Lecture 2

Formalising Problems

Problems as Functions

Problems as Functions

A **problem** for us would always mean **computing a function** $f : \{0,1\}^* \to \{0,1\}^*$, whose

Problems as Functions

A **problem** for us would always mean **computing a function** $f : \{0,1\}^* \to \{0,1\}^*$, whose input and output are finite length binary strings.

Problems as Functions

A **problem** for us would always mean **computing a function** $f : \{0,1\}^* \to \{0,1\}^*$, whose input and output are finite length binary strings.

We can express all problems as computing a function using **binary encoding**.

Problems as Functions

A **problem** for us would always mean **computing a function** $f : \{0,1\}^* \to \{0,1\}^*$, whose input and output are finite length binary strings.

We can express all problems as computing a function using **binary encoding**.

Examples:

Problems as Functions

A **problem** for us would always mean **computing a function** $f: \{0,1\}^* \to \{0,1\}^*$, whose input and output are finite length binary strings.

We can express all problems as computing a function using **binary encoding**.

Examples:

PRIMES: Given an integer x, decide if x is a prime.

Problems as Functions

A **problem** for us would always mean **computing a function** $f : \{0,1\}^* \to \{0,1\}^*$, whose input and output are finite length binary strings.

We can express all problems as computing a function using **binary encoding**.

Examples:

PRIMES: Given an integer x, decide if x is a prime.

$f_{PRIMES} : \{0,1\}^* \to \{0,1\}^*$ such that

Problems as Functions

A **problem** for us would always mean **computing a function** $f : \{0,1\}^* \to \{0,1\}^*$, whose input and output are finite length binary strings.

We can express all problems as computing a function using **binary encoding**.

Examples:

PRIMES: Given an integer x, decide if x is a prime.

$f_{PRIMES} : \{0,1\}^* \to \{0,1\}^*$ such that

$$f_{PRIMES}(x) = \begin{cases} 1, & \text{if } dec(x) \text{ is a prime number} \\ 0, & \text{if } dec(x) \text{ is not a prime number} \end{cases}$$

Problems as Functions

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \rightarrow \{0,1\}^*$ such that

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \to \{0,1\}^*$ such that

$$f_{FACTOR}(x) = \begin{cases} 1, & \text{if } x \text{ is a binary encoding of } a \,\&\, b \text{ s.t. } a \text{ divides } b \\ 0, & \text{if } x \text{ is a binary encoding of } a \,\&\, b \text{ s.t. } a \text{ does not divide } b \end{cases}$$

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \to \{0,1\}^*$ such that

How can we encode two numbers in one binary string?

$$f_{FACTOR}(x) = \begin{cases} 1, & \text{if } x \text{ is a binary encoding of } a \;\&\; b \text{ s.t. } a \text{ divides } b \\ 0, & \text{if } x \text{ is a binary encoding of } a \;\&\; b \text{ s.t. } a \text{ does not divide } b \end{cases}$$

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \rightarrow \{0,1\}^*$ such that

How can we encode two numbers in one binary string?

$$f_{FACTOR}(x) = \begin{cases} 1, & \text{if } x \text{ is a binary encoding of } a \;\&\; b \text{ s.t. } a \text{ divides } b \\ 0, & \text{if } x \text{ is a binary encoding of } a \;\&\; b \text{ s.t. } a \text{ does not divide } b \end{cases}$$

Good Encoding Practices:

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \to \{0,1\}^*$ such that

How can we encode two numbers in one binary string?

$$f_{FACTOR}(x) = \begin{cases} 1, & \text{if } x \text{ is a binary encoding of } a \;\&\; b \text{ s.t. } a \text{ divides } b \\ 0, & \text{if } x \text{ is a binary encoding of } a \;\&\; b \text{ s.t. } a \text{ does not divide } b \end{cases}$$

Good Encoding Practices:

- If $p \neq q$, then $\langle p \rangle \neq \langle q \rangle$

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \to \{0,1\}^*$ such that

How can we encode two numbers in one binary string?

$$f_{FACTOR}(x) = \begin{cases} 1, & \text{if } x \text{ is a binary encoding of } a \text{ \& } b \text{ s.t. } a \text{ divides } b \\ 0, & \text{if } x \text{ is a binary encoding of } a \text{ \& } b \text{ s.t. } a \text{ does not divide } b \end{cases}$$

Good Encoding Practices:

- If $p \neq q$, then $\langle p \rangle \neq \langle q \rangle$, where $\langle p \rangle$ denotes binary encoding of p.

