CS 577- Intro to Algorithms

Reductions

Dieter van Melkebeek

November 10, 2020

Outline

Outline

Paradigm

Solve a computational problem A using a blackbox for another computational problem B.

Outline

Paradigm

Solve a computational problem A using a blackbox for another computational problem B.

Motivation

Outline

Paradigm

Solve a computational problem A using a blackbox for another computational problem B.

Motivation

- Modular design

Outline

Paradigm

Solve a computational problem A using a blackbox for another computational problem B.

Motivation

- Modular design
- NP-completeness

Outline

Paradigm

Solve a computational problem A using a blackbox for another computational problem B.

Motivation

- Modular design
- NP-completeness

Today

Outline

Paradigm

Solve a computational problem A using a blackbox for another computational problem B.

Motivation

- Modular design
- NP-completeness

Today

- Notion

Outline

Paradigm

Solve a computational problem A using a blackbox for another computational problem B.

Motivation

- Modular design
- NP-completeness

Today

- Notion
- Example where A and B have efficient algorithms

Outline

Paradigm

Solve a computational problem A using a blackbox for another computational problem B.

Motivation

- Modular design
- NP-completeness

Today

- Notion
- Example where A and B have efficient algorithms
- Examples where A and B have no (known) efficient algorithms

Outline

Paradigm

Solve a computational problem A using a blackbox for another computational problem B.

Motivation

- Modular design
- NP-completeness

Today

- Notion
- Example where A and B have efficient algorithms
- Examples where A and B have no (known) efficient algorithms: optimization vs search vs decision

Notion

Notion

Let A and B be two computational problems.

Notion

Let A and B be two computational problems.

Definition

A reduction from A to B is an algorithm for A that can make use of a blackbox for B.

Notion

Let A and B be two computational problems.

Definition

A reduction from A to B is an algorithm for A that can make use of a blackbox for B.

Queries

- On a given input x of problem A, reduction can make multiple queries x' to the blackbox for B.

Notion

Let A and B be two computational problems.

Definition

A reduction from A to B is an algorithm for A that can make use of a blackbox for B.

Queries

- On a given input x of problem A, reduction can make multiple queries x' to the blackbox for B.
- For a valid query x' of problem B, the blackbox returns a valid output y' for problem B on input x'.

Notion

Let A and B be two computational problems.

Definition

A reduction from A to B is an algorithm for A that can make use of a blackbox for B.

Queries

- On a given input x of problem A, reduction can make multiple queries x' to the blackbox for B.
- For a valid query x' of problem B, the blackbox returns a valid output y' for problem B on input x'.
- Often times one query suffices.

Bipartite Matching and Integral Max Flow

Bipartite Matching and Integral Max Flow

A : Bipartite Matching

Bipartite Matching and Integral Max Flow

A : Bipartite Matching

Input: bipartite graph $G = (V, E)$
where $V = L \sqcup R$ and $E \subseteq L \times R$

Bipartite Matching and Integral Max Flow

A : Bipartite Matching

Input: bipartite graph $G = (V, E)$
where $V = L \sqcup R$ and $E \subseteq L \times R$

Output: matching M such that $|M|$ is maximized

Bipartite Matching and Integral Max Flow

A : Bipartite Matching

Input: bipartite graph $G = (V, E)$
where $V = L \sqcup R$ and $E \subseteq L \times R$

Output: matching M such that $|M|$ is maximized

B : Integral Max Flow

Bipartite Matching and Integral Max Flow

A : Bipartite Matching

Input: bipartite graph $G = (V, E)$
where $V = L \sqcup R$ and $E \subseteq L \times R$

Output: matching M such that $|M|$ is maximized

B : Integral Max Flow

Input: network $N = (V', E', c, s, t)$

Bipartite Matching and Integral Max Flow

A : Bipartite Matching

Input: bipartite graph $G = (V, E)$
where $V = L \sqcup R$ and $E \subseteq L \times R$

