CS 341: ALGORITHMS

Lecture 20: intractability II – complexity class NP

Readings: see website

Trevor Brown
https://student.cs.uwaterloo.ca/~cs341
trevor.brown@uwaterloo.ca

THIS TIME

- **Finishing TSP reductions**
- Complexity class **NP**
  - Oracles, certificates, polytime verification algorithms

RECALL

- So far we know
  - TSP-Dec $\le_P^T$ TSP-Optimal Value
  - TSP-Dec $\le_P^T$ TSP-Optimization
- **In progress**
  - **TSP-Optimal Value $\le_P^T$ TSP-Dec**

Travelling Salesperson Problems

Problem 7.5
TSP-Optimization
Instance: *A graph G and edge weights $w : E \to \mathbb{Z}^+$.*
Find: *A hamiltonian cycle H in G such that $w(H) = \sum_{e \in H} w(e)$ is minimized.*

Problem 7.6
TSP-Optimal Value
Instance: *A graph G and edge weights $w : E \to \mathbb{Z}^+$.*
Find: *The minimum T such that there exists a hamiltonian cycle H in G with $w(H) = T$.*

Problem 7.7
TSP-Decision
Instance: *A graph G, edge weights $w : E \to \mathbb{Z}^+$, and a target T.*
Question: *Does there exist a hamiltonian cycle H in G with $w(H) \le T$?*

What's the size of the input $I = (G, w)$?

$Size(I) = Size(G) + Size(w)$

So, suppose G is represented as an **array of adjacency lists** (one list for each vertex),
with each list containing **edges** to neighbouring vertices,
and an edge is represented by a **weight** and the **name** of the target vertex

Bits to store **weight** of the edge
(storing $w(e)$ takes $\log w(e) + 1$ bits)

Bits to store the **name** of the target vertex (in 1..|V|)

$$Size(I) = |V| + \underbrace{\sum_{e \in E} (\log w(e) + 1 + \log|V| + 1)}_{\textbf{For all edges}}$$

Array of empty lists for all vertices v

Let's relate this to runtime… what's the runtime?

TSP-Optimal Value $\le_P^T$ TSP-Dec

Let's assume $O(1)$ time for operations on weights

Technically not needed to show polytime.. But simplifies

Algorithm: *TSP-OptimalValue-Solver*(G, w)
 external *TSP-Dec-Solver*
 $hi \leftarrow \sum_{e \in E} w(e)$ — $O(|E|)$
 $lo \leftarrow 0$ — $O(1)$
 if not *TSP-Dec-Solver*(G, w, hi) **then return** (∞) — $O(1)$ for the oracle
 while $hi > lo$ — # iterations: $O(\log(hi - lo))$ $= \log \sum_{e \in E} w(e)$
  do $\left\{ \begin{array}{l} mid \leftarrow \lfloor \frac{hi + lo}{2} \rfloor \\ \textbf{if } \textit{TSP-Dec-Solver}(G, w, mid) \\ \quad \textbf{then } hi \leftarrow mid \\ \quad \textbf{else } lo \leftarrow mid + 1 \end{array} \right\}$ — $O(1)$
 return (hi)

Runtime $T(I) \in O(|E| + \log \sum_{e \in E} w(e))$

COMPARING $T(I)$ AND $Size(I)$

- $T(I) \quad \in O(|E| + \log \sum_{e \in E} w(e))$
- $Size(I) \quad = |V| + \sum_{e \in E} (\log w(e) + 1 + \log|V| + 1)$
  $= |V| + \Sigma_{e \in E} (\log w(e) + 1) + \boldsymbol{\Sigma_{e \in E}} (\log|V| + 1)$
  $= |V| + \Sigma_{e \in E} (\log w(e) + 1) + \Sigma_{e \in E} (\log|V|) + \boldsymbol{|E|}$
- Want to show $T(I) \in O(Size(I)^c)$ for some constant c (we show c=1)

$O(\boldsymbol{|E|} + \log \sum_{e \in E} w(e)) \subseteq^? O(|V| + \Sigma_{e \in E} (\log w(e) + 1) + \Sigma_{e \in E} \log|V| + \boldsymbol{|E|})$

$\Leftrightarrow O(\log \sum_{e \in E} w(e)) \subseteq^? O(|V| + \Sigma_{e \in E} (\log w(e) + 1) + \Sigma_{e \in E} \log|V|)$

How to compare $\log \sum_{e \in E} w(e)$ and $\Sigma_{e \in E} (\log w(e) + 1)$?

