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 $\textbf{NP}$
  • Oracles, certificates, polytime verification algorithms
RECALL

- So far we know
  - TSP-Dec $\leq_T^p$ TSP-Optimal Value
  - TSP-Dec $\leq_T^p$ TSP-Optimization
- In progress
  - TSP-Optimal Value $\leq_T^p$ TSP-Dec

### Travelling Salesperson Problems

<table>
<thead>
<tr>
<th>Problem 7.5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TSP-Optimization</strong></td>
</tr>
<tr>
<td><strong>Instance:</strong> A graph $G$ and edge weights $w : E \rightarrow \mathbb{Z}^+$.</td>
</tr>
<tr>
<td><strong>Find:</strong> A hamiltonian cycle $H$ in $G$ such that $w(H) = \sum_{e \in H} w(e)$ is minimized.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem 7.6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TSP-Optimal Value</strong></td>
</tr>
<tr>
<td><strong>Instance:</strong> A graph $G$ and edge weights $w : E \rightarrow \mathbb{Z}^+$.</td>
</tr>
<tr>
<td><strong>Find:</strong> The minimum $T$ such that there exists a hamiltonian cycle $H$ in $G$ with $w(H) = T$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem 7.7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TSP-Decision</strong></td>
</tr>
<tr>
<td><strong>Instance:</strong> A graph $G$, edge weights $w : E \rightarrow \mathbb{Z}^+$, and a target $T$.</td>
</tr>
<tr>
<td><strong>Question:</strong> Does there exist a hamiltonian cycle $H$ in $G$ with $w(H) \leq T$?</td>
</tr>
</tbody>
</table>
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.

Array of empty lists for all vertices $v$

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 $1..|V|$)

Let’s relate this to runtime... what’s the runtime?
TSP-Optimal Value $\leq \frac{T}{T'}$ TSP-Dec

Let's assume $O(1)$ time for operations on weights. Technically not needed to show polytime, but simplifies.

Algorithm: TSP-OptimalValue-Solver($G, w$)

\begin{align*}
\text{external} & \quad TSP-Dec-Solver \\
hi & \leftarrow \sum_{e \in E} w(e) \quad 0(|E|) \\
lo & \leftarrow 0 \quad O(1) \\
\text{if not} & \quad TSP-Dec-Solver(G, w, hi) \quad \text{then return} \quad (\infty) \\
\text{while} & \quad hi > lo \\
& \left\{ \\
& \begin{aligned}
& \text{mid} \leftarrow \left\lfloor \frac{hi + lo}{2} \right\rfloor \\
& \text{if} & \quad TSP-Dec-Solver(G, w, mid) \quad \text{then} \quad hi \leftarrow mid \\
& \text{else} & \quad lo \leftarrow mid + 1 \\
& \end{aligned}
\end{align*}

\text{return} \quad (hi)

# iterations: $O(\log(hi - lo))$

= $\log \sum_{e \in E} w(e)$

$O(1)$

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

$0(1)$ for the oracle
COMPARING $T(I)$ AND $\text{Size}(I)$

- $T(I) \in O(|E| + \log \sum_{e \in E} w(e))$
- $\text{Size}(I) = |V| + \sum_{e \in E} (\log w(e) + 1) + \log|V| + 1$
  - $= |V| + \sum_{e \in E} (\log w(e) + 1) + \sum_{e \in E} (\log|V| + 1)$
  - $= |V| + \sum_{e \in E} (\log w(e) + 1) + \sum_{e \in E} (\log|V|) + |E|$

- Want to show $T(I) \in O(\text{Size}(I)^c)$ for some constant $c$ (we show $c=1$)

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

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

**How to compare** $\log \sum_{e \in E} w(e)$ and $\sum_{e \in E} (\log w(e) + 1)$?
COMPARING $T(I)$ AND Size($I$)

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

• $\sum_{e \in E} (\log w(e) + 1) = (\log w(e_1) + 1) + (\log w(e_2) + 1) + \cdots + \left(\log \left( w(e_{|E|}) \right) + 1 \right)$

• Can we combine these terms into one log using $\log x + \log y = \log xy$?

