جامعة الملك سعود
King Saud University

Theory of Computation
CSC 339 – Spring 2021

Chapter-7: part1
Time Complexity

King Saud University

Department of Computer Science

Dr. Azzam Alsudais

Introduction

- **Now that we’ve covered computability theory topics, it’s time to look at complexity theory.**

Introduction

- **Now that we’ve covered computability theory topics, it’s time to look at complexity theory.**

- **What’s the difference between computability and complexity theories?**

Complexity Theory

- **Solvable (decidable) problems may be difficult to solve without incurring the use of significant amount of resources (time, space, etc).**

Complexity Theory

- **Solvable (decidable) problems may be difficult to solve without incurring the use of significant amount of resources (time, space, etc).**

- **In computability theory, we were concerned with the decidability of problems (or languages for that matter).**

Complexity Theory

- **Solvable (decidable) problems may be difficult to solve without incurring the use of significant amount of resources (time, space, etc).**

- **In computability theory, we were concerned with the decidability of problems (or languages for that matter).**

- **In complexity theory, we would like to take a closer look at solvable problems and see how difficult the solution would be.**

Complexity Theory

- **Solvable (decidable) problems may be difficult to solve without incurring the use of significant amount of resources (time, space, etc).**

- **In computability theory, we were concerned with the decidability of problems (or languages for that matter).**

- **In complexity theory, we would like to take a closer look at solvable problems and see how difficult the solution would be.**

- **In this part, we will study time complexity.**

Complexity Theory: Time Complexity

- **How much time does a given TM take to process some input?**

Complexity Theory: Time Complexity

- **How much time does a given TM take to process some input?**
- **Does the TM always take the same amount of time (number of steps) to process strings of the same length?**

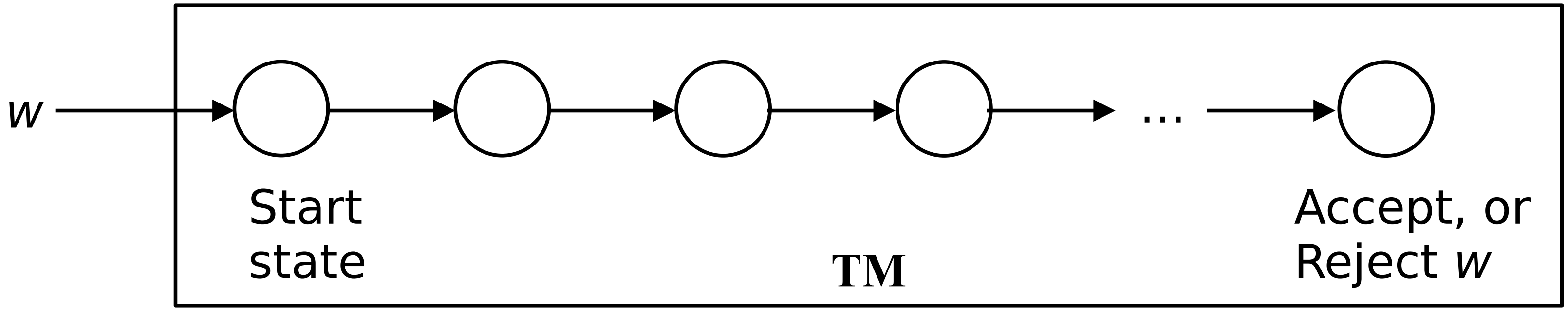
Complexity Theory: Time Complexity

- **How much time does a given TM take to process some input?**
- **Does the TM always take the same amount of time (number of steps) to process strings of the same length?**
  - **aaabbb → may take *n* number of steps**
  - **ababab → may take *n+3* number of steps, and so on.**

Complexity Theory: Time Complexity

- **How much time does a given TM take to process some input?**

- **Does the TM always take the same amount of time (number of steps) to process strings of the same length?**
  - **aaabbb → may take *n* number of steps**
  - **ababab → may take *n+3* number of steps, and so on.**



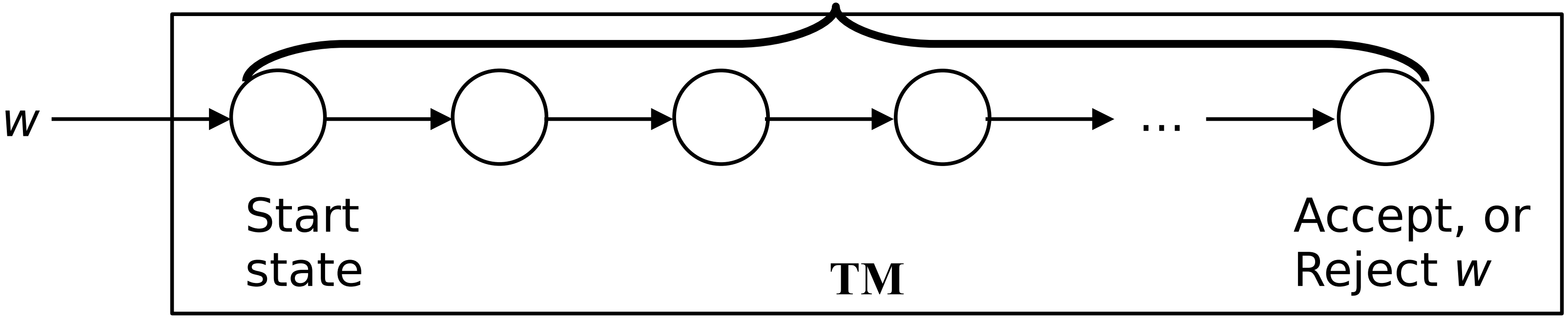
Complexity Theory: Time Complexity

- **How much time does a given TM take to process some input?**

- **Does the TM always take the same amount of time (number of steps) to process strings of the same length?**

