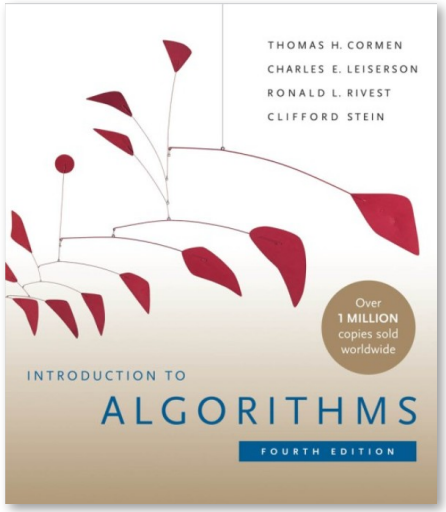


INTRODUCTION TO ALGORITHMS

FOURTH EDITION

Algorithms (Recap)

Algorithm

Definition

- An algorithm is any well-defined computational procedure that
  - takes as input a value or a set of values,
  - outputs some value or a set of values,
    - in a finite amount of time.

Algorithm

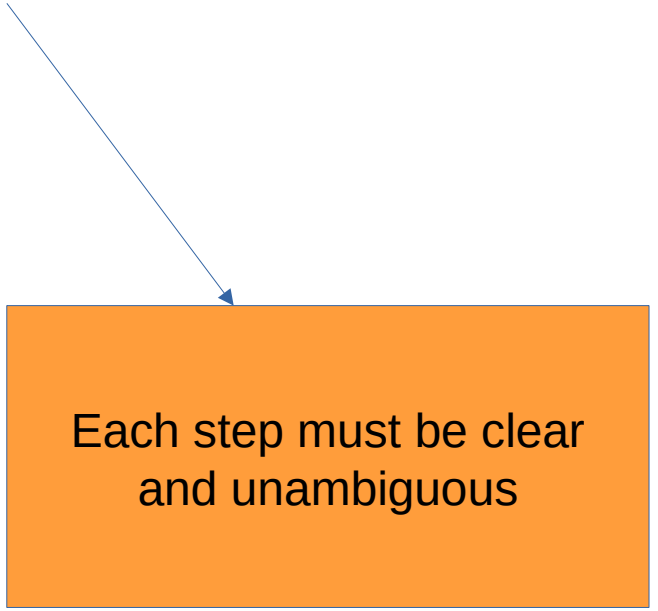
Definition

- An algorithm is any well-defined computational procedure that
  - takes as input a value or a set of values,
  - outputs some value or a set of values,
    - in a finite amount of time.

Algorithm

Definition

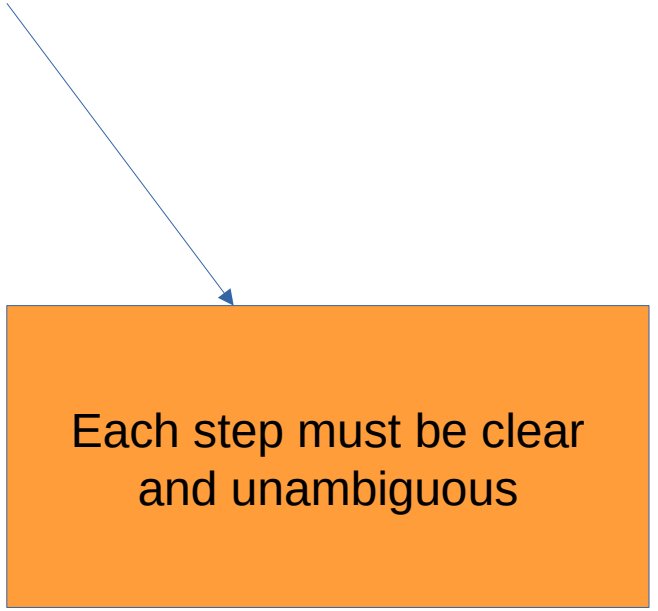
- An algorithm is any well-defined computational procedure that
  - takes as input a value or a set of values,
  - outputs some value or a set of values,
    - in a finite amount of time.



Algorithm

Definition

- An algorithm is any well-defined computational procedure that
  - takes as input a value or a set of values,
  - outputs some value or a set of values,
    - in a finite amount of time.



If the number looks big, show something different.

Vs

If the number is greater than 10, then print ‘High’

Algorithm

Definition

- An algorithm is any well-defined computational procedure that
  - takes as input a value or a set of values,
  - outputs some value or a set of values,
    - in a finite amount of time.

Each step must be clear and unambiguous

It finishes in a finite number of steps (it halts)

* well-defined (unambiguous)
* halt

Problems

- Checking if a number is prime

Do you have an algorithm?

Problems

- Checking if a number is prime → Do you have an algorithm?

- Check if x is divisible by any of the numbers 2 to x-1.
- If yes, output YES, otherwise output NO.

Problems

- Checking if a number is prime

Do you have an algorithm?