• $\sum_{e \in E} (\log w(e) + 1) = (\log w(e_1) + \log 2) + + \cdots + \left(\log \left( w(e_{|E|}) \right) + \log 2 \right)$

• $\sum_{e \in E} (\log w(e) + 1) = \log 2w(e_1) 2w(e_2) \cdots 2w(e_{|E|}) = \log \prod_{e \in E} 2w(e)$

• 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 $\mathbb{Z}^+$, $\log \prod_{e \in E} 2w(e) \geq \log \sum_{e \in E} w(e)$
COMPARING $T(I)$ AND $\text{Size}(I)$

• We in fact show $T(I) \in O(\text{Size}(I))$

\[
O(\log \sum_{e \in E} w(e)) \subseteq \, O(|V| + \sum_{e \in E} (\log w(e) + 1) + \sum_{e \in E} \log |V|)
\]

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

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

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

So this reduction has runtime that is polynomial in the input size!
Need to prove:
TSP-OptimalValue-Solver(G,w)
returns the weight $W$
of the shortest Hamiltonian Cycle (HC) in $G$

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

So TSP-OptimalValue-Solver is polytime... But is it a correct reduction from TSP-Optimal Value to TSP-Dec?
We have therefore shown: 

**TSP-Optimal Value** is **polytime**, reducible to **TSP-Dec**

So, if an $O(1)$ 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)

---

**Algorithm: TSP-OptimalValue-Solver**

```plaintext
Algorithm: TSP-OptimalValue-Solver(G, w)

external TSP-Dec-Solver

hi ← $\sum_{e \in E} w(e)$

lo ← 0

if not TSP-Dec-Solver(G, w, hi) then return (\infty)

while hi > lo

    mid ← $\frac{hi + lo}{2}$

    do

        if TSP-Dec-Solver(G, w, mid)
            then hi ← mid
        else lo ← mid + 1

    return (hi)
```

So, TSP-OptimalValue-Solver is **polytime**, and is a **correct** reduction.
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...
### Input Size Cheat Sheet

<table>
<thead>
<tr>
<th>Input $I$</th>
<th>Perfectly fine choices of $\text{Size}(I)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>int $x$</td>
<td>1 or $\lceil \log(x) \rceil + 1$ (can simplify to $\log(x) + 1$ or $\log x$)</td>
</tr>
<tr>
<td>Graph $(V,E)$</td>
<td>$</td>
</tr>
<tr>
<td>$A[1..n]$ of int</td>
<td>$n$ or $\sum_i (\log(A[i]) + 1)$</td>
</tr>
<tr>
<td>$n \times n$ matrix $m$</td>
<td>$n^2$ or $\sum_{i,j} (\log(m_{ij}) + 1)$</td>
</tr>
</tbody>
</table>

**Examples of BAD choices of $\text{Size}(I)$**

<table>
<thead>
<tr>
<th>Input $I$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>int $x$</td>
<td>$x$</td>
</tr>
<tr>
<td>Graph $(V,E)$</td>
<td>$2^{</td>
</tr>
<tr>
<td>$A[1..n]$ of int</td>
<td>$2^n$ or $\sum_i A[i]$</td>
</tr>
</tbody>
</table>

**Exponentially larger than optimal representation!**

**Pick any expression that makes your analysis easy**

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

**Technically any pseudo-polynomial combination of these terms is fine.** For example, the following is fine: $(|E|^{100} + |V|^2) \cdot \sum_{e \in E} (\log(w(e)) + 1)$
• So far we know
  • TSP-Dec $\leq^T_P$ TSP-Optimal Value
  • TSP-Dec $\leq^T_P$ TSP-Optimization
  • TSP-Optimal Value $\leq^T_P$ TSP-Dec
• Let’s show
  • TSP-Optimization $\leq^T_P$ TSP-Dec
**WHAT ABOUT REDUCING TSP-OPTIMIZATION TO TSP-DEC?**

**Problem 7.5**

**TSP-Optimization**
- **Instance:** A graph $G$ and edge weights $w : E \rightarrow \mathbb{Z}^+$.  
- **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 $w : E \rightarrow \mathbb{Z}^+$, and a target $T$.  
- **Question:** Does there exist a Hamiltonian cycle $H$ in $G$ with $w(H) \leq T$?