COMPARING $T(I)$ AND $Size(I)$

- **How to compare $\log \sum_{e \in E} w(e)$ and $\Sigma_{e \in E}(\log w(e) + 1)$?**
- $\Sigma_{e \in E}(\log w(e) + 1) = (\log w(e_1) + 1) + (\log w(e_2) + 1) + \cdots + \left(\log\left(w\left(e_{|E|}\right)\right) + 1\right)$
- Can we combine these terms into one log using $\log x + \log y = \log xy$?
- $\Sigma_{e \in E}(\log w(e) + 1) = (\log w(e_1) + \log 2) + + \cdots + \left(\log\left(w\left(e_{|E|}\right)\right) + \log 2\right)$
- $\Sigma_{e \in E}(\log w(e) + 1) = \log 2w(e_1)\, 2w(e_2)\, \ldots\, 2w\left(e_{|E|}\right) = \log \prod_{e \in E} 2w(e)$
- So how to compare $\log \prod_{e \in E} 2w(e)$ and $\log \sum_{e \in E} w(e)$?
  - **All** $w(e)$ are positive integers, so $\prod_{e \in E} 2w(e) \geq \sum_{e \in E} w(e)$
  - Since log is increasing on $\mathbb{Z}^+$, $\log \prod_{e \in E} 2w(e) \geq \log \sum_{e \in E} w(e)$

7

COMPARING $T(I)$ AND $Size(I)$

- We in fact show $T(I) \in O(Size(I))$

$O(\log \sum_{e \in E} w(e)) \subseteq^? O(|V| + \Sigma_{e \in E}(\log w(e) + 1) + \Sigma_{e \in E} \log|V|)$

How to compare $\log \sum_{e \in E} w(e)$ and $\Sigma_{e \in E}(\log w(e) + 1)$?

We just saw $\Sigma_{e \in E}(\log w(e) + 1) = \log \prod_{e \in E} 2w(e) \geq \log \sum_{e \in E} w(e)$

So $T(I) \in O(Size(I)^c)$ where $c = 1$

So this reduction has runtime that is polynomial in the input size!

8

TSP-Optimal Value $\leq_P^T$ TSP-Dec

So TSP-OptimalValue-Solver is polytime… But is it a **correct reduction from TSP-Optimal Value to TSP-Dec?**

Algorithm: *TSP-OptimalValue-Solver*(G, w)
external *TSP-Dec-Solver*
$hi \leftarrow \sum_{e \in E} w(e)$
$lo \leftarrow 0$
if not *TSP-Dec-Solver*(G, w, hi) **then return** (∞)
while $hi > lo$
do $\begin{cases} mid \leftarrow \lfloor \frac{hi+lo}{2} \rfloor \\ \textbf{if } \textit{TSP-Dec-Solver}(G, w, mid) \\ \quad \textbf{then } hi \leftarrow mid \\ \quad \textbf{else } lo \leftarrow mid + 1 \end{cases}$
return (hi)

Need to prove:
TSP-OptimalValue-Solver(G,w)
returns the weight $\boldsymbol{W}$
of the **shortest Hamiltonian Cycle (HC) in G**

Sketch: We return ∞ iff there is **no HC**.
Key loop invariant: $W \in [lo, hi]$.
So, at termination when $hi = lo$,
we return exactly $hi = W$.

9

TSP-Optimal Value $\leq_P^T$ TSP-Dec

Algorithm: *TSP-OptimalValue-Solver*(G, w)
external *TSP-Dec-Solver*
$hi \leftarrow \sum_{e \in E} w(e)$
$lo \leftarrow 0$
if not *TSP-Dec-Solver*(G, w, hi) **then return** (∞)
while $hi > lo$
do $\begin{cases} mid \leftarrow \lfloor \frac{hi+lo}{2} \rfloor \\ \textbf{if } \textit{TSP-Dec-Solver}(G, w, mid) \\ \quad \textbf{then } hi \leftarrow mid \\ \quad \textbf{else } lo \leftarrow mid + 1 \end{cases}$
return (hi)

So, TSP-OptimalValue-Solver is **polytime**, and is a **correct** reduction.