- Check if x is divisible by any of the numbers from
- 2 to x-1 in an iterative fashion.
- If yes, output YES, otherwise output NO.

Well-defined?

Halt?

Some Problems

- Checking if a number is prime
- Gcd/hcf of two numbers
- Finding mean/median/mode of a list of number
- Calculating SI/CI
- Finding the maximum of a list of numbers
- Matrix multiplication
- Distance between two points in a plane
- Area/perimeter of some shapes
- …..

Algorithm

Definition

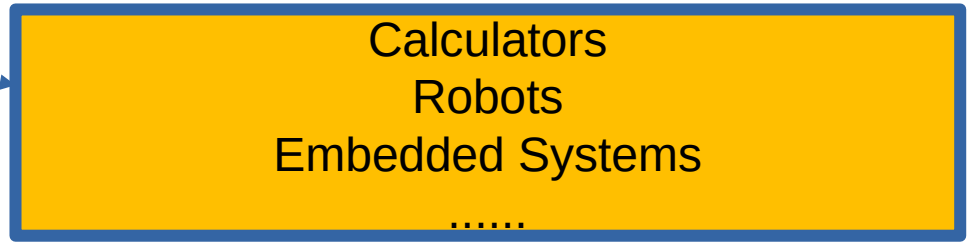
- An algorithm is any well-defined computational procedure that
  - takes as input a value or a set of values,
  - outputs some value or a set of values,
    - in a finite amount of time.

Each step must be clear and unambiguous

It finishes in a finite number of steps (it halts)

Computational Problem

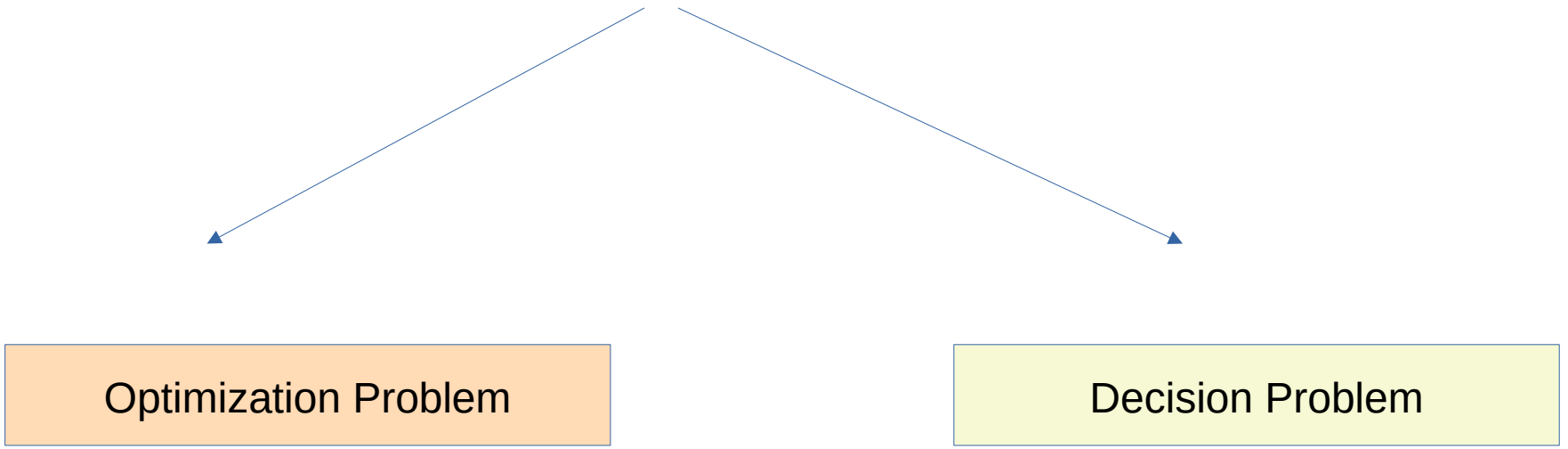
- A problem that can be solved using an **algorithm** by a computer or any computational device.



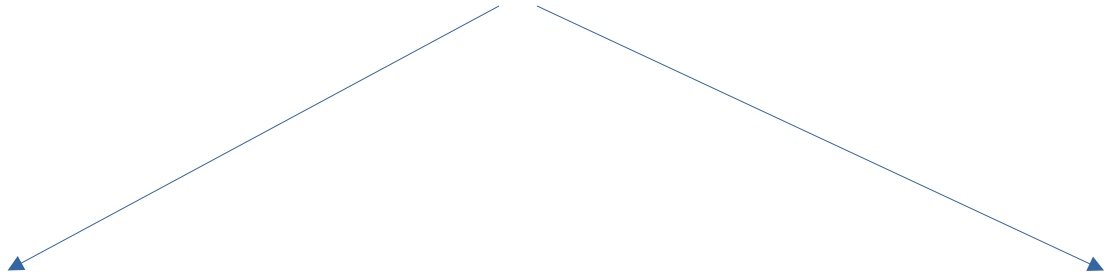
Take-away

- Algorithm is a tool for solving a well-specified computation problem.

