AAI2007 Introduction to Algorithms

Week 1: Course Introduction

Instructor: Jinyoung Han (jinyounghan@skku.edu)

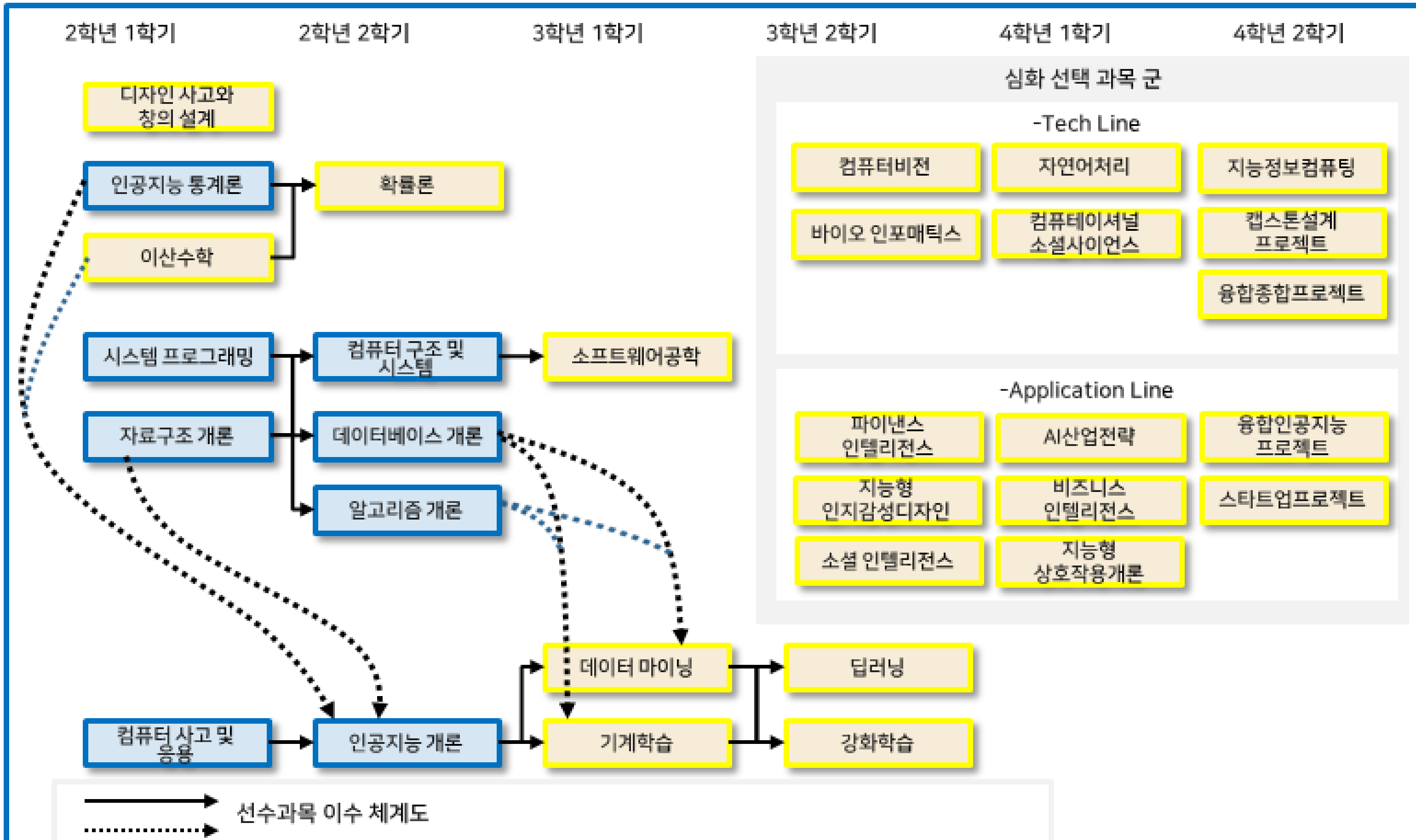
In This Lecture

Outline

1. Course Logistics
2. Algorithm Basic
3. Level Test
4. Programming Environment Setting

01. Course Logistics

Course Position



*유의 사항
- 2019년도 2학기 교육과정 개편에 따라, 인포매틱스 기초 → 시스템 프로그래밍으로 변경되었습니다.

