Computers, Complexity and Intractability

Purushothama B R

Department of Computer Science and Engineering
National Institute of Technology Goa ,INDIA

March 29, 2017

I want to Begin With... Michael Jordan's Quote

I've always believed that if you put in the work, the results will come. I don't do things half-heartedly. Because I know if I do, then I can expect half-hearted results.

Whimsical Example for Introduction

Whimsical Example for Introduction

- Suppose that you are employed in the halls of industry.

Whimsical Example for Introduction

- Suppose that you are employed in the halls of industry.
- One day your boss calls you into his office and confides that the company is about to enter the highly competitive "bandersnatch" market.

Whimsical Example for Introduction

- Suppose that you are employed in the halls of industry.
- One day your boss calls you into his office and confides that the company is about to enter the highly competitive "bandersnatch" market.
- For this reason,

Whimsical Example for Introduction

- Suppose that you are employed in the halls of industry.
- One day your boss calls you into his office and confides that the company is about to enter the highly competitive "bandersnatch" market.
- For this reason,
  - A good method is needed for determining whether or not any given set of specifications for a new bandersnatch component can be met,

Whimsical Example for Introduction

- Suppose that you are employed in the halls of industry.
- One day your boss calls you into his office and confides that the company is about to enter the highly competitive "bandersnatch" market.
- For this reason,
  - A good method is needed for determining whether or not any given set of specifications for a new bandersnatch component can be met, and

Whimsical Example for Introduction

- Suppose that you are employed in the halls of industry.
- One day your boss calls you into his office and confides that the company is about to enter the highly competitive "bandersnatch" market.
- For this reason,
  - A good method is needed for determining whether or not any given set of specifications for a new bandersnatch component can be met, and
  - If so,

Whimsical Example for Introduction

- Suppose that you are employed in the halls of industry.
- One day your boss calls you into his office and confides that the company is about to enter the highly competitive "bandersnatch" market.
- For this reason,
  - A good method is needed for determining whether or not any given set of specifications for a new bandersnatch component can be met, and
  - If so, for constructing a design that meets them.

Whimsical Example for Introduction

- Suppose that you are employed in the halls of industry.
- One day your boss calls you into his office and confides that the company is about to enter the highly competitive "bandersnatch" market.
- For this reason,
  - A good method is needed for determining whether or not any given set of specifications for a new bandersnatch component can be met, and
  - If so, for constructing a design that meets them.
- Since you are the company's chief algorithm designer, your charge is to find an efficient algorithm for doing this.

You may proceed to..

You may proceed to..

- Consult the bandersnatch department to determine exactly what the problem is?

You may proceed to..

- Consult the bandersnatch department to determine exactly what the problem is?
- You eagerly hurry back to your office, and plunge into the task.

You may proceed to..

- Consult the bandersnatch department to determine exactly what the problem is?
- You eagerly hurry back to your office, and plunge into the task.
- Some weeks later, your office filled with mountains of crumpled-up scratch paper,

You may proceed to..

- Consult the bandersnatch department to determine exactly what the problem is?
- You eagerly hurry back to your office, and plunge into the task.
- Some weeks later, your office filled with mountains of crumpled-up scratch paper, your enthusiasm has lessened considerably.

You may proceed to..

- Consult the bandersnatch department to determine exactly what the problem is?
- You eagerly hurry back to your office, and plunge into the task.
- Some weeks later, your office filled with mountains of crumpled-up scratch paper, your enthusiasm has lessened considerably.
- So far you have not been able to come up with any algorithm

You may proceed to..

- Consult the bandersnatch department to determine exactly what the problem is?
- You eagerly hurry back to your office, and plunge into the task.
- Some weeks later, your office filled with mountains of crumpled-up scratch paper, your enthusiasm has lessened considerably.
- So far you have not been able to come up with any algorithm substantially better than searching through all possible designs.

You may proceed to..

- Consult the bandersnatch department to determine exactly what the problem is?
- You eagerly hurry back to your office, and plunge into the task.
- Some weeks later, your office filled with mountains of crumpled-up scratch paper, your enthusiasm has lessened considerably.
- So far you have not been able to come up with any algorithm substantially better than searching through all possible designs.
- This would involve years of computation time for just one set of specifications.