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \rightarrow \{0,1\}^*$ such that

How can we encode two numbers
in one binary string?

$$f_{FACTOR}(x) = \begin{cases} 1, & \text{if } x \text{ is a binary encoding of } a \text{ \& } b \text{ s.t. } a \text{ divides } b \\ 0, & \text{if } x \text{ is a binary encoding of } a \text{ \& } b \text{ s.t. } a \text{ does not divide } b \end{cases}$$

Good Encoding Practices:

- If $p \neq q$, then $\langle p \rangle \neq \langle q \rangle$, where $\langle p \rangle$ denotes binary encoding of p.
- Getting p from $\langle p \rangle$ must be "easy".

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \to \{0,1\}^*$ such that

How can we encode two numbers in one binary string?

$$f_{FACTOR}(x) = \begin{cases} 1, & \text{if } x \text{ is a binary encoding of } a \text{ \& } b \text{ s.t. } a \text{ divides } b \\ 0, & \text{if } x \text{ is a binary encoding of } a \text{ \& } b \text{ s.t. } a \text{ does not divide } b \end{cases}$$

Good Encoding Practices:

- If $p \neq q$, then $\langle p \rangle \neq \langle q \rangle$, where $\langle p \rangle$ denotes binary encoding of p.
- Getting p from $\langle p \rangle$ must be "easy".
- Every binary string must correspond to some object.

Problems as Functions

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \to \{0,1\}^*$ such that

How can we encode two numbers in one binary string?

$$f_{FACTOR}(x) = \begin{cases} 1, & \text{if } x \text{ is a binary encoding of } a \,\&\, b \text{ s.t. } a \text{ divides } b \\ 0, & \text{if } x \text{ is a binary encoding of } a \,\&\, b \text{ s.t. } a \text{ does not divide } b \end{cases}$$

Good Encoding Practices:

- If $p \neq q$, then $\langle p \rangle \neq \langle q \rangle$, where $\langle p \rangle$ denotes binary encoding of p.
- Getting p from $\langle p \rangle$ must be "easy".
- Every binary string must correspond to some object.
- $\langle p \rangle$ shouldn't be too long.

Encoding Integer Tuples/Pairs

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1:

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1:

- Convert a and b into binary and concatenate them.

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1:

- Convert a and b into binary and concatenate them.

For instance,

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1:

- Convert a and b into binary and concatenate them.

For instance,

$$(3,5)$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1:

- Convert a and b into binary and concatenate them.

For instance,

$$(3,5) \longrightarrow 11101$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1:

- Convert a and b into binary and concatenate them.

For instance,

$$(3,5) \longrightarrow 11101 \qquad\qquad (14,1)$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1:

- Convert a and b into binary and concatenate them.

For instance,

$$(3,5) \longrightarrow 11101 \qquad\qquad (14,1) \longrightarrow 11101$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1:

- Convert a and b into binary and concatenate them.

Different pairs have same encoding

For instance,

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

Different pairs have same encoding

For instance,

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

For instance,

Different pairs have same encoding

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Method 2:

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

Different pairs have same encoding

For instance,

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Method 2:

- Convert a into binary and further replace each 0 by 00 and 1 by 11.

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

Different pairs have same encoding

For instance,

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Method 2:

- Convert a into binary and further replace each 0 by 00 and 1 by 11. Do the same for b.

