CS 7220 – Computational Complexity and Algorithm Analysis

Spring 2016

Slides 1

Pascal Hitzler

Data Semantics Laboratory

Wright State University, Dayton, OH

http://www.pascal-Hitzler.de

Today’s Session

1. A motivating example
2. What is *Computational Complexity* all about?
3. More examples
4. A computational complexity success story
5. Organizational matters

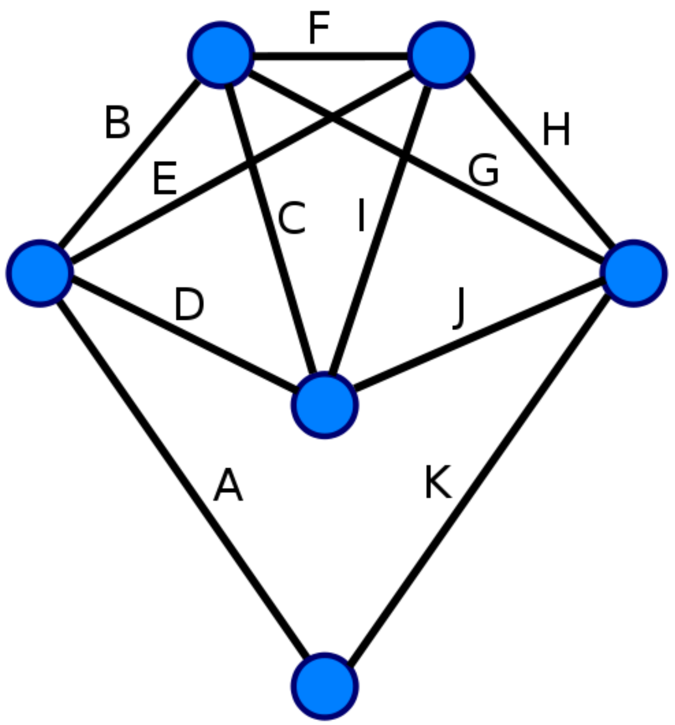
A motivating example

Input: A connected graph (undirected)

Output: “yes” if the graph has a path which

- **visits each edge exactly once and**
- **starts and ends on the same vertex.**

Output: “no” otherwise



Find an algorithm for this problem.

[Such graphs are called *Eulerian*.]

[Such paths are called *Eulerian cycles*.]

Naive algorithm 1 – brute force

- Graph consists of
  - m vertices: 1,2,...,m
  - n edges, written as (x,y) [edge between vertex x and vertex y]

```
for each permutation P of the n edges
      [i.e., P = ( $(x_1,y_1),\ldots,(x_n,y_n)$ )]
  output “yes” if P constitutes an Eulerian cycle
output “no” if no Eulerian cycle was found
```

Is this a good algorithm?

How to improve?

Naive algorithm 1 – brute force

- **Graph consists of**
  - **m vertices: 1,2,...,m**
  - **n edges, written as (x,y) [edge between vertex x and vertex y]**

```
for each permutation P of the n edges
      [i.e., P = ( ($x_1$,$y_1$),...,($x_n$,$y_n$) )]
  output "yes" if P constitutes an Eulerian cycle
output "no" if no Eulerian cycle was found
```

How costly is this? (roughly, order of magnitude)

If no Eulerian cycle is found, we have to check n! permutations (that's the *worst case*).

n! is quite a lot!

n	n!
5	120
10	3,628,800
15	¼ 1.3 ¢10^{12}
20	¼ 2.4 ¢10^{18}
50	¼ 3 ¢10^{64}
70	¼ 10^{100}

10^{100} – That's more than there are particles in the universe

n! is quite a lot!

10^1	
10^2	
10^3	Number of students in the college of engineering
10^4	Number of students enrolled at WSU
10^6	Number of people in Dayton Metro
10^7	Number of people in Ohio Number of seconds in a year
10^8	Number of people in Germany
10^10	Number of stars in the galaxy Number of people on earth Number of milliseconds per year
10^20	Number of stars in the universe
10^80	Number of particles in the universe

Naive algorithm 2 - backtracking

- Graph consists of
  - m vertices: 1,2,...,m
  - n edges, written as (x,y) [edge between vertex x and vertex y]

```
Fix the first vertex, say, x₁.
Make a systematic depth-first search on the graph
            edges.
  For each resulting maximal path P, if P is an
            Eulerian cycle, output “yes”.
If no Eulerian cycle is found, output “no”.
```

Algorithm is better – but is it *significantly* better?