You certainly don't want to return to your boss's office and report

"I can't find an efficient algorithm, I guess I'm just too dumb."

Your Reputation at Stake!!!

Your Reputation at Stake!!!

- To avoid serious damage to your position within the company,

Your Reputation at Stake!!!

- To avoid serious damage to your position within the company,
- It would be much better if you could prove that the bandersnatch problem is **inherently intractable**,

Your Reputation at Stake!!!

- To avoid serious damage to your position within the company,
- It would be much better if you could prove that the bandersnatch problem is **inherently intractable**, that no algorithm could possibly solve it Quickly .

You stride Confidently into your boss office and proclaim:

"I can't find an efficient algorithm, because no such algorithm is possible!"

Still..... Unfortunate!!!!

Still..... Unfortunate!!!!

- Unfortunately, proving **inherent intractability** can be just **as hard as finding efficient algorithms.**

Still..... Unfortunate!!!!

- Unfortunately, proving **inherent intractability** can be just **as hard as finding efficient algorithms.**
- Even the best theoreticians have been stymied in their attempts to obtain such proofs for commonly encountered hard problems.

A Ray of Hope: Theory of NP-Completeness

A Ray of Hope: Theory of NP-Completeness

- The theory of NP-completeness

- The theory of NP-completeness
  - Provides many straightforward techniques for proving that a given problem is

A Ray of Hope: Theory of NP-Completeness

- The theory of NP-completeness
  - Provides many straightforward techniques for proving that a given problem is
    - **"just as hard"** as a large number of other problems that are widely recognized as being difficult.

A Ray of Hope: Theory of NP-Completeness

- The theory of NP-completeness
  - Provides many straightforward techniques for proving that a given problem is
    - **"just as hard"** as a large number of other problems that are widely recognized as being difficult.
- Armed with these techniques,

A Ray of Hope: Theory of NP-Completeness

- The theory of NP-completeness
  - Provides many straightforward techniques for proving that a given problem is
    - **"just as hard"** as a large number of other problems that are widely recognized as being difficult.
- Armed with these techniques,
  - You might be able to prove that the bandersnatch problem is **NP-complete**,

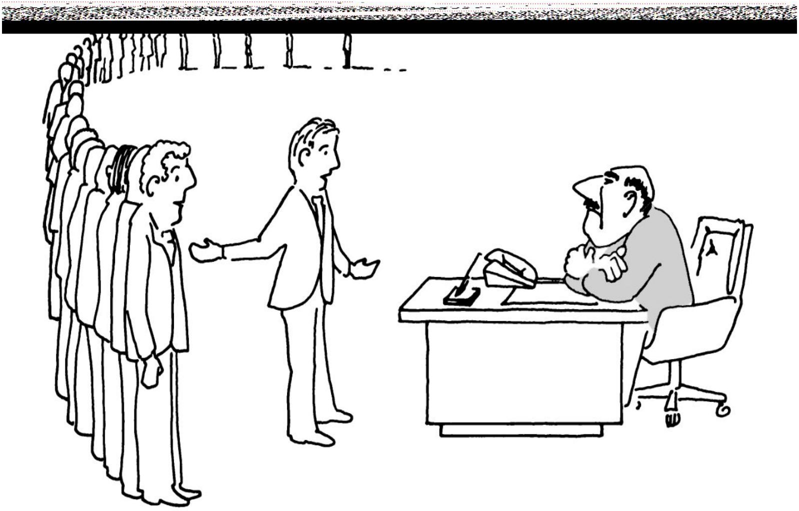
A Ray of Hope: Theory of NP-Completeness

- The theory of NP-completeness
  - Provides many straightforward techniques for proving that a given problem is
    - **"just as hard"** as a large number of other problems that are widely recognized as being difficult.
- Armed with these techniques,
  - You might be able to prove that the bandersnatch problem is **NP-complete**,and

A Ray of Hope: Theory of NP-Completeness

- The theory of NP-completeness
  - Provides many straightforward techniques for proving that a given problem is
    - **"just as hard"** as a large number of other problems that are widely recognized as being difficult.
- Armed with these techniques,
  - You might be able to prove that the bandersnatch problem is **NP-complete**,and
  - It is equivalent to all these other hard problems.