01. Course Logistics

Course Info

- Instructor: 한진영 교수
  - Office: 국제관 313호
  - Tel: 02-740-1797
  - Email: jinyounghan@skku.edu
  - Office hour: Wed. 13:00-15:00
    - Appointment by email

- TAs
  - 김지나: jinakimcool@gmail.com
  - 강지원: kanjiwon@gmail.com
  - 윤지우: yoonjeewoo@gmail.com

01. Course Logistics

Course Overview

- Goals
  - Understanding basic computer algorithms
  - Hands-on experience on solving problems with programming

- Course Structure
  - Lecture (6:00pm~7:30pm)
  - Practice (7:30pm~9:00pm)
    - PC or Notebook

- Prerequisite
  - Data Structure, Programming
    - C or Python?

01. Course Logistics

Grading Policy

- Scores breakdown
  - Lecture attendance (10)
  - Quiz or Homework (10)
  - Practice (20)
  - Midterm/Final (50)

01. Course Logistics

Resources

- Lecture slides
  - Will be uploaded to the system
  - All the information will be given in the system, so please check the system frequently

- Reference material
  - Introduction to Algorithms
  - 쉽게 배우는 알고리즘
  - 뇌를 자극하는 알고리즘

01. Course Logistics

No Cheating

- No submission is better than cheating
  - Zero Tolerance Cheating Policy
- Definition of cheating in this class
  - Knowingly or unknowingly participating in the submission of unoriginal work for any test (e.g., lab exercise, project)
  - Answer to roll-call instead of another student is also regarded as cheating
- Penalty
  - Assign a fail grade

01. Course Logistics

Tentative Schedule

수업일	내용
9/4	Course Introduction, Algorithm Basic, Level Test
9/11	Order of Complexity, List
9/18	Stack, Queue
9/25	Tree, Binary Search Tree (BST), 건학 기념일
10/2	Priority Queue, Heap
10/9	Graph (1), 한글날
10/16	Graph (2)
10/23	Sorting
10/30	Searching
11/6	Midterm
11/13	Hash Table
11/20	Dynamic Programming (1)
11/27	Dynamic Programming (2)
12/4	Greedy Algorithms
12/11	NP-completeness
12/18	Final

Algorithm Basic

Algorithm

- An algorithm: a set of instructions, typically to solve a class of problems or perform a computation
- Typical design (or problem solving methods) of algorithms
  - divide-and-conquer, dynamic programming, greedy approach, backtracking, branch-and-bound
- Algorithm Analysis
  - Efficiency analysis through computational complexity
  - Lower bounds for sorting and searching problems
  - Intractability (NP-completeness)

02. Algorithm Basic

Algorithm Description

- Natural languages
- Programming languages
- Pseudo-code
  - similar, but not identical, to C++/Java.
  - Notable exceptions: unlimited array index, variable-sized array, mathematical expressions, use of arbitrary types, convenient control structure, and etc.
    - E.g., low <= x && x <= high -> low <= x <= high
    - E.g., temp = x; x = y; y = temp -> exchange x and y

An Example

- Title: Sequential Search
- Problem
  - Is the key x in the array S of n keys?
- Inputs (parameters)
  - Positive integer n, array of keys S indexed from 1 to n.
- Outputs
  - The location of x in S. (0 if x is not in S.)
- Algorithm
  - Starting with the first item in S, compare x with each item in S in sequence until x is found or S is exhausted. If x is found, answer yes; if x is not found, answer no.

An Example

- Pseudo-code

```
void seqsearch(int n, // Input(1)
               const keytype S[], // (2)
               keytype x, // (3)
               index& location) // Output
{
       location = 1;
       while (location <= n && S[location] != x)
               location++;
       if (location > n)
               location = 0;
}
```

An Example

- How many comparisons for searching x in S?
  - Depends on the location of x in S
  - In worst case, we should compare (n) times.
  - In best case,
- Any better algorithm to get the same solution?
  - No, we can’t, unless there is an extra information in S.

An Example

- Next approach: Binary Search
- Problem
  - Is the key x in the array S of n keys?
  - Determine whether x is in the **sorted array S** of n keys.
- Inputs (parameters)
  - Positive integer n, sorted (non-decreasing order) array of keys S indexed from 1 to n
- Outputs
  - The location of x in S. (0 if x is not in S.)