*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?*
TSP-Optimization \( \leq_T P \) TSP-Dec

**Algorithm:** TSP-Optimization-Solver \((G = (V, E), w)\) external TSP-OptimalValue-Solver, TSP-Dec-Solver

\[ T^* \leftarrow \text{TSP-OptimalValue-Solver}(G, w) \]

if \( T^* = \infty \) then return (“no hamiltonian cycle exists”)

\[ w_0 \leftarrow w \]
\[ H \leftarrow \emptyset \]

for all \( e \in E \)

\[
\begin{cases}
  w_0[e] &\leftarrow \infty \\
  \text{if not} &\quad \text{TSP-Dec-Solver}(G, w_0, T^*) \\
  \text{then} &\quad \begin{cases}
    w_0[e] &\leftarrow w[e] \\
    H &\leftarrow H \cup \{e\}
  \end{cases}
\end{cases}
\]

return \((H)\)

**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 \( H \)

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.
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!
TSP-Optimization $\leq_T^P$ TSP-Dec

**Algorithm:** TSP-Optimization-Solver($G = (V, E), w$) returns $T^*$
- 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 $\{ w_0[e] \leftarrow \infty$
    - if not TSP-Dec-Solver($G, w_0, T^*$)
      - then $\{ w_0[e] \leftarrow w[e]$
        - $H \leftarrow H \cup \{e\}$
  - end do
- return $(H)$

- $O(m)$ to copy matrix
- $O(1)$ to create list
- $0(m)$ iterations
- $0(1)$ per iteration
- $\text{poly}(\text{Size}(I))$
- $O(1)$ to copy matrix

**Runtime:**
- $\text{poly}(|E|) = \Omega(m)$
- Clearly $O(m) \in O(\text{Size}(I))$
- So runtime is in $\text{poly}(\text{Size}(I))$
- So yes, this is a polytime reduction

**What’s the runtime?**
- Let’s assume unit costs for simplicity
- Runtime = $\text{poly}(\text{Size}(I)) + O(m)$

**What’s Size($I$)?**
- (What’s a “useful” lower bound?)
- $\text{Size}(I) = \Omega(|E|) = \Omega(m)$

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)
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)]
COMPLEXITY CLASS NP

NP: Non-deterministic polynomial time

Note: only one of my sections got here
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

Suppose I give you a certificate consisting of an array of numbers, and claim it represents such a subset.

If I’m telling the truth, then we call this a yes-certificate. It is essentially a proof that “yes” is the correct output.

Of course, I might lie and give you a subset that does not sum to zero...

I could even give you numbers that are not in the input...

Finding such a subset can be extremely difficult.

Can you determine whether I am lying in polynomial time?

Can you use a yes-certificate to solve the problem efficiently?
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?

Given such an oracle, this algorithm would solve subset-sum:

```python
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)
```

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.
BONUS SLIDES
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:

\[ P_1(x) = 5x^2 + 10x + 100 \]
\[ P_2(x) = 20x^3 + 20 \]
\[ P_1(x)P_2(x) = (5x^2 + 10x + 100)(20x^3 + 20) = 100x^5 + 200x^4 + 2000x^3 + 100x^2 + 200x + 2000 \]
Let’s assume $O(\log w)$ time for reading/writing/arithmetic operations on each weight $w$ (and $O(\log w)$ space).

Algorithm:  

$\text{TSP-Optimization-Solver}(G = (V, E), w)$

external $\text{TSP-OptimalValue-Solver}, \text{TSP-Dec-Solver}$

$T^* \leftarrow \text{TSP-OptimalValue-Solver}(G, w)$

if $T^* = \infty$ then return (“no hamiltonian cycle exists”)

$w_0 \leftarrow w$

$H \leftarrow \emptyset$

for all $e \in E$

$\begin{array}{l}
\text{do } \\
\{ w_0[e] \leftarrow \infty \}
\end{array}$

if not $\text{TSP-Dec-Solver}(G, w_0, T^*)$ then

$\begin{array}{l}
\{ w_0[e] \leftarrow w[e] \}
\end{array}$

$H \leftarrow H \cup \{ e \}$

return $(H)$

Suppose we show this is $poly(\text{Size}(I))$

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

What’s the runtime on such an input?

Runtime $= \text{poly}(\text{Size}(I))$

$+ O(m + \sum_{u,v \in V} \log w(u, v))$

What’s $\text{Size}(I)$?

(or a useful lower bound on it)

$\text{Size}(I) = O(|E| + \sum_{u,v \in V} \log w(u, v))$

Clearly $O(m + \sum_{u,v \in V} \log w(u, v)) \in \text{poly}(\text{Size}(I))$

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