You march into your boss's office and announce:



"I can't find an efficient algorithm, but neither can all these famous people."

Saved... Happy :)

Saved... Happy :)

- At the very least, this would inform your boss that it would do no good to fire you and hire another expert on algorithms.

However....

However....

- Discovering that a problem is NP-complete is usually just the beginning of work on that problem.

However....

- Discovering that a problem is NP-complete is usually just the beginning of work on that problem.
- The needs of the bandersnatch department won't disappear overnight simply because their problem is known to be NP-complete.

However....

- Discovering that a problem is NP-complete is usually just the beginning of work on that problem.
- The needs of the bandersnatch department won't disappear overnight simply because their problem is known to be NP-complete.
- However, the knowledge that it is NP-complete does provide valuable information about what lines of approach have the potential of being most productive.

Special Cases rather the General Case

Special Cases rather the General Case

- Certainly the search for an efficient, exact algorithm should be accorded low priority.

Special Cases rather the General Case

- Certainly the search for an efficient, exact algorithm should be accorded low priority.
- It is now more appropriate to concentrate on other, **less ambitious**, approaches.

Special Cases rather the General Case

- Certainly the search for an efficient, exact algorithm should be accorded low priority.
- It is now more appropriate to concentrate on other, **less ambitious**, approaches.
- Look for efficient algorithms that solve various **special cases** of the general problem.

Special Cases rather the General Case

- Certainly the search for an efficient, exact algorithm should be accorded low priority.
- It is now more appropriate to concentrate on other, **less ambitious**, approaches.
- Look for efficient algorithms that solve various **special cases** of the general problem.
- You might even relax the problem somewhat, looking for a fast algorithm that merely finds designs that meet **most** of the component specifications.

Primary Application of theory of NP-Completeness

To assist the algorithm designers in directing their **problem solving** efforts toward those approaches that have the greatest likelihood of leading to **useful algorithms**.

Problems, Algorithms and Complexity

Basic Terms: Problem and Instance of a Problem

Basic Terms: Problem and Instance of a Problem

- **Problem** is a general question to be answered, usually possessing several **parameters**, or free variables, whose values are left unspecified.

Basic Terms: Problem and Instance of a Problem

- **Problem** is a general question to be answered, usually possessing several **parameters**, or free variables, whose values are left unspecified.
- A problem is described by giving:

Basic Terms: Problem and Instance of a Problem

- **Problem** is a general question to be answered, usually possessing several **parameters**, or free variables, whose values are left unspecified.
- A problem is described by giving:
  - A general description of all its parameters, and

Basic Terms: Problem and Instance of a Problem

- **Problem** is a general question to be answered, usually possessing several **parameters**, or free variables, whose values are left unspecified.
- A problem is described by giving:
  - A general description of all its parameters, and
  - A statement of what properties the answer, or solution, is required to satisfy.

Basic Terms: Problem and Instance of a Problem

- **Problem** is a general question to be answered, usually possessing several **parameters**, or free variables, whose values are left unspecified.
- A problem is described by giving:
  - A general description of all its parameters, and
  - A statement of what properties the answer, or solution, is required to satisfy.
- An **instance of a problem** is obtained by specifying particular values for all the problem parameters.

Classical Traveling Salesman Problem

Classical Traveling Salesman Problem

- The parameters of "Traveling Salesman Problem" consist of a finite set $C = \{c_1, c_2, \ldots, c_m\}$ of "cities" and,

Classical Traveling Salesman Problem

- The parameters of "Traveling Salesman Problem" consist of a finite set $C = \{c_1, c_2, \ldots, c_m\}$ of "cities" and,
- For each pair of cities c_i, c_j in C, the "distance" $d(c_i, c_j)$ between them.