Output: matching M such that $|M|$ is maximized

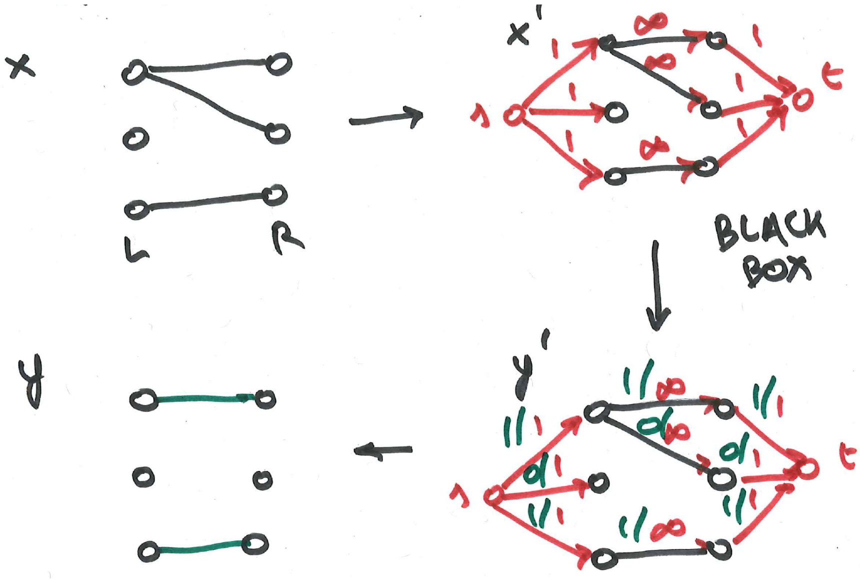
B : Integral Max Flow

Input: network $N = (V', E', c, s, t)$

Output: integral flow f such that $\nu(f) \doteq f_{\text{out}}(s)$ is maximized

Reduction from Bipartite Matching to Integral Max Flow

Reduction from Bipartite Matching to Integral Max Flow



Correctness of a Reduction

Correctness of a Reduction

Definition

On every valid input x of A:

Correctness of a Reduction

Definition

On every valid input x of A:

- Each query x' to the blackbox is valid input of B.

Correctness of a Reduction

Definition

On every valid input x of A:

- Each query x' to the blackbox is valid input of B.
- Assuming all queries to the blackbox are answered correctly, the reduction produces a correct output y for A on input x.

Correctness of a Reduction

Definition

On every valid input x of A:

- Each query x' to the blackbox is valid input of B.
- Assuming all queries to the blackbox are answered correctly, the reduction produces a correct output y for A on input x.

Corollary

Replacing the blackbox for B by a correct algorithm for B yields a correct algorithm for A.

Reduciblity

Reduciblity

Definition

$A \leq B$ if there exists a reduction from A to B.

Reducibility

Definition

$A \leq B$ if there exists a reduction from A to B.

Properties

Reduciblity

Definition

$A \leq B$ if there exists a reduction from A to B.

Properties

- Reflexive: $A \leq A$

Reducibliity

Definition

$A \leq B$ if there exists a reduction from A to B.

Properties

- Reflexive: $A \leq A$
- Not symmetric: Bipartite Matching $\leq$ Halting Problem, but not the other way.

Reduciblity

Definition

$A \leq B$ if there exists a reduction from A to B.

Properties

- Reflexive: $A \leq A$
- Not symmetric: Bipartite Matching $\leq$ Halting Problem, but not the other way.
- Transitive: $A \leq B$ and $B \leq C$ implies $A \leq C$.

Reduciblity

Definition

$A \leq B$ if there exists a reduction from A to B.

Properties

- Reflexive: $A \leq A$
- Not symmetric: Bipartite Matching $\leq$ Halting Problem, but not the other way.
- Transitive: $A \leq B$ and $B \leq C$ implies $A \leq C$.
- If $A \leq B$ and B can be solved algorithmically, then A can be solved algorithmically.

Running Time of a Reduction

Running Time of a Reduction

Definition

Time to run the reduction assuming blackbox queries are answered instantaneously.

Running Time of a Reduction

Definition

Time to run the reduction assuming blackbox queries are answered instantaneously.

Case of one query:

Running Time of a Reduction

Definition

Time to run the reduction assuming blackbox queries are answered instantaneously.