In the worst case, fully connected graph, we have to check m! paths.

When do we know we have *the best* algorithm?

Smart algorithm

Theorem:

A connected undirected graph is Eulerian if and only if every vertex has an even number of edges (counting loops twice).

```
For each vertex x
  if number of edges of x is odd, output “No” and
                        stop.
Output “Yes”.
```

In the worst case, we have to make m checks, each of which consists of counting at most n edges.

Today’s Session

1. A motivating example
2. What is *Computational Complexity* all about?
3. More examples
4. A computational complexity success story
5. Organizational matters

What is Comp. Complexity all about?

- Some problems seem hard but are not. Identify them.
- Some problems seem easy but are not. Identify them.

- Know when to stop searching for a smarter algorithm.
  [And instead turn to optimizations and heuristics.]

- What does "computationally hard problem" mean exactly?
- In what sense can we really say that some problem is computationally harder than some other problem?

What is Comp. Complexity all about?

- It’s a part of *theoretical* computer science.
- It’s a formal theory of the analysis of computational hardness of problems.
- It’s probably rarely going to help you directly in practice.
- But indirectly, in form of having a systematic understanding of problem hardness, it is indispensable.

What is Comp. Complexity all about?

We will certainly also learn about the

P = NP?

problem.

What it is.
Why it is important.

Why some people make such a fuzz about it.

Some basic notions

- **Problem:**
  A mapping from input to output.

- **Algorithm:**
  A method or a process followed to solve a problem.

- **A problem can have many algorithms.**

Some basic notions

- **Problem:**
  A mapping from input to output.

- **Algorithm:**
  A method or a process followed to solve a problem.

- **We focus on problems. Algorithm analysis is also interesting, but not as foundationally important.**

Some basic notions

- **Problem:**
  A mapping from input to output.
  - **We use the *order of magnitude* of the number of steps needed to solve a problem.**
    - **measured as a number which depends on the *input size*.**
  - **We are really interested in the *worst case* scenario.**
    - **i.e., how many steps do we need if the input is as unfavorable as possible?**

Can also be studied:

best case (usually not that interesting)

average case (of practical interest for concrete algorithms)

Today’s Session

1. A motivating example
2. What is *Computational Complexity* all about?
3. More examples
4. A computational complexity success story
5. Organizational matters

Linear Search

- **Input:** **An array *A* of integers, and a value *v*.**
- **Output:** **"Yes" if *v* is an element of *A*;**
  "No" otherwise.

A =

3	12	7	25	7	32	11	56	28	43	6	87	68	91	2

v = 28

Algorithms: *Exhaustive search; random search; sort and linear search; sort and binary search*

Linear Search

Not all inputs of a given size take the same time to run.

Sequential search for *v* in an array of *n* integers:

- **Begin at first element in array and look at each element in turn until *v* is found**

Best case:

Worst case:

Average case?

Linear Search – Average case

Case: i	Time: T(i)	Probability P(i)	Cost: T(i) * P(i)
1	1	1/n	1/n
2	2	1/n	2/n
3	3	1/n	3/n
…	…	…	…
n	n	1/n	1

$$\sum Cost = {}^{1}\!/_{n} + {}^{2}\!/_{n} + \cdots + {}^{n}\!/_{n}$$

$$= \frac{1}{n} \times \sum_{i=1}^{n} i$$

$$= \frac{1}{n} \times \frac{n(n+1)}{2} = \frac{n+1}{2}$$

Bubble Sort

- **Algorithm 1:**

```
for pass = 0…n-1 {
  for position = 0…n-1 {
    if (array[position] > array[position+1]) {
      swap (array[position], array[position+1])
    }
  }
}
```

Best, worst, average: $\approx (n^2)$

Bubble Sort

- **Algorithm 2:**

```
for pass = 0…n-1 {
  for position = 0…n-pass-1 {
    if (array[position] > array[position+1]) {
      swap (array[position], array[position+1])
    }
  }
}
```

Best, worst, average: $\approx (n^2)$

(Within a constant factor)

Bubble Sort

- **Algorithm 3:**

```
for pass = 0…n-1 {
  swaps = 0;
  for position = 0…n-pass-1 {
    if (array[position] > array[position+1]) {
      swap (array[position], array[position+1]);
      swaps++;
    }
  if (swaps == 0) return;
  }
}
```

Best: $\approx n$,

Worst: $\approx (n^2)$

Average = ??