Computational Problem



Computational Problem



Optimization Problem

Decision Problem

“Optimize the value”

“YES/NO”

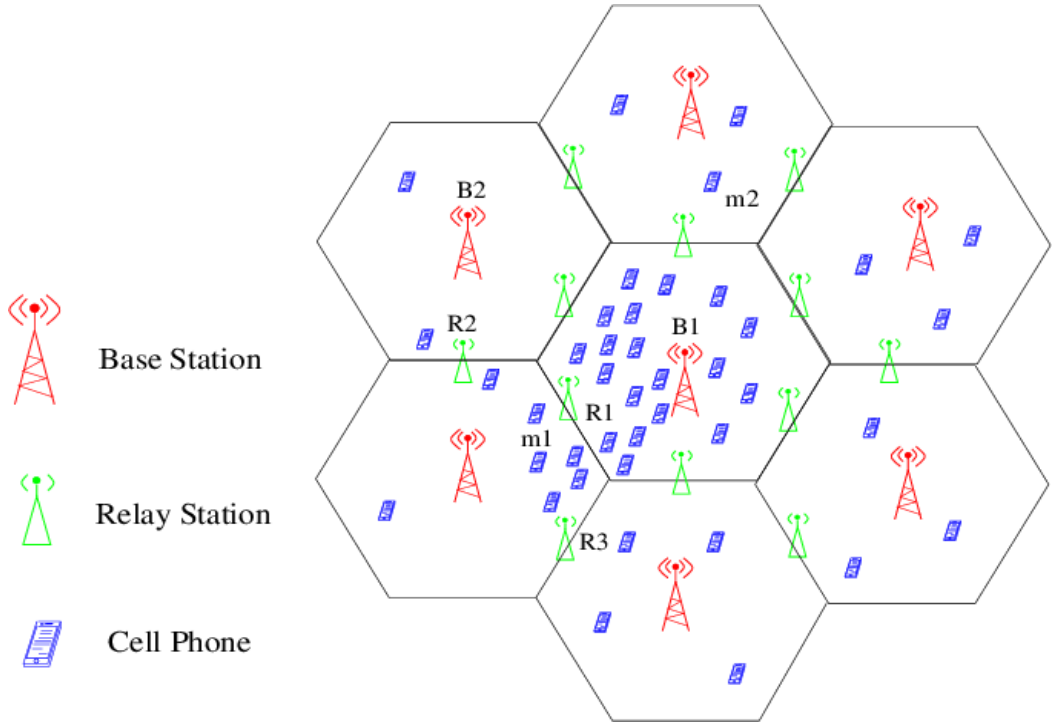
Computational Problem

- Input: A sequence of numbers <a1, a2, ….. , an>
- Output: A permutation <b1, b2, ….. , bn> of the input sequence such that b1≤b2≤ ….. ≤ bn.

Approaches?

Computational Problem

- Input: A set of base stations and clients.
- Output: The minimum number of frequencies required to be assigned to base stations for proper communication.

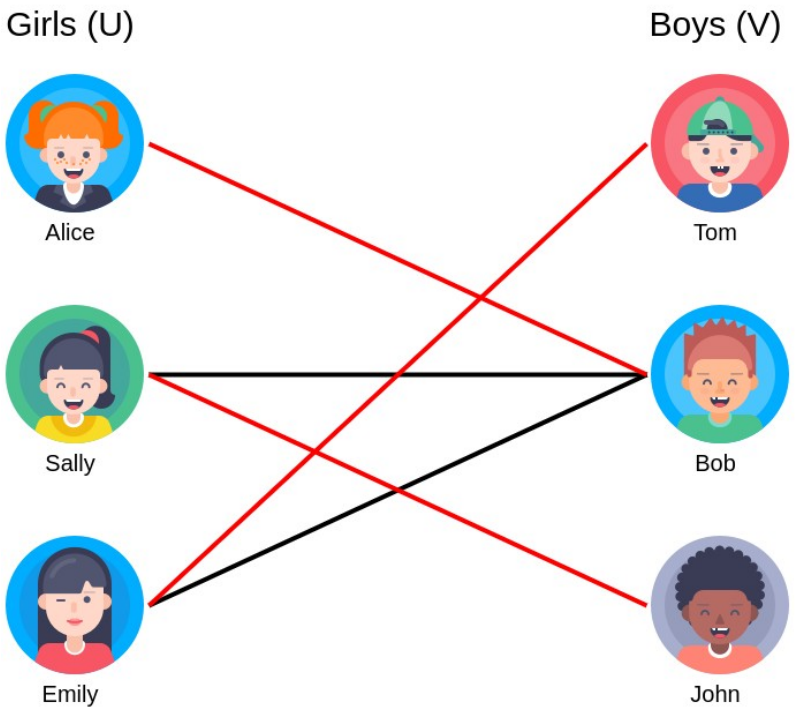


Computational Problem

- Input: Map
- Output: The minimum number of colors required to color the states such that
    - States that share a boundary are
    - colored distinctly.

