage, processors, bandwidth, power consumed, heat produced
- Is simple to describe, understand, analyze, implement, and debug

Robot Tour Optimization

Suppose you have a robot arm equipped with a tool, say a soldering iron. To enable the robot arm to do a soldering job, we must construct an ordering of the contact points, so the robot visits (and solders) the points in order.

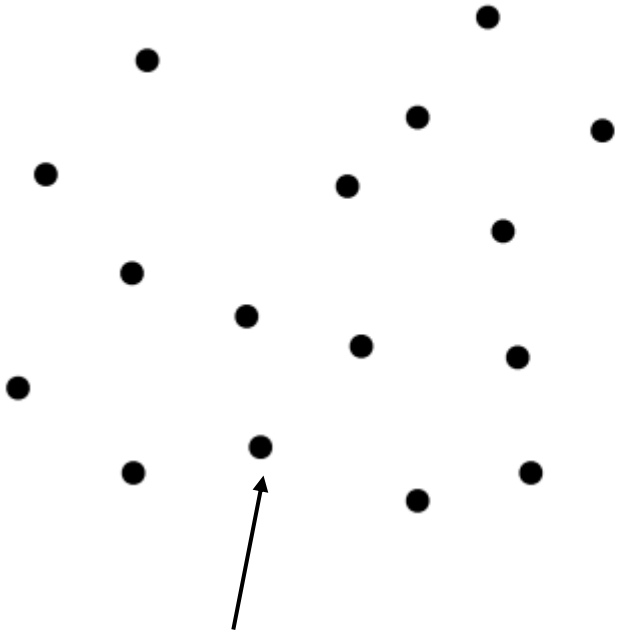
We seek the order which minimizes the testing time (i.e., travel distance) it takes to assemble the circuit board.

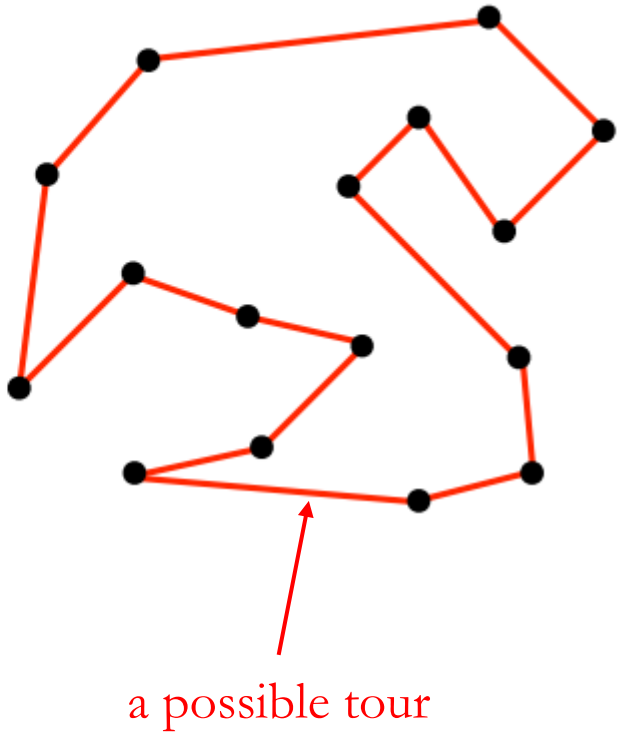
Credits to S. Skiena at SUNY Stony Brook for this example.

Find the Shortest Robot Tour

- You are given the job to program the robot arm. Give me an algorithm to find the best tour!

  (the distance between any pair of contact points is known)





Exhaustive Search

- We could try all possible orderings of the points, then select the one which minimizes the total length.

$d = \infty$
For each of the $n!$ permutations Π_i of the n points
    If $(cost(\Pi_i) \leq d)$ then
        $d = cost(\Pi_i)$ and $P_{min} = \Pi_i$
Return P_{min}

Since all possible orderings are considered, we are guaranteed to end up with the shortest possible tour.

Exhaustive Search is Slow!

- Because it tries all $n!$ permutations, it is much too slow to use when there are more than 10-20 points.