  - **aaabbb → may take *n* number of steps**
  - **ababab → may ... on.**

Running Time = number of transitions



Complexity Theory: Time Complexity

- **How much time does a given TM take to process some input?**
- **Does the TM always take the same amount of time (number of steps) to process strings of the same length?**
  - **aaabbb → may take *n* number of steps**
  - **ababab → may take *n+3* number of steps, and so on.**
- **We compute the running time of an algorithm (or a TM) as a function of the length of the string representing the input.**

Complexity Theory: Time Complexity

- **Worst-case analysis**
  - **We consider the longest running time of all inputs of a particular length.**

Complexity Theory: Time Complexity

- **Worst-case analysis**
  - **We consider the longest running time of all inputs of a particular length.**
- **Average-case analysis**
  - **We consider the average running time of all inputs of a particular length.**

Complexity Theory: Time Complexity

Definition 7.1

Let M be a deterministic Turing machine that halts on all inputs. The ***running time*** or ***time complexity*** of M is the function f: $N \rightarrow N$, where $f(n)$ is the maximum number of steps that M uses on any input of length n. If $f(n)$ is the running time of M , we say that M runs in time $f(n)$ and that M is an $f(n)$ time Turing machine. Customarily, we use n to represent the length of the input.

Complexity Theory: Asymptotic Analysis

- **To further simplify the measurement of time complexity, we use asymptotic analysis.**

Complexity Theory: Asymptotic Analysis

- **To further simplify the measurement of time complexity, we use asymptotic analysis.**

- **The goal is to simplify how we refer to** $f(n)$**.**

Complexity Theory: Asymptotic Analysis

- **To further simplify the measurement of time complexity, we use asymptotic analysis.**
- **The goal is to simplify how we refer to** $f(n)$**.**
- **The main idea is to consider the highest order term.. Disregarding any coefficients and lower order terms.**

Complexity Theory: Asymptotic Analysis

- **To further simplify the measurement of time complexity, we use asymptotic analysis.**

- **The goal is to simplify how we refer to** *f(n)***.**

- **The main idea is to consider the highest order term.. Disregarding any coefficients and lower order terms.**

- **e.g.,** $f(n) = 6n^3 + 2n^2 + 20n + 45$

  - $f(n)$ **is asymptotically at most** n^3**.**

Complexity Theory: Asymptotic Analysis

- **To further simplify the measurement of time complexity, we use asymptotic analysis.**

- **The goal is to simplify how we refer to** *f(n)*.

- **The main idea is to consider the highest order term.. Disregarding any coefficients and lower order terms.**

- **e.g.,** $f(n) = 6n^3 + 2n^2 + 20n + 45$

  - $f(n)$ **is asymptotically at most** n^3.

- ***Asymptotic notation* is also known as *big-O notation*.**

Definition 7.2

Let f and g be functions $f, g : N \rightarrow R+$. Say that $f(n) = O(g(n))$ if positive integers c and n_0 exist such that for every integer $n \geq n_0$,

$$f(n) \leq cg(n).$$

When $f(n) = O(g(n))$, we say that $g(n)$ is an <u>upper bound</u> for $f(n)$, or more precisely, that $g(n)$ is an asymptotic upper bound for $f(n)$, to emphasize that we are suppressing constant factors.

Definition 7.2

Let f and g be functions $f, g : N \rightarrow R+$. Say that $f(n) = O(g(n))$ if positive integers c and n_0 exist such that for every integer $n \geq n_0$,

$$f(n) \leq cg(n).$$

When $f(n) = O(g(n))$, we say that $g(n)$ is an upper bound for $f(n)$, or more precisely, that $g(n)$ is an asymptotic upper bound for $f(n)$, to emphasize that we are suppressing constant factors.

Intuitively, $f(n) = O(g(n))$ means that f is less than or equal to g if we disregard differences up to a constant factor.

Complexity Theory: Asymptotic Analysis

- **To further simplify the measurement of time complexity, we use asymptotic analysis.**

- **The goal is to simplify how we refer to** $f(n)$**.**

- **The main idea is to consider the highest order term.. Disregarding any coefficients and lower order terms.**

- **e.g.,** $f(n) = 6n^3 + 2n^2 + 20n + 45$
  - $f(n)$ **is asymptotically at most** n^3**.**

> *i* Is $f(n) = O(n^4)$?

- ***Asymptotic notation* is also known as *big-O notation*.**

Complexity Theory: Asymptotic Analysis

- **Examples**
  - $\sqrt{n} = O(n)$.
  - $n = O(n \log \log n)$.
  - $n \log \log n = O(n \log n)$.
  - $n \log n = O(n^2)$.
  - $n^2 = O(n^3)$.