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

Different pairs have same encoding

For instance,

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Method 2:

- Convert a into binary and further replace each 0 by 00 and 1 by 11. Do the same for b.
- Concatenate both the binary strings with a 01 in the middle.

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

For instance,

Different pairs have same encoding

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Method 2:

- Convert a into binary and further replace each 0 by 00 and 1 by 11. Do the same for b.
- Concatenate both the binary strings with a 01 in the middle.

For instance,

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

Different pairs have same encoding

For instance,

$$(3{,}5) \longrightarrow 11101 \qquad (14{,}1) \longrightarrow 11101$$

Method 2:

- Convert a into binary and further replace each 0 by 00 and 1 by 11. Do the same for b.
- Concatenate both the binary strings with a 01 in the middle.

For instance,

$$(3{,}5)$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

Different pairs have same encoding

For instance,

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Method 2:

- Convert a into binary and further replace each 0 by 00 and 1 by 11. Do the same for b.
- Concatenate both the binary strings with a 01 in the middle.

For instance,

$$(3,5) \longrightarrow 1111\textcolor{blue}{01}110011$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

Different pairs have same encoding

For instance,

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Method 2:

- Convert a into binary and further replace each 0 by 00 and 1 by 11. Do the same for b.
- Concatenate both the binary strings with a 01 in the middle.

For instance,

$$(3,5) \longrightarrow 1111\color{blue}{01}\color{black}110011 \qquad 110000\color{blue}{01}\color{black}1100$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

For instance,

Different pairs have same encoding

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Method 2:

- Convert a into binary and further replace each 0 by 00 and 1 by 11. Do the same for b.
- Concatenate both the binary strings with a 01 in the middle.

For instance,

$$(3,5) \longrightarrow 111101110011 \qquad (4,2) \longrightarrow 110000011100$$

Encoding Integer Tuples/Pairs

Let (a, b) be a pair of integers:

Method 1: ✗

- Convert a and b into binary and concatenate them.

For instance,

Different pairs have same encoding

$$(3,5) \longrightarrow 11101 \qquad (14,1) \longrightarrow 11101$$

Method 2: ✓

- Convert a into binary and further replace each 0 by 00 and 1 by 11. Do the same for b.
- Concatenate both the binary strings with a 01 in the middle.

For instance,

$$(3,5) \longrightarrow 1111\,01\,110011 \qquad (4,2) \longrightarrow 110000\,01\,1100$$

Search vs Decision Problem

Search vs Decision Problem

Search problems are computational problems where we have to **find** a solution or inform

Search vs Decision Problem

Search problems are computational problems where we have to **find** a solution or inform that no solutions exist.

Search vs Decision Problem

Search problems are computational problems where we have to **find** a solution or inform that no solutions exist. For instance,

Search vs Decision Problem

Search problems are computational problems where we have to **find** a solution or inform that no solutions exist. For instance,

SEARCH_PATH: Given a graph G and vertices $u, v \in G$, find a path from u to v or inform if no such paths exist.

Search vs Decision Problem

Search problems are computational problems where we have to **find** a solution or inform that no solutions exist. For instance,

SEARCH_PATH: Given a graph G and vertices $u, v \in G$, find a path from u to v or inform if no such paths exist.

Decision problems are computational problems where we have to **decide** whether a solution exists.

Search vs Decision Problem

Search problems are computational problems where we have to **find** a solution or inform that no solutions exist. For instance,

SEARCH_PATH: Given a graph G and vertices $u, v \in G$, find a path from u to v or inform if no such paths exist.

Decision problems are computational problems where we have to **decide** whether a solution exists. For instance,

Search vs Decision Problem

Search problems are computational problems where we have to **find** a solution or inform that no solutions exist. For instance,

SEARCH_PATH: Given a graph G and vertices $u,v \in G$, find a path from u to v or inform if no such paths exist.

Decision problems are computational problems where we have to **decide** whether a solution exists. For instance,

DEC_PATH: Given a graph G and vertices $u,v \in G$, decide if a path from u to v exist.