We have therefore shown:
TSP-Optimal Value is polytime reducible to TSP-Dec

So, if an $\boldsymbol{O(1)}$ implementation of TSP-Dec-Solver exists, then we have a **polytime** implementation of TSP-Optimal-Value-Solver!

In fact, TSP-OptimalValue-Solver remains **polytime** even if the implementation of the **oracle runs in polytime** instead of O(1)! (bonus slides)

10

PROVING REDUCTIONS CORRECT

- **In more complex reductions** where we **transform the input** before calling the oracle, we will need a **more complex proof**:
- (A) If there is a(n optimal) solution in the input, our transformation will preserve that solution so the oracle can find it, and
- (B) Our transformation doesn't introduce new solutions that are ***not*** present in the original input
  - *(i.e., if we find a solution in the transformed input, there was a corresponding solution in the original input)*

More on this later…

11

INPUT SIZE CHEAT SHEET

Input I	**Perfectly fine** choices of $Size(I)$
int x	1 or $\lfloor \log(x) \rfloor + 1$ **(can simplify to $\log(x) + 1$ or $\log x$)**
Graph (V, E) with weights W:	$\lvert V\rvert$ or $\lvert E\rvert$ or $\lvert V\rvert^2$ or $\lvert V\rvert + \lvert E\rvert$ or $\sum_{e \in E}(\log(w(e)) + 1)$ or $\sum_{u,v \in V}(\log(w(u,v)) + 1)$ or **any sum of terms above**
$A[1..n]$ of int	n or $\sum_i(\log(A[i]) + 1)$
n x n matrix m	n^2 or $\sum_{i,j}(\log(m_{ij}) + 1)$

To write down x=1, need log(1)+1=1 bit. For x=2 this is 2 bits. For x=4, 3 bits.

Pick any expression that makes your analysis easy

Exponentially larger than optimal representation!

Input I	**Examples of *BAD*** choices of $Size(I)$
int x	x
Graph (V, E)	$2^{\lvert V\rvert}$ or $\lvert V\rvert^{\lvert E\rvert}$ or $\sum_{e \in E} w(e)$
$A[1..n]$ of int	2^n or $\sum_i A[i]$

Pseudo-polynomial ~= no exponentiation of non-constant terms

Technically any **pseudo-polynomial combination** of these terms is fine.
For example, the following is fine:
$(|E|^{100} + |V|^2) \cdot \sum_{e \in E}(\log(w(e)) + 1)$

12

- So far we know
  - TSP-Dec $\le_p^T$ TSP-Optimal Value
  - TSP-Dec $\le_p^T$ TSP-Optimization
  - TSP-Optimal Value $\le_p^T$ TSP-Dec
- Let's show
  - TSP-Optimization $\le_p^T$ TSP-Dec

13

WHAT ABOUT REDUCING **TSP-OPTIMIZATION** TO TSP-DEC?

Problem 7.5

TSP-Optimization

Instance: *A graph G and edge weights $w : E \to \mathbb{Z}^+$.*

Find: *A hamiltonian cycle H in G such that $w(H) = \sum_{e \in H} w(e)$ is minimized.*

Need to return the **actual** minimum Hamiltonian Cycle!

Problem 7.7

TSP-Decision

Instance: *A graph G, edge weights $w : E \to \mathbb{Z}^+$, and a target T.*

Question: *Does there exist a hamiltonian cycle H in G with $w(H) \le T$?*

Given only a **single bit** of information **per call** to the oracle

We already know how to get the **weight T^*** of the minimum HC…

Idea: Use T^* along with calls to the oracle to somehow figure out **which edges** are involved in the minimum HC?

14

TSP-Optimization $\le_P^T$ TSP-Dec

Algorithm: *TSP-Optimization-Solver*$(G = (V, E), w)$
external *TSP-OptimalValue-Solver*, *TSP-Dec-Solver*
$T^* \leftarrow$ *TSP-OptimalValue-Solver*(G, w)
if $T^* = \infty$ **then return** ("no hamiltonian cycle exists")
$w_0 \leftarrow w$
$H \leftarrow \emptyset$
for all $e \in E$
do $\begin{cases} w_0[e] \leftarrow \infty \\ \textbf{if not } \textit{TSP-Dec-Solver}(G, w_0, T^*) \\ \quad \textbf{then } \begin{cases} w_0[e] \leftarrow w[e] \\ H \leftarrow H \cup \{e\} \end{cases} \end{cases}$
return (H)

To remove any dependence on this "other oracle," simply replace this call with the reduction **code** we showed

Already know this call is poly-time reducible to TSP-Dec!