Computational Problem

- Input: A set of Girls and Boys, and their compatibilities.
- Output: Find the maximum number of pairs that are compatible.

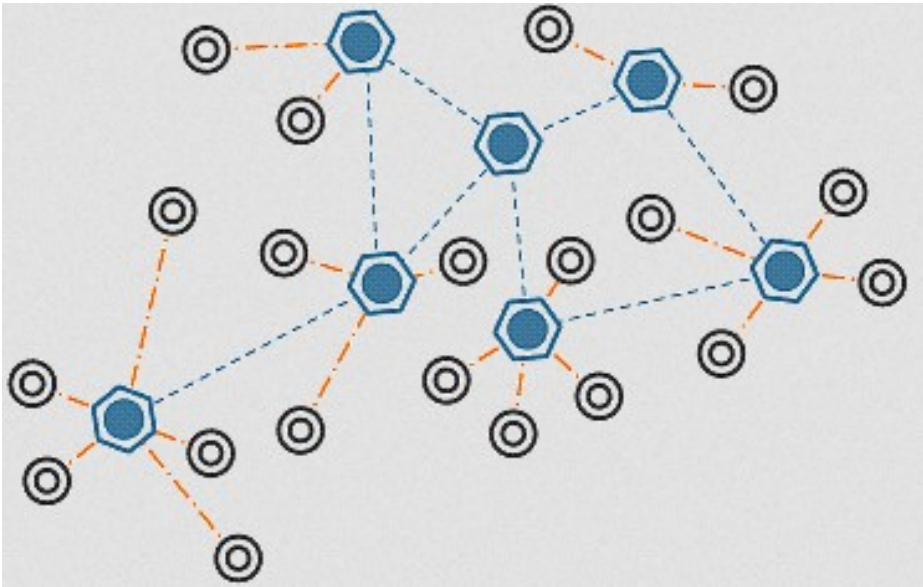


Computational Problem

- Input: Potential surveillance points, and the underlying graph.
- Output: The minimum number of cameras to be installed such that each region in covered.

Computational Problem

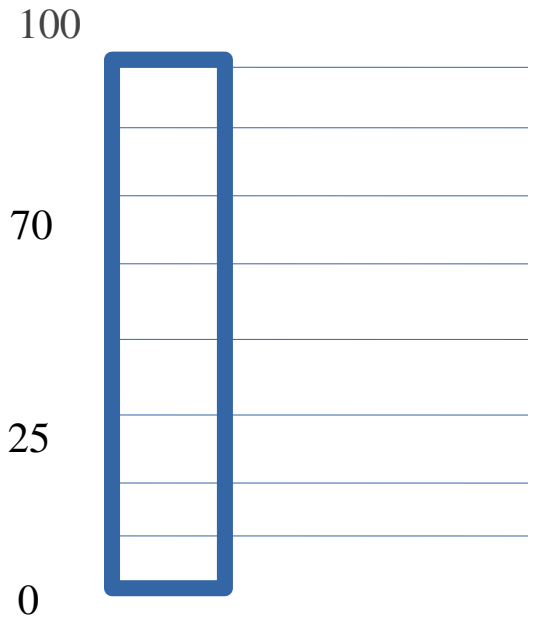
- Input: Potential surveillance points, and the underlying graph.
- Output: The minimum number of cameras to be installed such that each region in covered.



Computational Problem *

- Input: 2 eggs, and a 100-floor building
- Output: Find the highest floor from which you can drop an egg without breaking it, using the minimum number of drops.

Approaches ?

- 

How to Solve these problems?

- Algorithms using
  - ✓ Recursion
  - ✓ Divide and conquer
  - ✓ Dynamic programming
  - ✓ Greedy
- Reduction to an known problem
- Advanced techniques like combinatorial optimization, LP, randomization, advanced data structures, etc.

Correct Algorithm

- Problem: find the minimum of all the numbers in a list.

1. Scan through the list and find the maximum element

2. Output that element.

Alg-1

1. Scan through the list and find the minimum element

2. Output that element.

Alg-2

Correct Algorithm

- Problem: find the minimum of all the numbers in a list.

1. Scan through the list and find the maximum element

2. Output that element.

Alg-1

1. Scan through the list and find the minimum element

2. Output that element.

Alg-2

Correct Algorithm

- Problem: find the minimum of all the numbers in a list.

1. Output the first element of the list

Alg-1

1. Scan through the list and find the minimum element

2. Output that element.

Alg-2

Correct Algorithm

- Problem: find the minimum of all the numbers in a list.

1. Output the first element of the list

Alg-1

1. Scan through the list and find the minimum element

2. Output that element.

Alg-2

Correct Algorithm

- An algorithm
  - ✓ is **correct** if it returns the **correct answer**
    - ✗ for all the instances.