Search vs Decision Problem

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).
2. Answer **Yes** or **No** depending on the answer from the 1st step.

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).
2. Answer **Yes** or **No** depending on the answer from the 1st step.

Observation:

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).
2. Answer **Yes** or **No** depending on the answer from the 1st step.

Observation: If *SEARCH_PATH* is polynomial-time solvable, then so is *DEC_PATH*.

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).
2. Answer **Yes** or **No** depending on the answer from the 1st step.

Observation:

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).
2. Answer **Yes** or **No** depending on the answer from the 1st step.

Observation: If *DEC_PATH* is not polynomial-time solvable, then so is *SEARCH_PATH*.

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).
2. Answer **Yes** or **No** depending on the answer from the 1st step.

Observation: If *DEC_PATH* is not polynomial-time solvable, then so is *SEARCH_PATH*.

We focus on decision problems because:

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).
2. Answer **Yes** or **No** depending on the answer from the 1st step.

Observation: If *DEC_PATH* is not polynomial-time solvable, then so is *SEARCH_PATH*.

We focus on decision problems because:

- They are mathematically simple.

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).
2. Answer **Yes** or **No** depending on the answer from the 1st step.

Observation: If *DEC_PATH* is not polynomial-time solvable, then so is *SEARCH_PATH*.

We focus on decision problems because:

- They are mathematically simple.
- Lower bounds for decision problems implies lower bounds for search problems.

Search vs Decision Problem

Solving *DEC_PATH* via *SEARCH_PATH:*

1. Solve *SEARCH_PATH* on (G, u, v).
2. Answer **Yes** or **No** depending on the answer from the 1st step.

Observation: If *DEC_PATH* is not polynomial-time solvable, then so is *SEARCH_PATH*.

We focus on decision problems because:

- They are mathematically simple.
- Lower bounds for decision problems implies lower bounds for search problems.

Decision Problems as Boolean Functions as Languages

Decision Problems as Boolean Functions as Languages

A **boolean function** is a function of the form $f : \{0,1\}^* \to \{0,1\}$.

Decision Problems as Boolean Functions as Languages

A **boolean function** is a function of the form $f : \{0,1\}^* \to \{0,1\}$.

A **language** is a subset of $\{0,1\}^*$.

Decision Problems as Boolean Functions as Languages

A **boolean function** is a function of the form $f: \{0,1\}^* \to \{0,1\}$.

A **language** is a subset of $\{0,1\}^*$.

Note: Decision problems can be posed as boolean functions or as languages.

Decision Problems as Boolean Functions as Languages

A **boolean function** is a function of the form $f: \{0,1\}^* \to \{0,1\}$.

A **language** is a subset of $\{0,1\}^*$.

Note: Decision problems can be posed as boolean functions or as languages.

Example:

Decision Problems as Boolean Functions as Languages

A **boolean function** is a function of the form $f : \{0,1\}^* \to \{0,1\}$.

A **language** is a subset of $\{0,1\}^*$.

Note: Decision problems can be posed as boolean functions or as languages.

Example:

FACTOR: Given integers a and b, decide if a is a factor of b.

Decision Problems as Boolean Functions as Languages

A **boolean function** is a function of the form $f : \{0,1\}^* \to \{0,1\}$.

A **language** is a subset of $\{0,1\}^*$.

Note: Decision problems can be posed as boolean functions or as languages.

Example:

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \to \{0,1\}$ such that …

Decision Problems as Boolean Functions as Languages

A **boolean function** is a function of the form $f : \{0,1\}^* \to \{0,1\}$.

A **language** is a subset of $\{0,1\}^*$.

Note: Decision problems can be posed as boolean functions or as languages.

Example:

FACTOR: Given integers a and b, decide if a is a factor of b.

$f_{FACTOR} : \{0,1\}^* \to \{0,1\}$ such that …

$FACTOR = \{\langle a, b \rangle \mid a \text{ is a factor of } b\}$