If removing edge e removes **every** Hamiltonian cycle of minimum weight

then e is part of **every** minimum Hamiltonian cycle, and we add it to H (and add it back into the graph)

At the end, the graph contains precisely the edges that are needed to produce a minimum HC

[Correctness] **Loop invariant:** there exists a HC of weight T^* in w_0

By the end of the loop, H contains all finite edges in w_0

So some HC C of weight T^* **is contained in H**

15

At the end of the algorithm, there is a Hamiltonian Cycle C of optimal weight T^* **contained in** H

If **H is precisely C**, then we are done. **Suppose not** to obtain a contradiction.

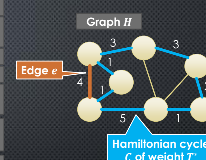
In this case, there are some **other edges** in H as well.

Let e be one such edge.

Consider the iteration when e was processed. Note e was **not removed** in this iteration!

Doing so would remove **all** Hamiltonian Cycles of weight T^*, **including C**.

This means the edge must be part of C---contradiction!



16

TSP-Optimization $\le_P^T$ TSP-Dec

Algorithm: *TSP-Optimization-Solver*$(G = (V, E), w)$
external *TSP-OptimalValue-Solver*, *TSP-Dec-Solver*
$T^* \leftarrow$ *TSP-OptimalValue-Solver*(G, w) — $poly(Size(I))$
if $T^* = \infty$ **then return** ("no hamiltonian cycle exists") — $O(1)$
$w_0 \leftarrow w$ — $O(m)$ to copy matrix
$H \leftarrow \emptyset$ — $O(1)$ to create list
for all $e \in E$ — $O(m)$ iterations
do $\begin{cases} w_0[e] \leftarrow \infty \\ \textbf{if not } \textit{TSP-Dec-Solver}(G, w_0, T^*) \\ \quad \textbf{then } \begin{cases} w_0[e] \leftarrow w[e] \\ H \leftarrow H \cup \{e\} \end{cases} \end{cases}$ — $O(1)$ per iteration
return (H)

So this is a **correct** reduction. Is it a **polytime reduction**?

What's the runtime?

Let's assume unit costs for simplicity

Runtime = $poly(Size(I)) + O(m)$

What's $Size(I)$? (What's a "useful" lower bound?)

$Size(I) = \Omega(|E|) = \Omega(m)$

Clearly $O(m) \in O(Size(I)^1)$
So runtime is in $poly(Size(I))$

So **yes**, this is a **polytime reduction**

What would change if we precisely counted the number of bits in each edge, weight, etc., in $Size(I)$?

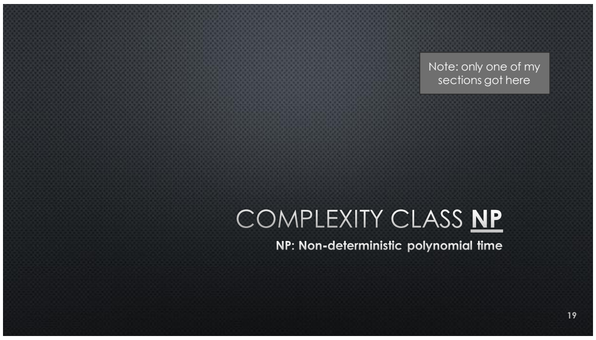
What if **operations on weight w** took **$O(\log w)$ time**? (bonus slides)

