Advanced Algorithms and Data Structures
Assignment 4

Isabella Odorico	Caroline Kierkegaard	Mikkel Willén	Fie Hammer
jkd427	qlj556	bmq419	gsr530

January 3, 2024

NP-Completeness

34.4-6

We have a polynomial-time algorithm to decide formula satisfiability and we let ϕ be a boolean formula with n variables $\{x_1, x_2, ..., x_n\}$. We can use this algorithm to find a satisfying assignment in polynomial time by following this pseudocode:

```
Use the algorithm on phi to decide if it is satisfiable
if NO
    Return "No satisfying assignment can be found"
else
    Initialise an array A of zeros
    for i = 1 to n
        x_i = 0
        Apply the algorithm to check satisfiability of phi where x_i = 0
        if YES
            A[i] = 0
        else
            A[i] = 1
```

So our given algorithm runs in polynomial time, and it is called n times. Hence we have polynomial runtime: $O(n \cdot n^k) = O(n^k)$.

34.5-1

To show that SUBGRAPH-ISOMORPHISM is an NP-complete problem of two undirected graphs G_1 and G_2, we first show it is in NP.

We verify G_1 and G_2 are isomorphic graphs, which can be done in polynomial time, as we have to check that the isomorphism is an edge preserving bijection.

Next we show that subgraph isomorphism is NP-hard by showing

CLIQUE $\leq_P$ SUBGRAPH-ISOMORPHISM.

If we let G_1 be a clique with $k = |V_1|$ and $G_1 = G_2 = (V_2, E_2)$. If $k > |V_2|$ then G_1 is not a subgraph. If $K \leq |V_2|$ then G_2 will contain a clique of size k if and only if G_1 is a subgraph of G_2. Since G_1 is isomorphic to itself, then it will be a subgraph isomorphism of G_2, showing we can reduce CLIQUE to SUBGRAPH-ISOMORPHISM in polynomial time.

34.5-6

To show that the Hamiltonian-path problem is NP-complete, we have to show that is is both in NP and NP-hard. We have already shown it to be in NP in assignment 3 34.2-6. Here is a repeat of our answer:

"If we know there is a path from u to v in the graph G, we can check all paths from u to another vertex, and then from that vertex and to yet another vertex. If we continue doing this, we get a running time of $O(n!)$ where n is the number of vertices in the graph G. This follows, since when we have a vertex u, we have $n - 1$ vertices left, when we then move on we have $n - 2$ vertices left and so on. Thus it can be verified by a polynomial-time algorithm and therefore belongs in NP."

Then we have to show that it is also NP-hard. We do this by making the reduction HAMILTONIAN-CYCLE $\leq_P$ HAMILTONIAN-PATH. We do this by creating a new edge between two vertices u and v. Then we check if this is a HAMILTONIAN-CYCLE. If it is, there must be a HAMILTONIAN-PATH starting in u and ending in v. If not, we choose a new combination of vertices not yet checked, and repeat. We can at most create $|V|! = V^2 - V = O(V^2)$ new edges, meaning we can reduce HAMILTONIAN-PATH to HAMILTONIAN-CYCLE in polynomial time.

34.5-7

We formulate a related decision problem:

Given a graph and an integer k, does there exist a simple cycle of maximum length k in the graph?

The decision problem is NP-complete if it is both in NP and NP-hard.

It is in NP, because we can verify if the answer is correct in polynomial time. This is in part due to how there are no repeated vertices, as per the definition given of the longest-simple-cycle problem, and hence there is only a need for transversal of the cycle once.

We show it is NP-hard by making the reduction LONGEST-SIMPLE-CYCLE $\leq_P$ HAMILTONIAN-CYCLE. We start by checking if there is a HAMILTONIAN-CYCLE. If there is we return yes, if not we begin to remove vertices from G. We start by trying all combination of removing 1 vertex, and check if there is a HAMILTONIAN-CYCLE. Then we check all combination of remove 2 vertices, and so on. We return yes, as soon as there is a HAMILTONIAN-CYCLE. We will at most run this algorithm $|V|!$ times, since that is all the combination of vertices we can remove, thus showing the reduction from HAMILTONIAN-CYCLE to LONGEST-SIMPLE-CYCLE.

Exponential algorithms and parameterized complexity

Exercise 1

Exercise 2

a.