1. Output the first element of the list

Alg-1

1. Scan through the list and find the minimum element

2. Output that element.

Alg-2

Correct Algorithm

- An algorithm for a computational problem
  - ✓ is **CORRECT** if it returns the **correct answer**
    - ✗ for all the instances.

1. Output the first element of the list

Alg-1

1. Scan through the list and find the minimum element

2. Output that element.

Alg-2

What is the need to study algorithms if computers were infinitely fast?

- Algorithm should halt and output the correct solution on each instance.
- Any correct method for solving a problem is good enough.

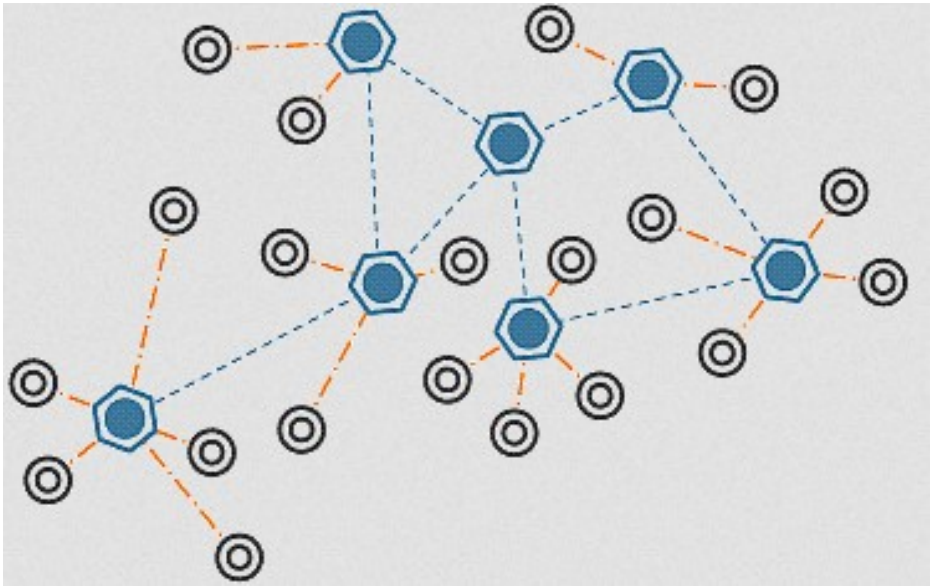
Computational Problem (revisited)

- Input: Map
- Output: The minimum number of colors required to color the states such that
  - States that share a boundary are
  - colored distinctly.

Approaches?

Computational Problem (revisited)

- Input: Potential surveillance points
- Output: The minimum number of cameras to be installed such that each region in covered.



Approaches?

What is the need to study algorithms if computers were infinitely fast?

- Algorithm should halt and output the correct solution on each instance.
- Any correct method for solving a problem is good enough.

Computers are not infinitely fast!

- Computing time is a bounded resource.
- Storage space is another resource.

Models of Computation vs Computers

- Model of Computation: an idealized mathematical construct that describes the primitive instructions and other details
- Computer: an actual physical device that implements a very specific model of computation

Question: What model of computation will we use to design algorithms? (in this course)

RAM (Random-Access Machine)

Informal description:

- Basic data types are integer number/float/character
- Instructions execute one after another (no concurrent operations)
- Each instruction takes the same amount of time
- Each data access or storing – takes the same amount of time.

• RAM (Random-Access Machine)

Informal description:

- Basic data types are integer number/float/character
- Instructions execute one after another (no concurrent operations)
- Each instruction takes the same amount of time
- Each data access or storing – takes the same amount of time.

In this course: we design algorithms assuming RAM model

Correct Algorithm

- Problem: find the minimum of all the numbers in a **sorted** list (say ascending order).

1. Output the first element of the list

Alg-1

1. Scan through the list and find the minimum element

2. Output that element.

Alg-2

Correct Algorithm

- Problem: find the minimum of all the numbers in a **sorted** list (say ascending order).

1. Output the first element of the list

Alg-1

1. Scan through the list and find the minimum element

2. Output that element.

Alg-2

Computational Problem

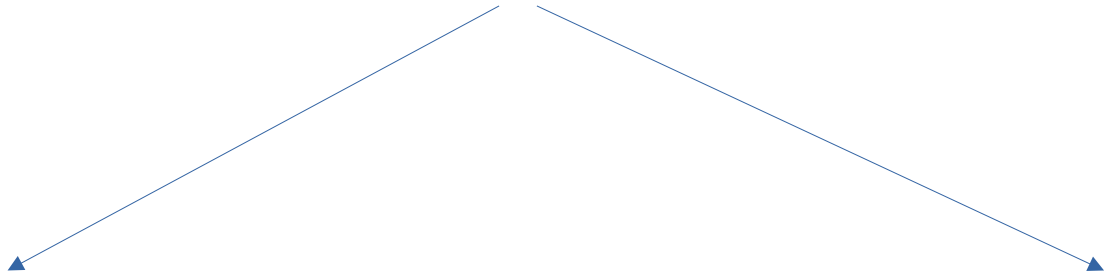
- Input: A sequence of numbers <a1, a2, ….. , an>
- Output: A permutation <b1, b2, ….. , bn> of the input sequence such that b1≤b2≤ ….. ≤ bn.