Today's Session

1. A motivating example
2. What is *Computational Complexity* all about?
3. More examples
4. A computational complexity success story
5. Organizational matters

A success story

- Research Area: Semantic Web
  Aimed at endowing information on the World Wide Web with "machine-processable meaning" (semantics).

- This is done using languages for representing knowledge.
  E.g., the knowledge on a website.
- These languages can also be used for querying this knowledge.

- These languages are also able to represent problems.
  Knowledge: A graph.
  Query: Does it have an Eulerian cycle?

- These languages differ in how "complex" the problems representable in them can be.

A success story

- Web Ontology Language OWL
  Recommended standard by the World Wide Web Consortium W3C.
  Established 2004, revised 2009.

- Research which led to OWL was driven by computational complexity analysis.

- Complexity used as a priori measure for runtime.

- Goal was finding a language which allows maximum freedom in specifying knowledge (problems), while being of minimal complexity.

- This approach paid off extremely well:
  Currently e.g. substantial commercial interest generated.

Today's Session

1. A motivating example
2. What is *Computational Complexity* all about?
3. More examples
4. A computational complexity success story
5. Organizational matters

Organizational Matters

- Textbook (required):
  Thomas A. Sudkamp, Languages and Machines, Third Edition, Addison Wesley, 2006.

- I assume that you have a working knowledge of Turing Machines as conveyed in this book, and that you know how to write down and read formal languages.
  If you do not, study the corresponding chapters in the textbook yourself, in particular chapters 2 and 8.

- I also assume that you have a working knowledge of first-order predicate logic, as conveyed e.g. in CS2210.

- Textbook (supplementary):
  Michael R. Garey and David S. Johnson, Computers and Intractability, Freeman, 1979

Organizational matters

- We use a public website for distribution of material: http://www.pascal-hitzler.de/teaching/s16/

- On that page, you will find a draft manuscript (pdf) which I will use for most of the class. Content of the later weeks will be added.

- The manuscript contains exercises. They are hard. But they are similar to what will be expected from you in the exams.
  - Try to solve all exercises by yourself.
  - Afterwards, consult the study table to get feedback.
  - Subsequently, make a better solution.
  - If you’re still uncertain, consult the study table again.

Compulsory graded homework

- There will be some homework assignments, which will be graded.
- I will announce these in class.
- You need to submit a “best effort” solution to all homework assignments, otherwise you will not be admitted to the final exam.

Organizational Matters

- **Study table: Mondays 4-6pm, run by David Carral, Russ 417.**
  If you need different appointments, contact him by email at **dcarralma@gmail.com**
  Extra study table on Tuesday 19th of January 4-6pm, Russ 417.
  If you come late to the study table, you may have to pick up David in Joshi 465.
  However, we will not make extra sessions in the exam weeks if attendance throughout the semester is generally low.
- **Office Hours: Tue 3:00-4:00pm, Joshi 483.**
  Email contact preferred.
  The office hour is not for content questions. For these, go to the study table or ask in class.
- **Grading:**
  First midterm exam: 30% - Tuesday February 16.
  Second midterm exam: 30% - date will be announced.
  Final exam: 40% - comprehensive.

Ethical Conduct

- You may (and are in fact encouraged to) work on exercises in teams. It’s up to you to find the right people for you. For graded homework, make sure you write it up by yourself after the discussion, we will not accept duplicates or close duplicates.
- For the exams, we will make the seating arrangements, and you will be required to bring a government-issued photo ID to prove your identity.
- If, during the exam, we see anybody handling a cellphone, this person will have to leave the exam immediately and will fail the class.
- If, during the exam, we see anybody attempting to give or receive help, this person will fail the class, and an official report will be filed.

We’re here to help you

- If you need additional help with the class, let us know.

- We will gladly help, provided that
  - you request help in a timely fashion,
  - you work hard and with dedication on your success, and
  - you have already exhausted all the standard help we provide (i.e., you have caught up on all preliminaries, you are in all classes, you read the book in detail, you attend all study tables, and you come prepared with serious attempts on all exercises).

That said …

- Attendance of class and study table is voluntary.
If you can learn all material from book and manuscript, do that.
But we will not help you to make up severe gaps then.
It is also your responsibility to stay current regarding all announcements made in class.

- Attendance of the exams is strictly required. Exceptions will only be made in cases of documented emergencies.