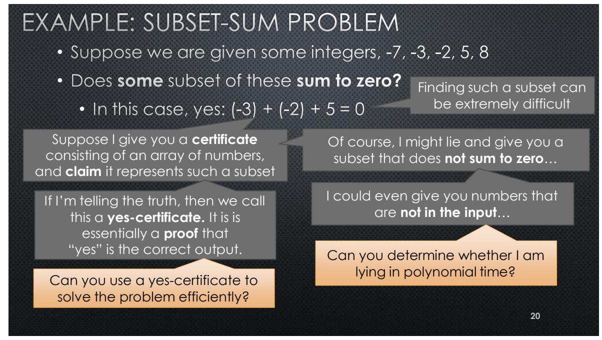
17

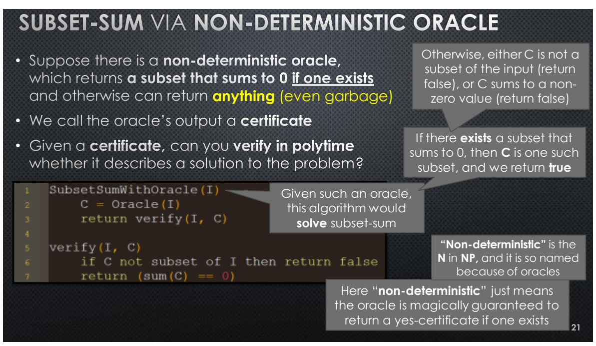
RECAP

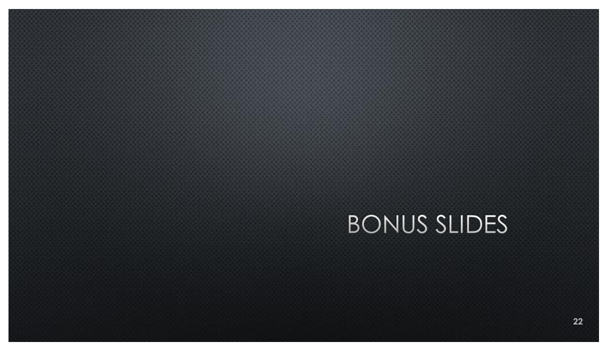
- Showed three flavours of TSP are **polytime-equivalent** (i.e., if you can solve one flavour in polytime, you can solve all three flavours in polytime)
  - One of these was a decision problem (yes/no), and the other two were not (total weight, actual cycle)
- **Decision and non-decision flavours** of a problem are often polytime-equivalent
- Proofs for a **polytime Turing reduction**
  - **Correctness** (return value is correct for every possible input)
  - **Polytime** (runtime is polynomial in the input size) [or poly(some lower bound on the input size)]

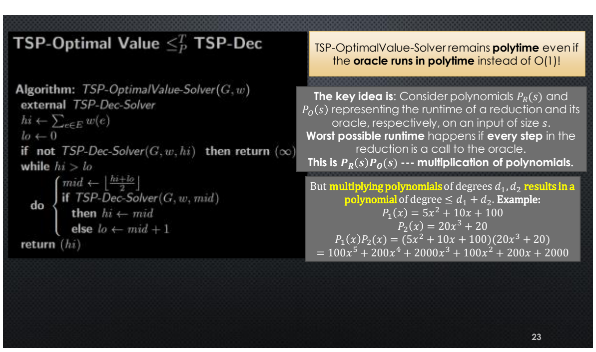
18

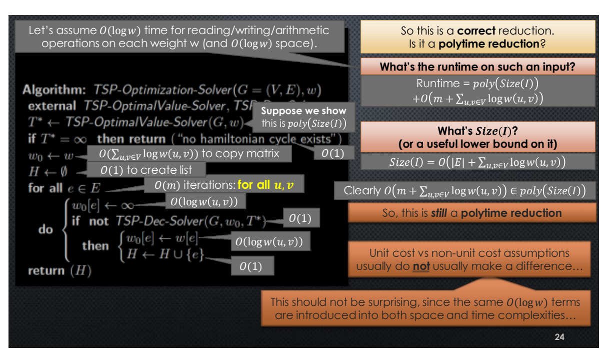

Note: only one of my sections got here
COMPLEXITY CLASS NP
NP: Non-deterministic polynomial time
19


EXAMPLE: SUBSET-SUM PROBLEM
• Suppose we are given some integers, -7, -3, -2, 5, 8
• Does some subset of these sum to zero?
• In this case, yes: (-3) + (-2) + 5 = 0
Finding such a subset can be extremely difficult
Suppose I give you a certificate consisting of an array of numbers, and claim it represents such a subset
Of course, I might lie and give you a subset that does not sum to zero…
If I'm telling the truth, then we call this a yes-certificate. It is is essentially a proof that "yes" is the correct output.
I could even give you numbers that are not in the input…
Can you determine whether I am lying in polynomial time?
Can you use a yes-certificate to solve the problem efficiently?
20