Classical Traveling Salesman Problem

- The parameters of "Traveling Salesman Problem" consist of a finite set $C = \{c_1, c_2, \ldots, c_m\}$ of "cities" and,
- For each pair of cities c_i, c_j in C, the "distance" $d(c_i, c_j)$ between them.
- A solution is an ordering $< c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$ of the given cities that minimizes

$$\left(\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) \right) + d(c_{\pi(m)}, c_{\pi(1)})$$

Classical Traveling Salesman Problem

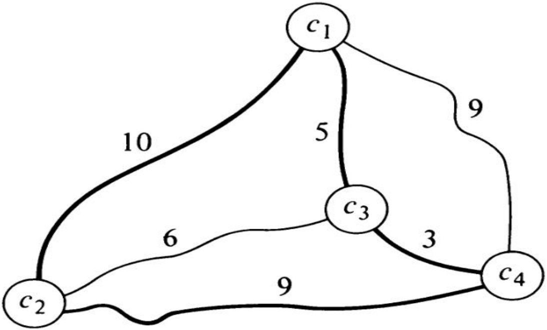
- The parameters of "Traveling Salesman Problem" consist of a finite set $C = \{c_1, c_2, \ldots, c_m\}$ of "cities" and,
- For each pair of cities c_i, c_j in C, the "distance" $d(c_i, c_j)$ between them.
- A solution is an ordering $< c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$ of the given cities that minimizes

$$\left(\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)})\right) + d(c_{\pi(m)}, c_{\pi(1)})$$

- This expression gives the length of the "tour" that starts at $c_{\pi(1)}$, visits each city in sequence, and then returns directly to $c_{\pi(1)}$ from the last city $c_{\pi(m)}$.

Instance of a Traveling Salesman Problem

$C = \{c_1, c_2, c_3, c_4\}$, $d(c_1, c_2) = 10$, $d(c_1, c_3) = 5$, $d(c_1, c_4) = 9$, $d(c_2, c_3) = 6$, $d(c_2, c_4) = 9$, $d(c_3, c_4) = 3$



The ordering $< c_1, c_2, c_4, c_3 >$ is a solution for this instance.

Algorithm

Algorithm

- Algorithms are general, step-by-step procedures for solving problems.

Algorithm

- Algorithms are general, step-by-step procedures for solving problems.
- For concreteness, treat them simply as being computer programs,written in some precise computer language.

Algorithm

- Algorithms are general, step-by-step procedures for solving problems.
- For concreteness, treat them simply as being computer programs,written in some precise computer language.
- An algorithm is said to solve a problem Π

Algorithm

- Algorithms are general, step-by-step procedures for solving problems.
- For concreteness, treat them simply as being computer programs,written in some precise computer language.
- An algorithm is said to solve a problem Π if that algorithm can be applied to any instance I of Π and

Algorithm

- Algorithms are general, step-by-step procedures for solving problems.
- For concreteness, treat them simply as being computer programs,written in some precise computer language.
- An algorithm is said to solve a problem Π if that algorithm can be applied to any instance I of Π and is guaranteed always to produce a solution for that instance I.

Notion of Efficiency

Notion of Efficiency

- In general, interest should be in finding the **most "efficient" algorithm** for solving a problem.

Notion of Efficiency

- In general, interest should be in finding the **most "efficient" algorithm** for solving a problem.
- In its broadest sense, the notion of efficiency involves all the various computing resources needed for executing an algorithm.

Notion of Efficiency

- In general, interest should be in finding the **most "efficient" algorithm** for solving a problem.
- In its broadest sense, the notion of efficiency involves all the various computing resources needed for executing an algorithm.
- However, by the "most efficient" algorithm, normally means the fastest.

Notion of Efficiency

- In general, interest should be in finding the **most "efficient" algorithm** for solving a problem.
- In its broadest sense, the notion of efficiency involves all the various computing resources needed for executing an algorithm.
- However, by the "most efficient" algorithm, normally means the fastest.
- Since time requirements are often a **dominant factor** determining whether or not a particular algorithm is efficient enough to be useful in practice, we concentrate primarily on this single resource.

Size of the Input Instance

Size of the Input Instance

- The time requirements of an algorithm are conveniently expressed in terms of a single variable, the **"size"** of a problem instance.

Size of the Input Instance

- The time requirements of an algorithm are conveniently expressed in terms of a single variable, the **“size”** of a problem instance.
- This reflects the amount of input data needed to describe the instance.