$d = \infty$
For each of the $n!$ permutations Π_i of the n points
    If $(cost(\Pi_i) \leq d)$ then
        $d = cost(\Pi_i)$ and $P_{min} = \Pi_i$
Return P_{min}

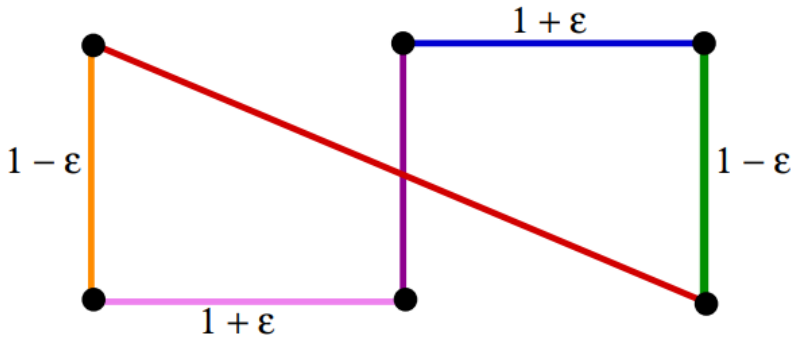
Closest Pair Tour

- One idea is to repeatedly connect the closest pair of points whose connection will not cause a cycle or a three-way branch, until all points are in one tour.

Let n be the number of points in the set
$d = \infty$
For $i = 1$ to $n - 1$ do
    For each pair of endpoints (x, y) of partial paths
        If $dist(x, y) \leq d$ then
            $x_m = x$, $y_m = y$, $d = dist(x, y)$
    Connect (x_m, y_m) by an edge
Connect the two endpoints by an edge.

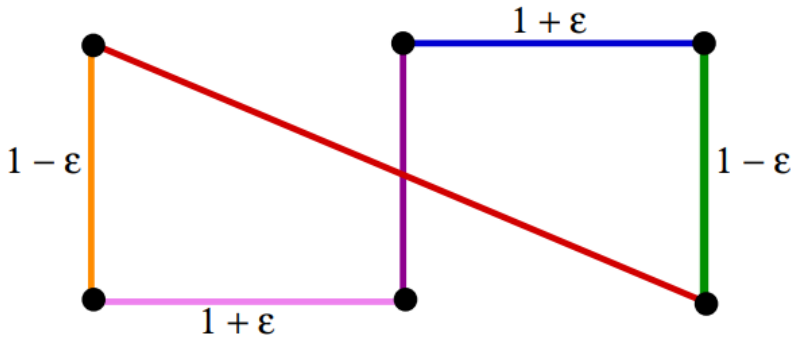
Closest Pair Tour is Wrong!

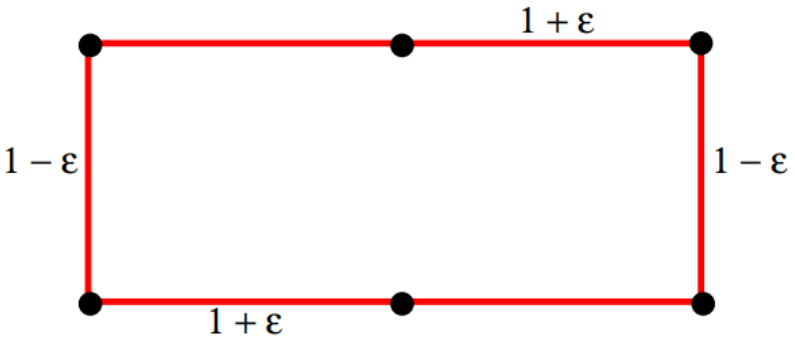
- Although it works correctly on the previous example, other data causes trouble:



Closest Pair Tour is Wrong!

- Although it works correctly on the previous example, other data causes trouble:

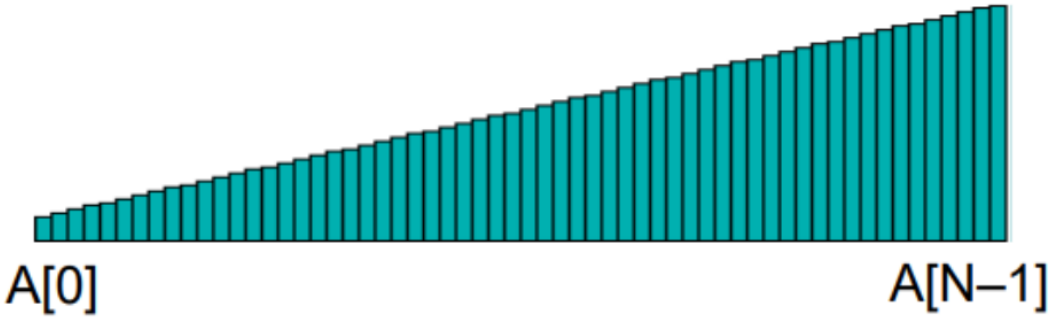




shortest tour.