We have the expression from the inductive step:

$$T(n) \leq 1 + s\left(2 \times 3^{(n-s)/3} - 1\right) = \left(\frac{s}{3^{s/3}}\right) \times 2 \times 3^{n/3} + 1 - s$$

Now in order to complete the inductive step for $n \geq 2$ and $s = 1$, we first simplify the expression but setting s to equal 1:

$$T(n) \leq 1 + 1\left(2 \times 3^{(n-1)/3} - 1\right) = \left(\frac{1}{3^{1/3}}\right) \times 2 \times 3^{n/3} + 1 - 1$$

We remove part of the equation in order to simplify the equation:

$$T(n) \leq= \left(\frac{1}{3^{1/3}}\right) \times 2 \times 3^{n/3} + 1 - 1$$

And simplify:

$$T(n) \leq \left(\frac{2}{3^{1/3}}\right) \times 3^{n/3}$$

And simplify further:

$$T(n) \leq 2 \times 3^{(n-1)/3}$$

$$2 \times 3^{(n-1)/3} \leq 2 \times 3^{n/3} - 1$$

As we know, our n is defined as $n \geq 2$ which is exactly what the above expressions require to be true. (technically it just requires that $n \geq 1, 5$)
Now with this in mind, we have proven that our induction step remains true for the bound we were looking for: $\left(\frac{s}{3^{s/3}}\right) \times 2 \times 3^{n/3} + 1 - s \leq 2 \times 3^{n/3} - 1$ for $n \geq 2$ and $s = 1$

b.

In order to complete the inductive step for $n \geq 2$ and $s > 1$, we start with the given expression from the inductive step:

$$T(n) \leq 1 + s\left(2 \times 3^{(n-s)/3} - 1\right) = \left(\frac{s}{3^{s/3}}\right) \times 2 \times 3^{n/3} + 1 - s$$

Now, using the provided lemma (3), which states that: $f(s) = 3^{s/3} - s \geq 0$.
This we can substitute this into our expression:

$$\left(\frac{s}{3^{s/3}}\right) \leq 1$$

Which gives us:

$$T(n) \leq 1 + s\left(2 \times 3^{(n-s)/3} - 1\right) \leq 1 + 2 \times 3^{(n-s)/3} - s$$

As we found in question a, we can replace $2 \times 3^{(n-s)/3} - s$ with $2 \times 3^{(n-1)/3}$ and we get:

$$T(n) \leq 1 + 2 \times 3^{(n-s)/3} - s \leq 2 \times 3^{(n-1)/3} - (s-1)$$

As s in this case always will be greater than 1: $s > 1$, we can conclude that: $(s-1)$ is negative. This always us to the last term in the inequality, and hence we can simplify to:

$$T(n) \leq 2 \times 3^{(n-s)/3}$$

We need to find the bound for this and can set it up by $2 \times 3^{n/3} - 1$. We can set up the following inequality:

$$2 \times 3^{(n-s)/3} \leq 2 \times 3^{n/3} - 1$$

We can simplify this by Dividing both sides by $2 \times 3^{n/3}$:

$$\frac{1}{3^{s/3}} \leq \frac{1}{2} - \frac{1}{2 \times 3^{n/3}}$$

Since we have that $s > 1$, we can conclude that $\frac{1}{3^{s/3}}$ is less than or equal to $\frac{1}{2}$, and the right-hand side is positive.
This means that the inequality holds true.

Exercise 3

Exercise 4

Dispositions

Disposition of NP Completeness Isabella (jkd427)

* I have used the same disposition for NP-completeness as in assignment 3:

- Introduction to NP completeness
  - Language
  - Encoding
  - Decision Problems
  - Polynomial time (example)
- Different classes
  - P vs NP
  - Co-NP
- Explain verification algorithms
- NP-completeness
  - Reducibility
  - NP-hard
  - CIRCUIT-SAT example and proof (if time)

Disposition of Exact exponential algorithms and parameterized complexity Isabella (jkd427)

- Introduction to Exact Exponential Algorithms
  - What are they?
  - Parameterized Complexity
  - Why are EEA and PC interesting?
- EEA example
  - Travelling Salesman Problem (explain and prove)
- PC example
  - Vertex Cover Problem (bar fight prevention) (explain)
  - Bounded search tree (if time)
- FPT vs XP
  - k-clique (if time)

NP Completeness (Caroline)

- Define language, encoding and decision problems
- Explain P vs NP (and co-NP)
- Explain verification algorithms
- Explain reducibility and NP-hard (NP-complete).
- Show CIRCUIT-SAT to be NP-complete.

Exact exponential algorithms and parameterized complexity (Caroline)

- Exact exponential algorithms
-
  - Exact TSP via Dynamic Programming
  - Dynamic Programming
- Parameterized problems
-
  - k-Vertex Cover
  - Kernelization
  - Bounded search tree
- FPT vs XP

NP-Completeness (Mikkel)

- Introduction
  - P and NP
  - co-NP
  - Language and encodings
- Verification algorithms
- Reducibility and NP-hard (NP-complete)
- Show Circuit-sat to be NP-complete

Exact exponential algorithms and parameterized complexity (Mikkel)

- Introduction
- Exact exponential algorithm
  - What is it?
  - Example with Travelling Salesman Problem
- Parameterized complexity
  - What is it?
  - Example with Vertex-Cover Problem

NP Completeness (Fie gsr530)

- What is N completeness
  - Language, Encoding and Decision Problems
- P vs NP
- Co-NP
- Verification algorithms
- Reducibility and NP-Hard
- CIRCUIT-SAT - NP-Complete

Exact exponential algorithms and parameterized complexity (Fie gsr530)

- Exact exponential algorithms
  - Explain it
- Parameterized problems
  - Explain it
- FPT vs EXP
  - Explain it