Size of the Input Instance

- The time requirements of an algorithm are conveniently expressed in terms of a single variable, the **"size"** of a problem instance.
- This reflects the amount of input data needed to describe the instance.
- This is convenient because we would expect the relative difficulty of problem instances to vary roughly with the size.

Size of the Problem

Size of the Problem

- Often the size of a problem instance is measured in an informal way.

Size of the Problem

- Often the size of a problem instance is measured in an informal way.
- In Travelling Salesman Problem, the number of cities is commonly used for this purpose.

Size of the Problem

- Often the size of a problem instance is measured in an informal way.
- In Travelling Salesman Problem, the number of cities is commonly used for this purpose.
- However, an $m - city$ problem instance includes, in addition to the labels of the m cities, a collection of $m(m-1)/2$ numbers defining the inter-city distances, and the sizes of these numbers also contribute to the amount of input data.

Size of the Problem

- Often the size of a problem instance is measured in an informal way.
- In Travelling Salesman Problem, the number of cities is commonly used for this purpose.
- However, an $m - city$ problem instance includes, in addition to the labels of the m cities, a collection of $m(m-1)/2$ numbers defining the inter-city distances, and the sizes of these numbers also contribute to the amount of input data.
- To deal with time requirements in a precise, mathematical manner, we must take care to define instance size in such a way that all these factors are taken into account.

Encoding Scheme

Encoding Scheme

- The description of a problem instance that we provide as input to the computer can be viewed as a

Encoding Scheme

- The description of a problem instance that we provide as input to the computer can be viewed as a **single finite string of symbols chosen from a finite input alphabet.**

Encoding Scheme

- The description of a problem instance that we provide as input to the computer can be viewed as a **single finite string of symbols chosen from a finite input alphabet.**
- Although there are many different ways in which instances of a given problem might be described,

Encoding Scheme

- The description of a problem instance that we provide as input to the computer can be viewed as a **single finite string of symbols chosen from a finite input alphabet.**
- Although there are many different ways in which instances of a given problem might be described,
  - **Assume** that one particular way has been chosen in advance and

Encoding Scheme

- The description of a problem instance that we provide as input to the computer can be viewed as a **single finite string of symbols chosen from a finite input alphabet.**
- Although there are many different ways in which instances of a given problem might be described,
  - **Assume** that one particular way has been chosen in advance and
  - that each problem has associated with it a fixed **encoding scheme.**

Encoding Scheme

Encoding Scheme

- An **encoding scheme** maps instances into the strings describing them.

Encoding Scheme

- An **encoding scheme** maps instances into the strings describing them.
- The **input length** for an instance I of a problem Π is defined to be the number of symbols in the description of I obtained from the encoding scheme for Π.
- It is this number, the input length, that is used as the **formal measure of instance size.**

Example

Example

- Instances of the traveling salesman problem might be described using the alphabet $c, [,], /, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.

Example

- Instances of the traveling salesman problem might be described using the alphabet $c, [,], /, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.
- A problem instance being encoded by the string "$c[1]c[2]c[3]c[4]//10/5/9//6/9//3$"

Example

- Instances of the traveling salesman problem might be described using the alphabet $c, [,], /, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.
- A problem instance being encoded by the string "$c[1]c[2]c[3]c[4]//10/5/9//6/9//3$"
- If this were the encoding scheme associated with the traveling salesman problem, then the input length for our example would be 32.

$\boxed{\text{Time Complexity Function}}$

Time Complexity Function

- The time complexity function for an algorithm expresses its time requirements by giving, for each possible input length, the **largest amount of time** needed by the algorithm to solve a problem instance of that size.

Time Complexity Function

- The time complexity function for an algorithm expresses its time requirements by giving, for each possible input length, the **largest amount of time** needed by the algorithm to solve a problem instance of that size.
- This function is not well-defined until one fixes the encoding scheme to be used for determining input length and the computer or computer model to be used for determining execution time.

Time Complexity Function

- The time complexity function for an algorithm expresses its time requirements by giving, for each possible input length, the **largest amount of time** needed by the algorithm to solve a problem instance of that size.
- This function is not well-defined until one fixes the encoding scheme to be used for determining input length and the computer or computer model to be used for determining execution time.
- However, the particular choices made for these will have little effect on the broad distinctions made in the theory of NP-completeness.