Approaches?

Algorithm

- Correctness of the algorithm
- Resources (time and space)

Computational Problem



Optimization Problem

Decision Problem

“Optimize the value”

“YES/NO”

Optimization/Decision Problem

GRAPH COLORING

Optimization/Decision Problem

GRAPH COLORING

min number of colors for a proper coloring of the graph

Optimization/Decision Problem

GRAPH COLORING

min number of colors for a proper coloring of the graph

(s,t)-SHORTEST PATH

Optimization/Decision Problem

GRAPH COLORING

min number of colors for a proper coloring of the graph

(s,t)-SHORTEST PATH

length of a shortest path between two vertices s and t

Optimization/Decision Problem

GRAPH COLORING

min number of colors for a proper coloring of the graph

(s,t)-SHORTEST PATH

length of a shortest path between two vertices s and t

HAMILTONIAN CYCLE

Optimization/Decision Problem

GRAPH COLORING

min number of colors for a proper coloring of the graph

(s,t)-SHORTEST PATH

length of a shortest path between two vertices s and t

HAMILTONIAN CYCLE

does there exist a cycle that contains every vertex of the graph

Optimization/Decision Problem

GRAPH COLORING

min number of colors for a proper coloring of the graph

(s,t)-SHORTEST PATH

length of a shortest path between two vertices s and t

HAMILTONIAN CYCLE

does there exist a cycle that contains every vertex of the graph

MST

Optimization/Decision Problem

GRAPH COLORING

min number of colors for a proper coloring of the graph

(s,t)-SHORTEST PATH

length of a shortest path between two vertices s and t

HAMILTONIAN CYCLE

does there exist a cycle that contains every vertex of the graph

MST

minimum weight of a tree that contains every vertex of the graph

SORTING

Optimization or Decision version?

SORTING

Optimization or Decision version?

Input: A sequence of numbers <a1, a2, ….. , an>
Output: A permutation <b1, b2, ….. , bn> of the input sequence such that $b1 \leq b2 \leq \ldots\ldots \leq bn$.

Suggestions?

- Maximize the sum of difference between consecutive elements?
- Minimize the sum of difference between consecutive elements?
- Does there exist a permutation in increasing sequence?
- ….

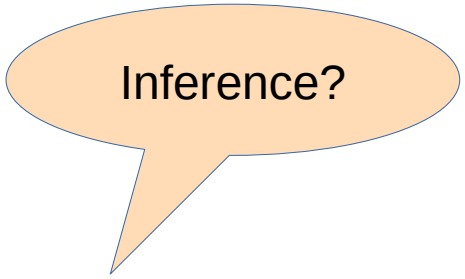
SORTING

Optimization or Decision version?

Input: A sequence of numbers <a1, a2, ….. , an>
Output: A permutation <b1, b2, ….. , bn> of the input sequence such that b1≤b2≤ ….. ≤ bn.

Suggestions?

- Maximize the sum of difference between consecutive elements?



SORTING

Optimization or Decision version?

Input: A sequence of numbers <a1, a2, ….. , an>
Output: A permutation <b1, b2, ….. , bn> of the input sequence such that b1≤b2≤ ….. ≤ bn.

Suggestions?

- Maximize the sum of difference between consecutive elements? $(b_{i+1}-b_i)$

- Maximize the sum of difference between consecutive elements? (b_i-b_{i+1})

SORTING

Optimization or Decision version?

Input: A sequence of numbers <a1, a2, ….. , an>
Output: A permutation <b1, b2, ….. , bn> of the input sequence such that b1≤b2≤ ….. ≤ bn.

Suggestions?

- Maximize the sum of difference between consecutive elements? $(b_{i+1}-b_i)$
  <10, 11, 12, 13> vs <10, 12, 11, 13>

- Maximize the sum of difference between consecutive elements? (b_i-b_{i+1})

SORTING

Optimization or Decision version?

Input: A sequence of numbers <a1, a2, ….. , an>
Output: A permutation <b1, b2, ….. , bn> of the input sequence such that b1≤b2≤ ….. ≤ bn.

Suggestions?

- Maximize the sum of difference between consecutive elements? $(b_{i+1}-b_i)$
  <10, 11, 12, 13> vs <10, 12, 11, 13>

- Maximize the sum of difference between consecutive elements? (b_i-b_{i+1})
  <10, 11, 12, 13> vs <13, 12, 11, 10>

SORTING

Optimization or Decision version?

Input: A sequence of numbers <a1, a2, ….. , an>
Output: A permutation <b1, b2, ….. , bn> of the input sequence such that $b1 \leq b2 \leq \ldots\ldots \leq bn$.

Suggestions?