SUBSET-SUM VIA NON-DETERMINISTIC ORACLE
• Suppose there is a non-deterministic oracle, which returns a subset that sums to 0 if one exists and otherwise can return anything (even garbage)
• We call the oracle's output a certificate
• Given a certificate, can you verify in polytime whether it describes a solution to the problem?
SubsetSumWithOracle(I)
C = Oracle(I)
return verify(I, C)
verify(I, C)
if C not subset of I then return false
return (sum(C) == 0)
Given such an oracle, this algorithm would solve subset-sum
Otherwise, either C is not a subset of the input (return false), or C sums to a non-zero value (return false)
If there exists a subset that sums to 0, then C is one such subset, and we return true
"Non-deterministic" is the N in NP, and it is so named because of oracles
Here "non-deterministic" just means the oracle is magically guaranteed to return a yes-certificate if one exists
21


BONUS SLIDES
22


TSP-Optimal Value $\leq_P^T$ TSP-Dec
Algorithm: TSP-OptimalValue-Solver(G, w)
external TSP-Dec-Solver
$hi \leftarrow \sum_{e \in E} w(e)$
$lo \leftarrow 0$
if not TSP-Dec-Solver(G, w, hi) then return (∞)
while $hi > lo$
do $mid \leftarrow \lfloor \frac{hi+lo}{2} \rfloor$
if TSP-Dec-Solver(G, w, mid)
then $hi \leftarrow mid$
else $lo \leftarrow mid + 1$
return (hi)
TSP-OptimalValue-Solver remains polytime even if the oracle runs in polytime instead of O(1)!
The key idea is: Consider polynomials $P_R(s)$ and $P_O(s)$ representing the runtime of a reduction and its oracle, respectively, on an input of size s.
Worst possible runtime happens if every step in the reduction is a call to the oracle.
This is $P_R(s)P_O(s)$ --- multiplication of polynomials.
But multiplying polynomials of degrees d_1, d_2 results in a polynomial of degree $\leq d_1 + d_2$. Example:
$P_1(x) = 5x^2 + 10x + 100$
$P_2(x) = 20x^3 + 20$
$P_1(x)P_2(x) = (5x^2 + 10x + 100)(20x^3 + 20)$
$= 100x^5 + 200x^4 + 2000x^3 + 100x^2 + 200x + 2000$
23


Let's assume $O(\log w)$ time for reading/writing/arithmetic operations on each weight w (and $O(\log w)$ space).
Algorithm: TSP-Optimization-Solver$(G = (V, E), w)$
external TSP-OptimalValue-Solver, TSP
$T^* \leftarrow$ TSP-OptimalValue-Solver(G, w)
if $T^* = \infty$ then return ("no hamiltonian cycle exists")
$w_0 \leftarrow w$
$H \leftarrow \emptyset$
for all $e \in E$
do $w_0[e] \leftarrow \infty$
if not TSP-Dec-Solver(G, w_0, T^*)
then $w_0[e] \leftarrow w[e]$
$H \leftarrow H \cup \{e\}$
return (H)
Suppose we show this is $poly(Size(I))$
$O(1)$
$O(\sum_{u,v \in V} \log w(u, v))$ to copy matrix
$O(1)$ to create list
$O(m)$ iterations: for all u, v
$O(\log w(u, v))$
$O(1)$
$O(\log w(u, v))$
$O(1)$
So this is a correct reduction.
Is it a polytime reduction?
What's the runtime on such an input?
Runtime = $poly(Size(I))$
$+O(m + \sum_{u,v \in V} \log w(u, v))$
What's $Size(I)$?
(or a useful lower bound on it)
$Size(I) = O(|E| + \sum_{u,v \in V} \log w(u, v))$
Clearly $O(m + \sum_{u,v \in V} \log w(u, v)) \in poly(Size(I))$
So, this is still a polytime reduction
Unit cost vs non-unit cost assumptions usually do not usually make a difference…
This should not be surprising, since the same $O(\log w)$ terms are introduced into both space and time complexities…
24