Time Complexity Function

- The time complexity function for an algorithm expresses its time requirements by giving, for each possible input length, the **largest amount of time** needed by the algorithm to solve a problem instance of that size.
- This function is not well-defined until one fixes the encoding scheme to be used for determining input length and the computer or computer model to be used for determining execution time.
- However, the particular choices made for these will have little effect on the broad distinctions made in the theory of NP-completeness.
- Hence, fix in mind a particular encoding scheme for each problem and a particular computer or computer model, and to think in terms of time complexity as determined from the corresponding input lengths and execution times.

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**
- In its simplest form a Turing machine consists of a

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**
- In its simplest form a Turing machine consists of a finite state control,

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**
- In its simplest form a Turing machine consists of a finite state control, a two-way infinite memory tape divided into squares, each of which can hold one of a finite number of symbols, and a

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**
- In its simplest form a Turing machine consists of a finite state control, a two-way infinite memory tape divided into squares, each of which can hold one of a finite number of symbols, and a read/write head.

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**
- In its simplest form a Turing machine consists of a finite state control, a two-way infinite memory tape divided into squares, each of which can hold one of a finite number of symbols, and a read/write head.
- In **one step** the machine can

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**
- In its simplest form a Turing machine consists of a finite state control, a two-way infinite memory tape divided into squares, each of which can hold one of a finite number of symbols, and a read/write head.
- In **one step** the machine can read the contents of one tape square,

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**
- In its simplest form a Turing machine consists of a finite state control, a two-way infinite memory tape divided into squares, each of which can hold one of a finite number of symbols, and a read/write head.
- In **one step** the machine can read the contents of one tape square, write a new symbol in the square,

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**
- In its simplest form a Turing machine consists of a finite state control, a two-way infinite memory tape divided into squares, each of which can hold one of a finite number of symbols, and a read/write head.
- In **one step** the machine can read the contents of one tape square, write a new symbol in the square, move the head one square left or right, and

Computer Models

- In order to study the efficiency of algorithms, we need a model of computation.
- Historically, the first machine model proposed was the **Turing machine.**
- In its simplest form a Turing machine consists of a finite state control, a two-way infinite memory tape divided into squares, each of which can hold one of a finite number of symbols, and a read/write head.
- In **one step** the machine can read the contents of one tape square, write a new symbol in the square, move the head one square left or right, and change the state of the control.

An Important Note

An Important Note

- The simplicity of Turing machines makes them very useful in high-level theoretical studies of computational complexity

An Important Note

- The simplicity of Turing machines makes them very useful in high-level theoretical studies of computational complexity
- But, they are not realistic enough to allow accurate analysis of practical algorithms.

An Important Note

- The simplicity of Turing machines makes them very useful in high-level theoretical studies of computational complexity
- But, they are not realistic enough to allow accurate analysis of practical algorithms.
- So, for this purpose a better model is the **Random Access Machine Model**.

Random Access Machine Model

A random access machine consists of:

Random Access Machine Model

A random access machine consists of:

- A finite program,

Random Access Machine Model

A random access machine consists of:

- A finite program, a finite collection of registers, each of which can store a single integer or real number, and

Random Access Machine Model

A random access machine consists of:

- A finite program, a finite collection of registers, each of which can store a single integer or real number, and a memory consisting of an array of n words, each of which has a unique address between 1 and n (inclusive) and can hold a single integer or real number.