Case of one query:

Running time of reduction consists of:

Running Time of a Reduction

Definition

Time to run the reduction assuming blackbox queries are answered instantaneously.

Case of one query:

Running time of reduction consists of:

- Time to construct out of the input x to A the query x' to B .

Running Time of a Reduction

Definition

Time to run the reduction assuming blackbox queries are answered instantaneously.

Case of one query:

Running time of reduction consists of:

- Time to construct out of the input x to A the query x' to B .
- Time to construct out of the answer y' of B to query x', the answer y for A on input x.

Running Time of a Reduction

Definition

Time to run the reduction assuming blackbox queries are answered instantaneously.

Case of one query:

Running time of reduction consists of:

- Time to construct out of the input x to A the query x' to B .
- Time to construct out of the answer y' of B to query x', the answer y for A on input x.
- Not time to compute y' out of x'.

Running Time of a Reduction

Definition

Time to run the reduction assuming blackbox queries are answered instantaneously.

Case of one query:

Running time of reduction consists of:

- Time to construct out of the input x to A the query x' to B .
- Time to construct out of the answer y' of B to query x', the answer y for A on input x.
- Not time to compute y' out of x'.

Corollary

Suppose reduction from A to B runs in time t. Replacing the blackbox for B by an algorithm for B that runs in time $t_B(n)$ yields an algorithm for A that runs in time $t + t \cdot t_B(t)$.

Polynomial Time

Polynomial Time

Bit-length

The bit-length of an input x is the number of bits needed to represent x.

Polynomial Time

Bit-length

The bit-length of an input x is the number of bits needed to represent x.

- binary strings: length

Polynomial Time

Bit-length

The bit-length of an input x is the number of bits needed to represent x.

- binary strings: length
- numbers: length of the binary representation (finite precision)

Polynomial Time

Bit-length

The bit-length of an input x is the number of bits needed to represent x.

- binary strings: length
- numbers: length of the binary representation (finite precision)
- graphs: $O(n^2)$ for adjacency matrix, $O(n + m \log n)$ for adjacency list

Polynomial Time

Bit-length

The bit-length of an input x is the number of bits needed to represent x.

- binary strings: length
- numbers: length of the binary representation (finite precision)
- graphs: $O(n^2)$ for adjacency matrix, $O(n + m \log n)$ for adjacency list
- ...

Polynomial Time

Bit-length

The bit-length of an input x is the number of bits needed to represent x.

- binary strings: length
- numbers: length of the binary representation (finite precision)
- graphs: $O(n^2)$ for adjacency matrix, $O(n + m \log n)$ for adjacency list
- ...

Definition

An algorithm/reduction runs in polynomial time if its running time is $O(n^c)$ for some constant c, where $n \doteq$ bit-length of the input.

Polynomial Time

Bit-length

The bit-length of an input x is the number of bits needed to represent x.

- binary strings: length
- numbers: length of the binary representation (finite precision)
- graphs: $O(n^2)$ for adjacency matrix, $O(n + m \log n)$ for adjacency list
- ...

Definition

An algorithm/reduction runs in polynomial time if its running time is $O(n^c)$ for some constant c, where $n \doteq$ bit-length of the input.

Robustness

Notion turns out to be the same for most (but perhaps not all) reasonable input representations and models of computation.

Polynomial–Time Reduciblity

Polynomial–Time Reduciblity

Definition

$A \leq^p B$ if there exists a polynomial-time reduction from A to B.

Polynomial–Time Reduciblity

Definition

$A \leq^p B$ if there exists a polynomial-time reduction from A to B.

Properties

Polynomial–Time Reduciblity

Definition

$A \leq^p B$ if there exists a polynomial-time reduction from A to B.

Properties

- Reflexive: $A \leq^p A$

Polynomial–Time Reduciblity

Definition

$A \leq^p B$ if there exists a polynomial-time reduction from A to B.

Properties

- Reflexive: $A \leq^p A$
- Not symmetric

Polynomial–Time Reduciblity

Definition

$A \leq^p B$ if there exists a polynomial-time reduction from A to B.

Properties

- Reflexive: $A \leq^p A$
- Not symmetric
- Transitive: $A \leq^p B$ and $B \leq^p C$ implies $A \leq^p C$.