No efficient, correct algorithm exists for the *traveling salesman problem*, as we will see later.

Example: Searching a Sorted Array



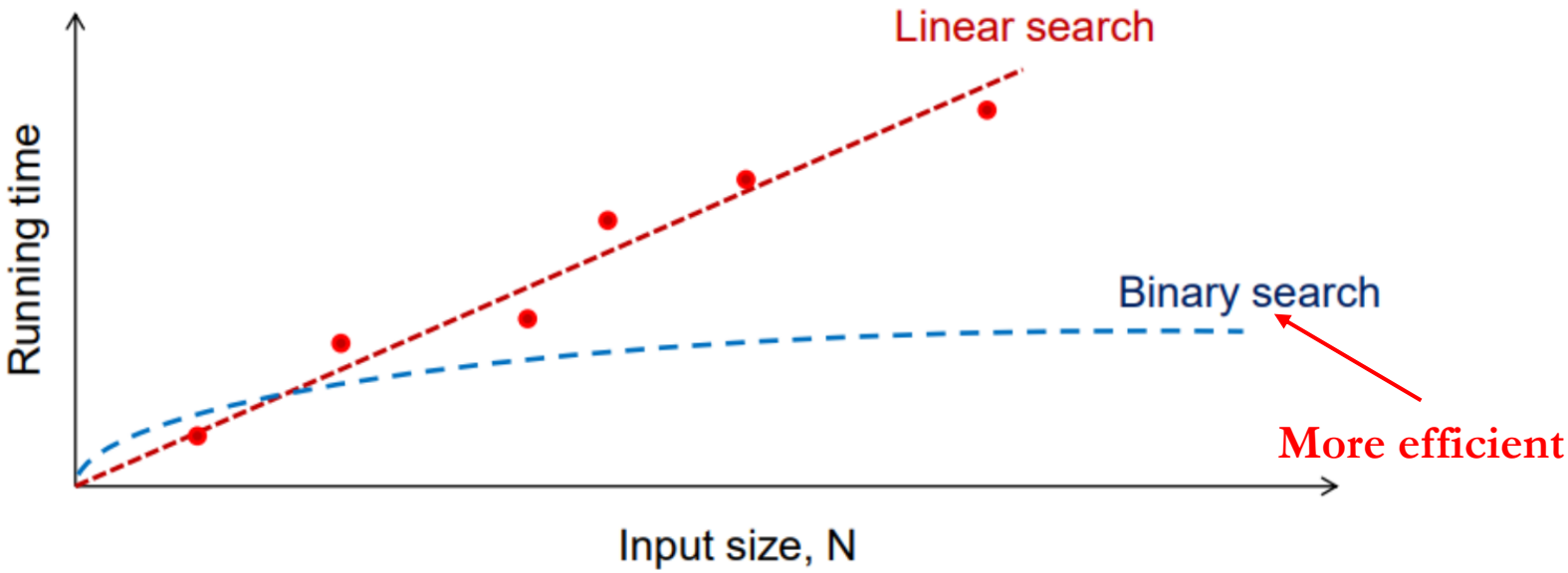
- Linear search runs in N "steps" in the worse case

```
for (i=0; i < N; i++)
      if (target == A[i]) {found it!}
```

- Binary search: $\leq \log_2 N$ "steps" in worst case

```
low = 0; high = N-1;
while (low <= high) {
      mid = (low + high) / 2;
      if (target == A[mid]) { found it! }
      if (target > A[mid]) low = mid+1;
      else high = mid-1;
}
```

Empirical Performance Testing



Choose data structures carefully, since some are much more efficient than others!

Next Class

Algorithm Analysis I

Reading: Weiss, chap. 2