Random Access Machine Model

A random access machine consists of:

- A finite program, a finite collection of registers, each of which can store a single integer or real number, and a memory consisting of an array of n words, each of which has a unique address between 1 and n (inclusive) and can hold a single integer or real number.
- In one step,

A random access machine consists of:

- A finite program, a finite collection of registers, each of which can store a single integer or real number, and a memory consisting of an array of n words, each of which has a unique address between 1 and n (inclusive) and can hold a single integer or real number.
- In one step, a random-access machine can perform a single arithmetic or logical operation on the contents of specified registers,

A random access machine consists of:

- A finite program, a finite collection of registers, each of which can store a single integer or real number, and a memory consisting of an array of n words, each of which has a unique address between 1 and n (inclusive) and can hold a single integer or real number.
- In one step, a random-access machine can perform a single arithmetic or logical operation on the contents of specified registers, fetch into a specified register the contents of a word whose address is in a register, or

A random access machine consists of:

- A finite program, a finite collection of registers, each of which can store a single integer or real number, and a memory consisting of an array of n words, each of which has a unique address between 1 and n (inclusive) and can hold a single integer or real number.
- In one step, a random-access machine can perform a single arithmetic or logical operation on the contents of specified registers, fetch into a specified register the contents of a word whose address is in a register, or store the contents of a register in a word whose address is in a register.

Pointer Machine

Pointer Machine

- Similar to Random access model but somewhat less powerful model.

Pointer Machine

- Similar to Random access model but somewhat less powerful model.
- A pointer machine differs from a random-access machine in that its memory consists of an extendable collection of nodes, each divided into a fixed number of named fields. A field can hold a number or a pointer to a node.

Pointer Machine

- Similar to Random access model but somewhat less powerful model.
- A pointer machine differs from a random-access machine in that its memory consists of an extendable collection of nodes, each divided into a fixed number of named fields. A field can hold a number or a pointer to a node.
- However, pointer machines make lower bound studies easier.

Pointer Machine

- Similar to Random access model but somewhat less powerful model.
- A pointer machine differs from a random-access machine in that its memory consists of an extendable collection of nodes, each divided into a fixed number of named fields. A field can hold a number or a pointer to a node.
- However, pointer machines make lower bound studies easier.
- A pointer machine can be simulated by a random-access machine in real time.

Pointer Machine

- Similar to Random access model but somewhat less powerful model.
- A pointer machine differs from a random-access machine in that its memory consists of an extendable collection of nodes, each divided into a fixed number of named fields. A field can hold a number or a pointer to a node.
- However, pointer machines make lower bound studies easier.
- A pointer machine can be simulated by a random-access machine in real time.
- One operation on a pointer machine corresponds to a constant number of operations on a random-access machine.

Properties shared by all three models

Properties shared by all three models

- **Sequential**

Properties shared by all three models

- **Sequential**
  - They carry out one step at a time.

Properties shared by all three models

- **Sequential**
  - They carry out one step at a time.
- **Deterministic**

Properties shared by all three models

- **Sequential**
  - They carry out one step at a time.
- **Deterministic**
  - The future behavior of the machine is uniquely determined by its present configuration. Outside

Complexity Measure

Complexity Measure

- Having picked a machine model, we must select a complexity measure.

Complexity Measure

- Having picked a machine model, we must select a complexity measure.
- One possibility is to measure the complexity of an algorithm by the length of its program.

Complexity Measure

- Having picked a machine model, we must select a complexity measure.
- One possibility is to measure the complexity of an algorithm by the length of its program.
- This measure is static, i.e., independent of the input values.

Complexity Measure

- Having picked a machine model, we must select a complexity measure.
- One possibility is to measure the complexity of an algorithm by the length of its program.
- This measure is static, i.e., independent of the input values.
- Program length is the relevant measure if an algorithm is only to be run once or a few times, and this measure has interesting theoretical uses.

Complexity Measure

- Having picked a machine model, we must select a complexity measure.
- One possibility is to measure the complexity of an algorithm by the length of its program.
- This measure is static, i.e., independent of the input values.
- Program length is the relevant measure if an algorithm is only to be run once or a few times, and this measure has interesting theoretical uses.
- But for our purposes a better complexity measure is a dynamic one, such as running time or storage space as a function of input size.

Complexity Measure

- Having picked a machine model, we must select a complexity measure.
- One possibility is to measure the complexity of an algorithm by the length of its program.
- This measure is static, i.e., independent of the input values.
- Program length is the relevant measure if an algorithm is only to be run once or a few times, and this measure has interesting theoretical uses.
- But for our purposes a better complexity measure is a dynamic one, such as running time or storage space as a function of input size.
- We shall use running time as our complexity measure.

QUESTIONS?

THANK YOU