Polynomial–Time Reducibllity

Definition

$A \leq^p B$ if there exists a polynomial-time reduction from A to B.

Properties

- Reflexive: $A \leq^p A$
- Not symmetric
- Transitive: $A \leq^p B$ and $B \leq^p C$ implies $A \leq^p C$.
- If $A \leq^p B$ and B can be solved in polynomial time, then A can be solved in polynomial time.

Independent Set Problems

Independent Set Problems

Definition

An independent set in a graph $G = (V, E)$ is a subset $S \subseteq V$ such that $E \cap S \times S = \emptyset$.

Independent Set Problems

Definition

An independent set in a graph $G = (V, E)$ is a subset $S \subseteq V$ such that $E \cap S \times S = \emptyset$.

Optimization problem

Input: graph G

Output: independent set S of G such that $|S|$ is maximized

Independent Set Problems

Definition

An independent set in a graph $G = (V, E)$ is a subset $S \subseteq V$ such that $E \cap S \times S = \emptyset$.

Optimization problem

Input: graph G

Output: independent set S of G such that $|S|$ is maximized

Search problem

Input: graph G, $k \in \mathbb{N}$

Output: independent set S of G such that $|S| \geq k$, or report that no such set exists

Independent Set Problems

Definition

An independent set in a graph $G = (V, E)$ is a subset $S \subseteq V$ such that $E \cap S \times S = \emptyset$.

Optimization problem

Input: graph G

Output: independent set S of G such that $|S|$ is maximized

Search problem

Input: graph G, $k \in \mathbb{N}$

Output: independent set S of G such that $|S| \geq k$, or report that no such set exists

Decision problem

Input: graph G, $k \in \mathbb{N}$

Output: whether independent set S with $|S| \geq k$ exists in G

Independent Set – Decision $\leq^p$ Search $\leq^p$ Optimization

Independent Set – Decision $\leq^p$ Search $\leq^p$ Optimization

- Decision $\leq^p$ Search

Independent Set – Decision $\leq^p$ Search $\leq^p$ Optimization

- Decision $\leq^p$ Search

  if Search(G, k) = "no solution" **then**
      return "no"
  else
      return "yes"

Independent Set – Decision $\leq^p$ Search $\leq^p$ Optimization

- Decision $\leq^p$ Search

  1: **if** Search(G, k) = "no solution" **then**
  2:     **return** "no"
  3: **else**
  4:     **return** "yes"

- Search $\leq^p$ Optimization

Independent Set – Decision $\leq^p$ Search $\leq^p$ Optimization

- Decision $\leq^p$ Search

  if Search(G, k) = "no solution" **then**
      return "no"
  else
      return "yes"

- Search $\leq^p$ Optimization

  $I \leftarrow$ Optimization(G)
  if $|I| \geq k$ **then**
      return I
  else
      return "no solution"

Independent Set – Optimization $\leq^p$ Search $\leq^p$ Decision

Independent Set – Optimization $\leq^p$ Search $\leq^p$ Decision

- Optimization $\leq^p$ Search

Independent Set – Optimization $\leq^p$ Search $\leq^p$ Decision

- Optimization $\leq^p$ Search

```
k ← 0
while Search(G, k) ≠ "no solution" do
    k ← k + 1
return Search(G, k)
```

Independent Set – Optimization $\leq^p$ Search $\leq^p$ Decision

- Optimization $\leq^p$ Search

  $k \leftarrow 0$
  while Search$(G, k) \neq$ "no solution" **do**
      $k \leftarrow k + 1$
  return Search(G, k)

  Number of queries can be reduced from $O(|V|)$ to $O(\log |V|)$ using binary search.

Independent Set – Optimization $\leq^p$ Search $\leq^p$ Decision

- Optimization $\leq^p$ Search

  ```
  k ← 0
  while Search(G, k) ≠ "no solution" do
      k ← k + 1
  return Search(G, k)
  ```

  Number of queries can be reduced from $O(|V|)$ to $O(\log |V|)$ using binary search.
- Search $\leq^p$ Decision: next lecture