- Maximize the sum of difference between consecutive elements? $(b_{i+1}-b_i)$
  <10, 11, 12, 13> vs <10, 12, 11, 13>

- Maximize the sum of difference between consecutive elements? (b_i-b_{i+1})
  <10, 11, 12, 13> vs <13, 12, 11, 10>

- Minimize the sum of difference between consecutive elements? (b_i-b_{i+1})

SORTING

Optimization or Decision version?

Input: A sequence of numbers <a1, a2, ….. , an>
Output: A permutation <b1, b2, ….. , bn> of the input sequence such that b1≤b2≤ ….. ≤ bn.

Suggestions?

- Maximize the sum of difference between consecutive elements? $(b_{i+1}-b_i)$
  <10, 11, 12, 13> vs <10, 12, 11, 13>

- Maximize the sum of difference between consecutive elements? (b_i-b_{i+1})
  <10, 11, 12, 13> vs <13, 12, 11, 10>

- Minimize the sum of difference between consecutive elements? (b_i-b_{i+1})
  <10, 11, 12, 13> vs <10, 12, 11, 13>

SORTING

Optimization or Decision version?

Input: A sequence of numbers <a1, a2, ….. , an>
Output: A permutation <b1, b2, ….. , bn> of the input sequence such that b1≤b2≤ ….. ≤ bn.

Suggestions?

- Maximize the sum of difference between consecutive elements? $(b_{i+1}-b_i)$
  <10, 11, 12, 13> vs <10, 12, 11, 13>

- Maximize the sum of difference between consecutive elements? (b_i-b_{i+1})
  <10, 11, 12, 13> vs <13, 12, 11, 10>

- Minimize the sum of difference between consecutive elements? (b_i-b_{i+1})
  <10, 11, 12, 13> vs <10, 12, 11, 13>

- Does there exist a permutation in increasing sequence?
- ….

Questions

Is GRAPH COLORING polynomial time solvable?

Questions

Is GRAPH COLORING polynomial time solvable?

Is MAP COLORING polynomial time solvable?

Questions

Can you formulate GRAPH COLORING as a decision problem?

Can you formulate GRAPH COLORING as an optimization problem?

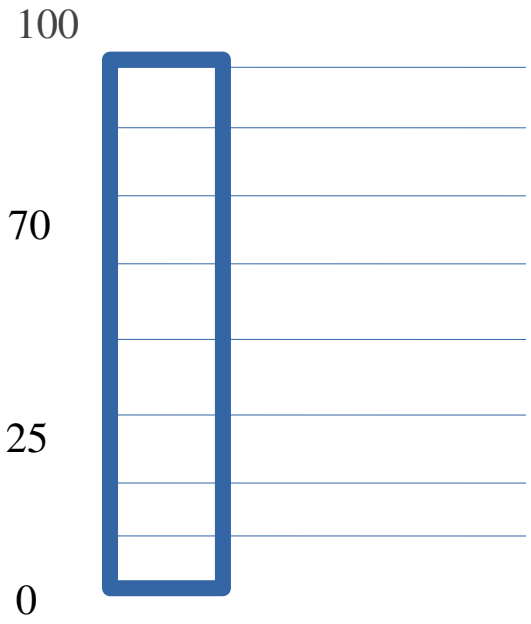
Questions

Can you use decision version to find the optimum value, for GRAPH COLORING?