An Example

- Pseudo-code

```
void binarysearch(int n, // Input(1)
                  const keytype S[], // (2)
                  keytype x, // (3)
                  index& location) // Output
{
        index low, high, mid;
        low = 1; high = n;
        location = 0;
        while (low <= high && location == 0) {
                mid = (low + high) / 2; // integer div
                if (x == S[mid]) location = mid;
                else if (x < S[mid]) high = mid – 1;
                else low = mid + 1;
}
```

An Example

- How many comparisons for searching x in S?
  - S is already sorted. We know it.
  - For each statement in a while-loop, the number of searching targets will be half.
  - In worst case, we should compare ($\log_2 n + 1$) times.
    - e.g. n = 32. S[16], S[16+8], S[24+4], S[28+2], S[30+1]. S[32]
  - In best case, trivial to answer.
- Any better algorithm to get the same solution?

An Example

- Number of comparisons done by sequential search and binary search when x is larger than all the array items

Array Size	Number of Comparisons by Sequential Search	Number of Comparisons by Binary Search
128	128	8
1,024	1,024	11
1,048,576	1,048,576	21
4,294,967,296	4,294,967,296	33

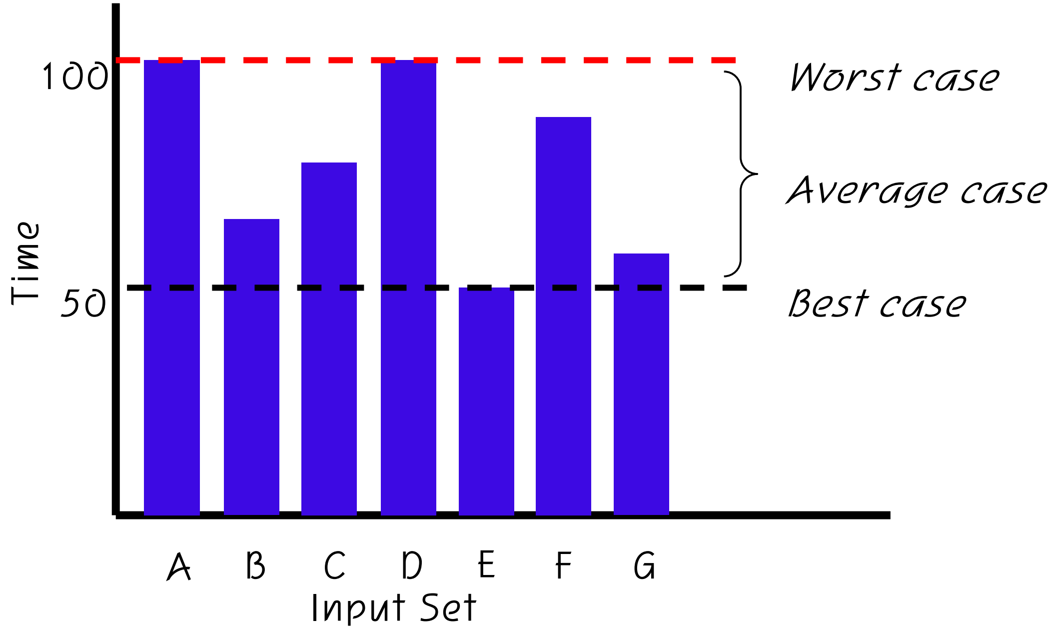
02. Algorithm Basic

Analysis

- Time complexity analysis
  - Determination of how many times the basic operation is done for each value of the input size
  - Should be independent of CPU, OS, Programming languages…
- Metrics
  - Basic operation
    - Comparisons, assignments, etc.
  - Input size
    - The number of elements in an array
    - The length of a list
    - The number of columns and rows in a matrix
    - The number of vertices and edges in a graph

Analysis

- We can consider the following three cases:
  - Best case
  - Average case
  - Worst case



02. Algorithm Basic

Analysis

- Among four possible analysis, WC, AC, and BC, which one is the right one?
- Think about …
  - you are working for a nuclear power plant
  - what if you are working for an internet shopping mall
- Which one do you think is the most useless?
- Which one do you think is the hardest to analyze?

Correctness

- Efficiency analysis vs. Correctness analysis
- In this course, when I say an algorithm analysis, I mean an efficiency analysis
- We can also analyze the correctness of an algorithm by developing a mathematical proof that the algorithm actually does what it is supposed to do
- An algorithm is incorrect…
  - if it does not stop for a given input or
  - if it gives a wrong answer for an input

Level Test

Programming Environment Setting

Thanks

Week 1: Course Introduction
Instructor: Jinyoung Han (jinyounghan@skku.edu)