Algorithm Design
Introduction

Prof. Dr. Brahim Hnich

February 22, 2013
Course Info

What is an Algorithm?

Why study algorithms?
Course Info

- Instructor: Prof. Dr. Brahim Hnich
Course Info

- Instructor:
  - Prof. Dr. Brahim Hnich
Course Info

▶ Instructor:
  ▶ Prof. Dr. Brahim Hnich
▶ Course Webpage
Course Info

- Instructor:
  - Prof. Dr. Brahim Hnich
- Course Webpage
  - http://homes.ieu.edu.tr/~bhnich
Course Info

▷ Course objective
Course Info

- Course objective
  - introduce algorithms by looking at the real-world problems motivating them
Course Info

- Course objective
  - introduce algorithms by looking at the real-world problems motivating them
- Core syllabus
Course Info

- Course objective
  - introduce algorithms by looking at the real-world problems motivating them
- Core syllabus
  - Greedy algorithms
Course Info

- Course objective
  - introduce algorithms by looking at the real-world problems motivating them

- Core syllabus
  - Greedy algorithms
  - Divide and conquer algorithms
Course Info

- Course objective
  - introduce algorithms by looking at the real-world problems motivating them

- Core syllabus
  - Greedy algorithms
  - Divide and conquer algorithms
  - Dynamic programming
Course Info

- Course objective
  - introduce algorithms by looking at the real-world problems motivating them

- Core syllabus
  - Greedy algorithms
  - Divide and conquer algorithms
  - Dynamic programming
  - Approximation algorithms

We will not cover all the chapters of the book
Textbook

- We will not cover all the chapters of the book
- We will not cover all sections of all covered chapters
Textbook

- We will not cover all the chapters of the book
- We will not cover all sections of all covered chapters
- We will not follow the order of the book
Textbook

- We will not cover all the chapters of the book
- We will not cover all sections of all covered chapters
- We will not follow the order of the book
- In few cases the notation and terminology will be different from that of the book
Textbook

- We will not cover all the chapters of the book
- We will not cover all sections of all covered chapters
- We will not follow the order of the book
- In few cases the notation and terminology will be different from that of the book
- The contents uncovered in slides/lectures are not required
Grading

- 2 Midterm exam worth 60%
Grading

- 2 Midterm exam worth 60%
  - Midterms will be on Week 7 and Week 11, respectively.
Grading

- 2 Midterm exam worth 60%
  - Midterms will be on Week 7 and Week 11, respectively.
- 1 Final exam worth 40%
Lecture Slides

- Will be available on class web site.
- Usually posted just before class.
- Class attendance is extremely important.
Lecture Slides

- Will be available on class web site.
- Usually posted just before class.
- **Class attendance is extremely important.** Lecture in class contains significant and substantial additions to material on the slides.
Definition
Definition

Chamber’s  A set of prescribed computational procedures for solving a problem; a step-by-step method for solving a problem.

Knuth, TAOCP  An algorithm is a finite, definite, effective procedure, with some input and some output.
Origin

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja’far Mohammed ben Musa.
Origin

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja’far Mohammed ben Musa.

2. From the Greek *algos* (meaning “pain,” also a root of “analgesic”) and *rythmos* (meaning “flow,” also a root of “rhythm”).
1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja’far Mohammed ben Musa.

2. From the Greek *algos* (meaning “pain,” also a root of “analgesic”) and *rythmos* (meaning “flow,” also a root of “rhythm”).
Origin

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja’far Mohammed ben Musa.

2. From the Greek *algos* (meaning “pain,” also a root of “analgesic”) and *rythmos* (meaning “flow,” also a root of “rhythm”).
1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja’far Mohammed ben Musa. He wrote “Kitab al-jabr wa’l-muqabala,” which evolved into today’s high school algebra text.

2. From the Greek *algos* (meaning “pain,” also a root of “analgesic”) and *rythmos* (meaning “flow,” also a root of “rhythm”).
Find Minimum

**INSTANCE:** Nonempty list $x_1, x_2, \ldots, x_n$ of integers.

**SOLUTION:** Pair $(i, x_i)$ such that $x_i = \min \{x_j \mid 1 \leq j \leq n\}$. 
Algorithm Example

Input: \(x_1, x_2, \ldots, x_n\)
Output: \((i, x_i)\)

\[i \leftarrow 1\]
For each \(j \in \{2, \ldots, n\}\)
  If \(x_j < x_i\)
    \[i \leftarrow j\]
return \((i, x_i)\)
Almost all big companies want coders with knowledge of algorithms. (Google, IBM, Microsoft, Oracle, Yahoo, etc)
Why algorithms?