Computational Problem *

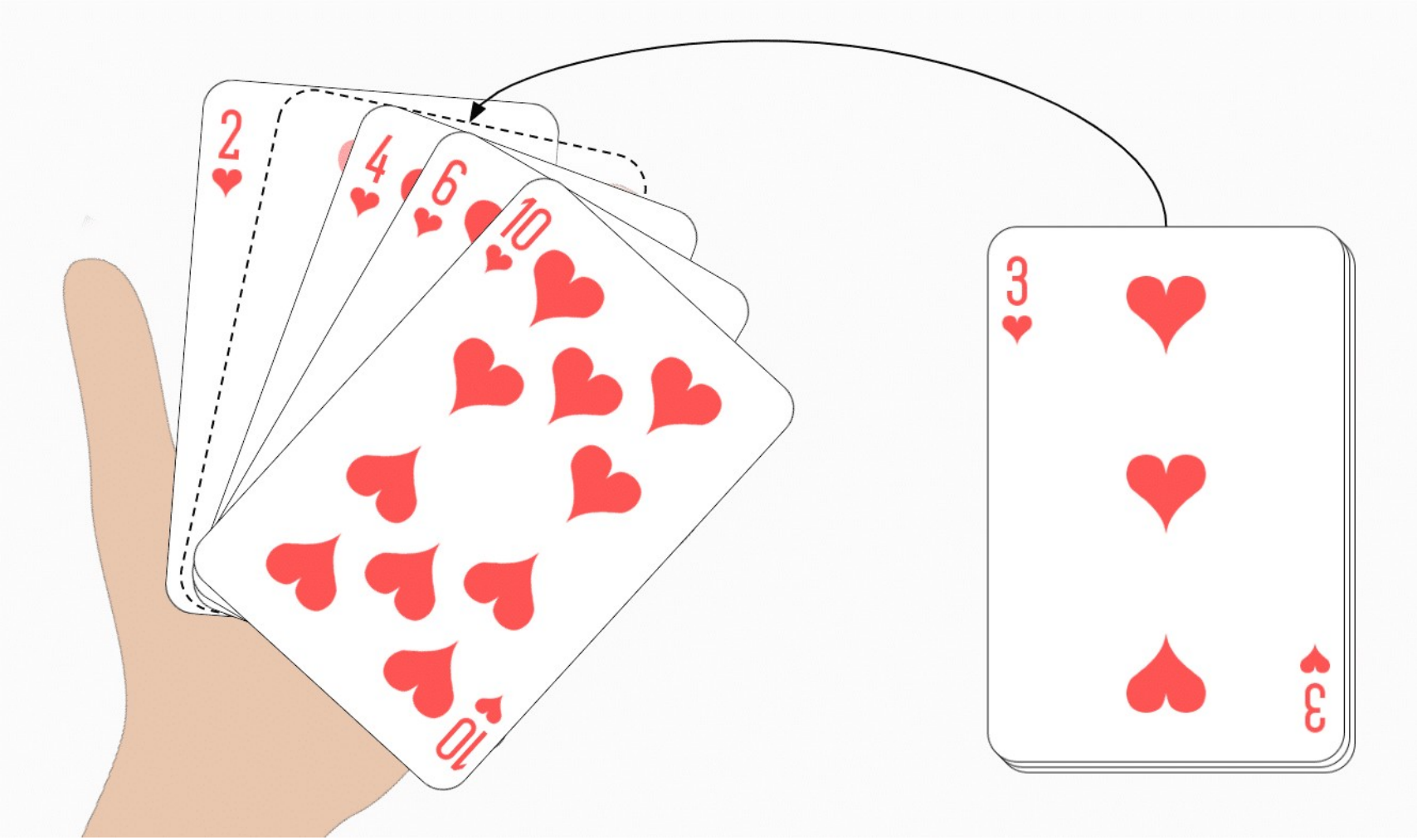
- Input: 2 eggs, and a 100-floor building
- Output: Find the highest floor from which you can drop an egg without breaking it, using the minimum number of drops.

Approaches ?

- 

INSERTION-SORT

INSERTION-SORT



INSERTION-SORT

EXAMPLE: INSERTION-SORT

```
INSERTION-SORT(A, n)
1   for i = 2 to n
2       key = A[i]
3       // Insert A[i] into the sorted subarray A[1 : i − 1].
4       j = i − 1
5       while j > 0 and A[j] > key
6           A[j + 1] = A[j]
7           j = j − 1
8       A[j + 1] = key
```

INSERTION-SORT

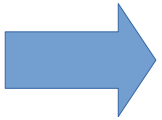
Will you get the same run time of the code when you run multiple times?

INSERTION-SORT

NO

Will you get the same run time of the code when you run multiple times?

So, we better analyse the algorithm itself



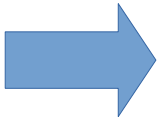
- How many times each line of pseudocode is executed?
- How long each line takes to run?

INSERTION-SORT

NO

Will you get the same run time of the code when you run multiple times?

So, we better analyse the algorithm itself



- How many times each line of pseudocode is executed?
- How long each line takes to run?

Find the dominating factor (to analyze other algorithms)

Design and Analysis of Algorithms

Analyzing time complexity

Exercise: For each function f(n) and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes f(n) microseconds.

	1 second (10^6 us)	1 minute (6*10^7 us)	1 hour (3.6 * 10^9 us)	1 day (8.64*10^9 us)
log n				
n				
n log n				
n^2				
n^3				
2^n				
n!				

Analyzing time complexity

Exercise: For each function f(n) and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes f(n) microseconds.

	1 second (10^6 us)	1 minute (6*10^7 us)	1 hour (3.6 * 10^9 us)	1 day (8.64*10^9 us)
log n	$2^{10^{\wedge}6}$			
n				
n log n				
n^2				
n^3				
2^n				
n!				

Analyzing time complexity

Exercise: For each function f(n) and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes f(n) microseconds.

	1 second (10^6 us)	1 minute (6*10^7 us)	1 hour (3.6 * 10^9 us)	1 day (8.64*10^9 us)
log n	2^{10^6}			
n	10^6			
n log n	~ 62,746			
n^2	~ 1,000			
n^3	~100			
2^n	~19			
n!	~9			

Analyzing time complexity

Exercise: For each function f(n) and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes f(n) microseconds.

	1 second (10^6 us)	1 minute (6*10^7 us)	1 hour (3.6 * 10^9 us)	1 day (8.64*10^9 us)
log n	2^{10^6}			$2^{8.64*10^9}$
n	10^6			
n log n	~62,746			
n^2	~1,000			
n^3	~100			
2^n	~19			~36
n!	~9			~16