4190.204

Data Structures

Fall 2018

Instructor

- Srinivasa Rao Satti
- Office: 301 – 401
- Email: ssrao@tcs.snu.ac.kr
- Office hours: Tue, Thu 1300-1400 or by appointment (email me)

- TA: Mohammad Sadegh Najafi
- Email: ds2018ta@tcs.snu.ac.kr

Course web site

- On etl.snu.ac.kr

  Contains

    Schedule, lecture slides, assignments/homework, and other course related information

Text books

- Data Structures, Algorithms, and Applications in Java, Second Edition

  by Sartaj Sahni, Silicon Press, 2005.

or

- Data Abstraction and Problem Solving with Java, Third Edition

  by Janet Prichard and Frank M. Carrano, Pearson, 2011.

Prerequisite

- Java programming 

Evaluation

- Assignments – 30%
- Midterm – 30%
- Final exam – 30%
- **Surprise** quizzes – 5%
- Lab Participation – 5%
- Assignments – programming problems + written exercises

Goals

- Understanding how data structures impact the performance of programs
- Solving computational problems using some basic data structures like lists, stacks, trees, graphs etc.
- Designing and writing large programs using object-oriented design principles

What the course is about

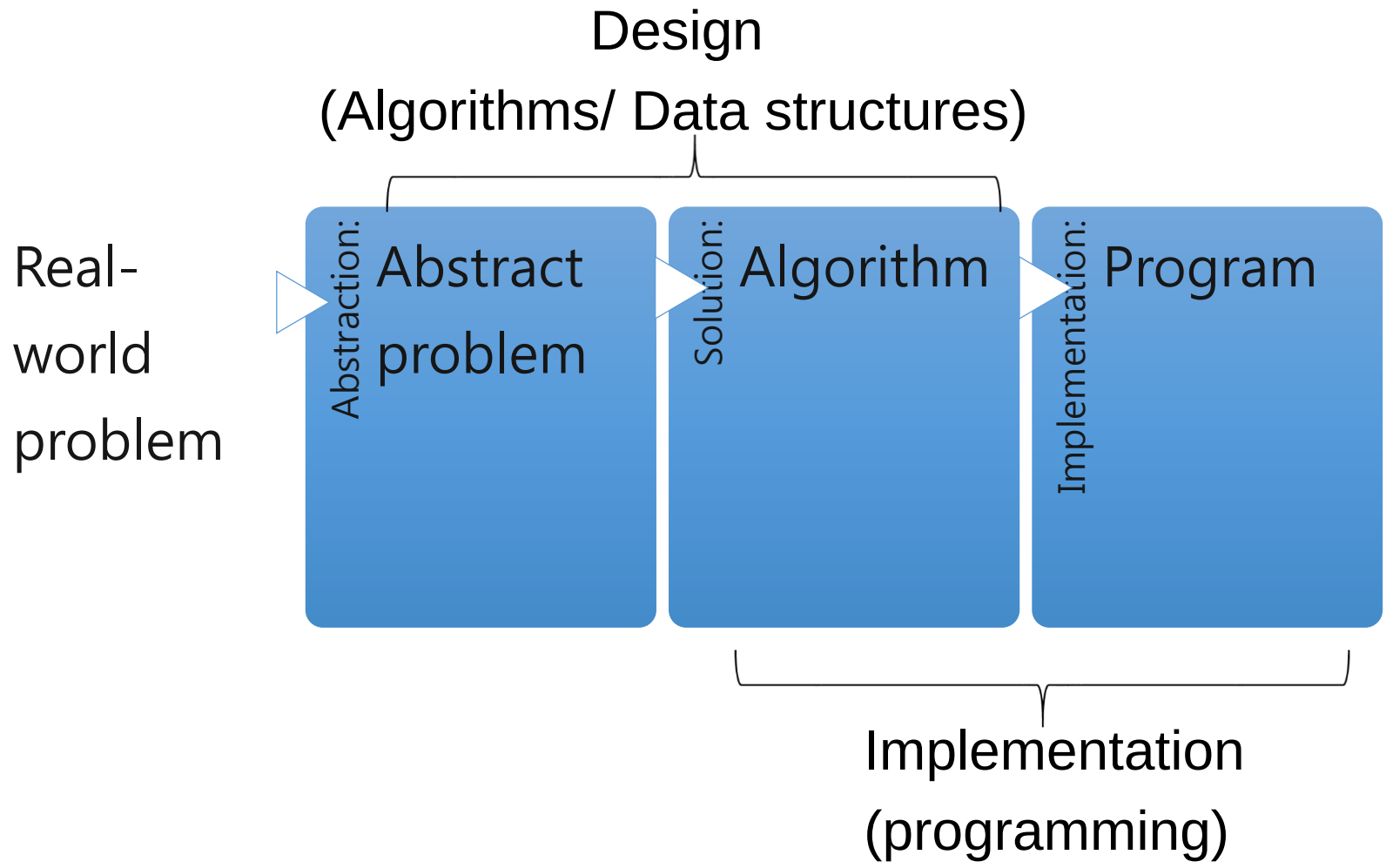
- Data structures is concerned with the representation and manipulation of data.
- All programs represent data in some way and manipulate the data.
- Data is represented in a `data structure' and manipulated through an `algorithm'.
- The study of data structures and algorithms is fundamental to Computer Science.