- Almost all big companies want coders with knowledge of algorithms. (Google, IBM, Microsoft, Oracle, Yahoo, etc)
- In most job interviews, they will ask you questions about algorithms
Why algorithms?

- Almost all big companies want coders with knowledge of algorithms. (Google, IBM, Microsoft, Oracle, Yahoo, etc)
- In most job interviews, they will ask you questions about algorithms
- Your knowledge about algorithms will set you apart from the large masses of interviewee who know only how to program
Why algorithms?

- Almost all big companies want coders with knowledge of algorithms. (Google, IBM, Microsoft, Oracle, Yahoo, etc)
- In most job interviews, they will ask you questions about algorithms
- Your knowledge about algorithms will set you apart from the large masses of interviewee who know only how to program
- If you want to start your own company, you should know that many startups are successful because they’ve found better algorithms for solving a problem (e.g., Google)
Why algorithms?

- You will write better and faster code
Why algorithms?

- You will write better and faster code
- You will learn to think more abstractly and mathematically
Why algorithms?

- You will write better and faster code
- You will learn to think more abstractly and mathematically
- It's the most challenging and interesting area of CS
A real job interview question

- You are given an array of integers between 1 and 1,000,000
A real job interview question

- You are given an array of integers between 1 and 1,000,000
- All integers between 1 and 1,000,000 are in the array at least once. One of them appears twice.
A real job interview question

- You are given an array of integers between 1 and 1,000,000.
- All integers between 1 and 1,000,000 are in the array at least once. One of them appears twice.
- Can you determine which of the integers is in the array twice?
A real job interview question

- You are given an array of integers between 1 and 1,000,000
- All integers between 1 and 1,000,000 are in the array at least once. One of them appears twice.
- Can you determine which of the integers is in the array twice?
- Can you do it while iterating over the array only once!
Naive algorithm

1. Create a new array of integers indexed by 1 through 1,000,000. We will use this array to count the occurrence of each integer. Initialize all entries to 0.

2. Iterate over the input array and each time a number is seen, increment its count in the new array.

3. Go through the new array and see which number occurs twice.

4. Return this number.
Naive algorithm

- Create a new array of integers indexed by 1 through 1,000,000. We will use this array to count the occurrence of each integer. Initialize all entries to 0.
- Iterate over the input array and each time a number is seen, increment its count in the new array.
Naive algorithm

- Create a new array of integers indexed by 1 through 1,000,000. We will use this array to count the occurrence of each integer. Initialize all entries to 0.
- Iterate over the input array and each time a number is seen, increment its count in the new array.
- Go through the new array and see which number occurs twice.
Naive algorithm

- Create a new array of integers indexed by 1 through 1,000,000. We will use this array to count the occurrence of each integer. Initialize all entries to 0.
- Iterate over the input array and each time a number is seen, increment its count in the new array.
- Go through the new array and see which number occurs twice.
- Return this number.
Naive algorithm analysis

- How long will this algorithm take?
Naive algorithm analysis

- How long will this algorithm take?
- We iterate through the number 1 to 1,000,000 three times!
Naive algorithm analysis

- How long will this algorithm take?
- We iterate through the number 1 to 1,000,000 three times!
- In addition, we use up a lot of space with this extra array
Naive algorithm analysis

- How long will this algorithm take?
- We iterate through the number 1 to 1,000,000 three times!
- In addition, we use up a lot of space with this extra array
- This is a waste of time and space
Naive algorithm analysis

- How long will this algorithm take?
- We iterate through the number 1 to 1,000,000 three times!
- In addition, we use up a lot of space with this extra array
- This is a waste of time and space
- Can we do better?
Ideas for a better algorithm

- Note that $\sum_{i=1}^{n} i = (n + 1)n/2$
- Let $S$ be the sum of the input array
- Let $x$ be the value of the repeated number
- Then $S = (1,000,000 + 1)1,000,000/2 + x$
- Thus $x = S - (1,000,000 + 1)1,000,000/2$
A better algorithm

- Iterate through the input array, summing up all the numbers, let $S$ be this sum
- Let $x = S - (1,000,000 + 1)1,000,000/2$
- Return $x$
Analysis

- This algorithm takes iterates through the input array just once
- It uses up essentially no extra space
- It is at least three times faster than the naive algorithm
- Further, if the input array is so large that it won’t fit in memory, this is the only algorithm which will work!
- These time and space bounds are the best possible
Take away message

- Designing good algorithms matters!
- Not always this easy to improve an algorithm
- However, with some thought and work, you can *almost always* get a better algorithm than the naive approach