- Algorithm: the essence of computational procedure – step by step instructions

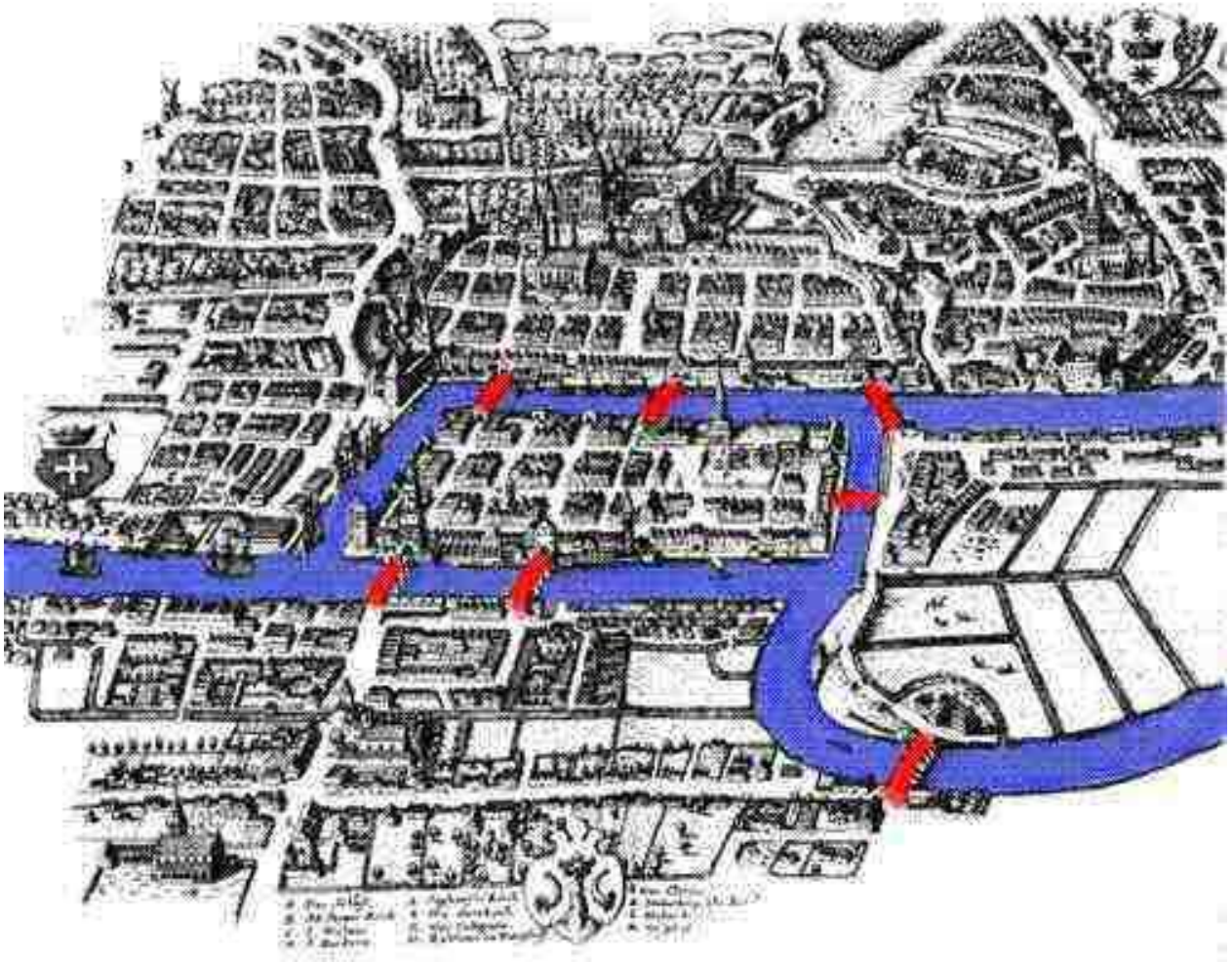
- Program: an implementation of an algorithm in some programming language

- Data structure: organization of data needed to solve the problem

Problem solving

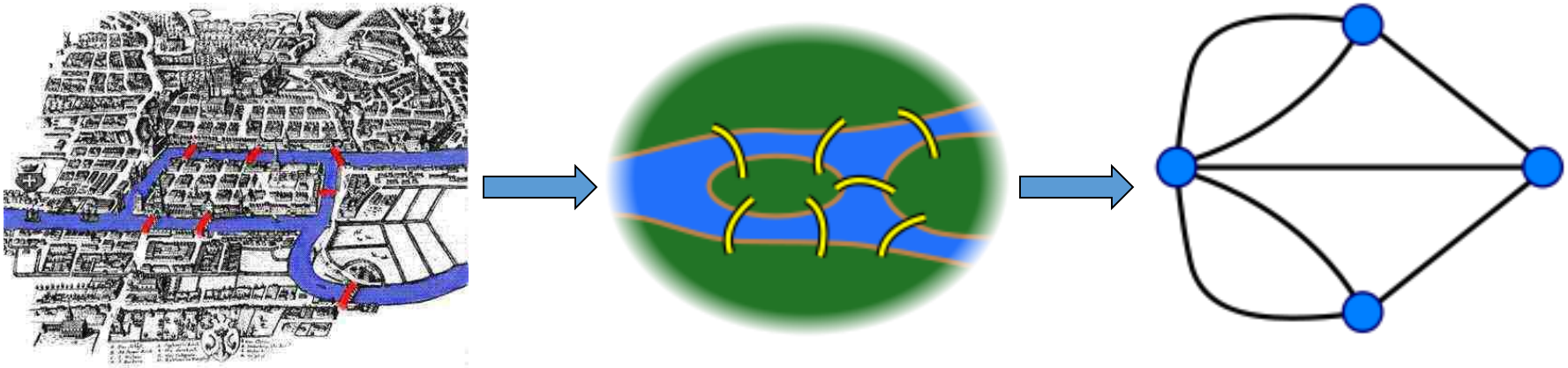


Konigsberg bridge problem



Konigsberg bridge problem

- Abstraction



No Algorithm !

Other (algorithmic) problems

- Show all possible ways of going from place A to place B by crossing at most 4 bridges
- Fastest (shortest) way to get from A to B
- Each problem may require the data to be represented in a different form

A simple problem

- Given a set of processes and their waiting times, return them in the decreasing order of their waiting times.

- Abstract problem: Given a set of numbers, sort them decreasing (or equivalently in increasing) order.

Sorting algorithm

- Let's use Insertion Sort
  - Start with a sequence of size 1
  - Insert the remaining elements, one at a time
  - Example: Sort 7, 3, 5, 6, 1
    - Start with 7 and insert 3 => 3, 7
    - Insert 5 => 3, 5, 7
    - Insert 6 => 3, 5, 6, 7
    - Insert 1 => 1, 3, 5, 6, 7

Insertion sort implementation

```
public static void insertionSort(int [] a) {
  for (int i = 1; i < a.length; i++) {
  // insert a[i] into a[0:i-1]
        int t = a[i];
        int j;
        for (j = i - 1; j >= 0 && t < a[j]; j--)
         a[j + 1] = a[j];
        a[j + 1] = t; }
  }
```

Performance measures/ Complexity

- Space/Memory
- Time
  - Count a particular operation
  - Count number of steps
  - Asymptotic complexity

Comparison Count

```
for (int i = 1; i < a.length; i++) {
  int t = a[i];
        int j;
        for (j = i - 1; j >= 0 && t < a[j]; j--)
         a[j + 1] = a[j];
         a[j + 1] = t;
}
```

Comparison Count

- Pick an instance characteristic, n
  - Eg. n = a.length for insertion sort

- Determine count as a function of this instance characteristic.

  The exact number of comparisons depends on the instance **a**, and varies greatly from instance to instance.

Comparison Count

- Worst-case count = maximum count
- Best-case count = minimum count
- Average count over all instances.

Worst-Case Comparison Count

```
for (j = i - 1; j >= 0 && t < a[j]; j--)
        a[j + 1] = a[j];
```

a = [1, 2, 3, 4] and t = 0 => 4 comparisons

a = [1, 2, 3,…,i] and t = 0 => i comparisons

Worst-Case Comparison Count

```
for (int i = 1; i < n; i++)
    for (j = i - 1; j >= 0 && t < a[j]; j--)
        a[j + 1] = a[j];
```

total comparisons = 1 + 2 + 3 + … + (n-1) = (n-1)n/2

(in the worst case)

Insertion sort

- What is the best case?
  - #Comparisons?
- Average case?
  - #Comparisons?

Counting comparisons

- The worst case number of comparisons performed by Insertion Sort can be reduced significantly by using `Binary Insertion Sort'
- But the overall running time doesn't improve significantly
- Counting only comparisons is not enough

What about counting `data movement'?

Sorting out sorting

- http://www.youtube.com/watch?v=YvTW7341kpA