NP Completeness Examples

CPSC 413 - Algorithm Design

Benyamin Bashari

NP

- In order to prove that the decision problem A is NP, we need to prove that there is a polynomial time verification algorithm V(input x, certificate y).
- Basically certificate (y) is an answer to the decision problem A and the verification algorithm can check the answer and determine if it is correct or not.

NP

- How to prove a decision problem A is NP?
  - State the input to the verification algorithm (input to the problem and a certificate).
  - Show that certificate size is polynomial in size to the remaining input.
  - Give the verification algorithm.
  - Prove that the verification algorithm is correct.
  - Show that the verification algorithm runs in polynomial time.

3-Coloring Problem

- Precondition:
  - Graph $G = (V, E)$
  - $|V| = n$
- Postcondition:
  - Coloring $C = (c_1, c_2, \ldots, c_n)$, where c_i is the color of the i^{th} node in the graph and $c_i \in \{red, green, blue\}$
  - No two adjacent node have the same color

Input of the Verification Algorithm

- Input:
  - Graph G = (V, E)
  - $C = (c_1, c_2, \ldots, c_n)$ (certificate)
- Certificate size:
  - Assume that we show that color is {blue, green, red} with 2 bits then certificate size is (2n)
  - Since n is the number of vertices in the graph then 2n is polynomial in size of G.

Verification Algorithm

- Verification Algorithm:

```
{"yes", "no"} V(G = (V, E), C[n]) {
   for a in V:
       if C[a] is not in {"red", "blue", "green"}
           return "no"
   for each edge (a, b) in E:
       if c[a] == c[b] //there is two adjacent nodes with the same color
           return "no"
   return "yes"
}
```

Runtime of the Verification Algorithm

```
{"yes", "no"} V(G = (V, E), C[n]) {
   for a in V:
       if C[a] is not in {"red", "blue", "green"}
           return "no"
   for each edge (a, b) in E:
       if c[a] == c[b] //there is two adjacent nodes with the same color
           return "no"
   return "yes"
}
```

- First for iterates over all the nodes and the second for iterates over all the edges so the runtime is O(|V| + |E|)
- This is obviously polynomial in the size of the input

Correctness of the Verification Algorithm

```
{"yes", "no"} V(G = (V, E), C[n]) {
   for a in V:
       if C[a] is not in {"red", "blue", "green"}
           return "no"
   for each edge (a, b) in E:
       if c[a] == c[b] //there is two adjacent nodes with the same color
           return "no"
   return "yes"
}
```

- If G = (V, E) is a “yes” instance of 3-Coloring, then there is a coloring C, such that no two adjacent nodes have the same color then V(G, C) returns “yes”.
- If G = (V, E) is a “no” instance then there is no coloring C, so every V(G, C) returns “no”.

NP Completeness Proof

- Assume that we want to prove that the decision problem A is NP Complete
  - We need to prove that A is NP (which we covered)
  - Then we need to find another NP Complete problem B and prove that $B \leq_P A$
    - Give an algorithm to transform the input to B to an input to A.
    - Prove that the transformation algorithm runs in polynomial time.
    - Let s be an input to B and s' the transformed input to A. Prove that s is a "yes" instance of B if and only if s' is a "yes" instance of A.

NP Completeness Proof

- Let s be an input to B and s' the transformed input to A. Prove that s is a "yes" instance of B if and only if s' is a "yes" instance of A.
  - If s is a "yes" instance of B then s' is a "yes" instance of A
  - If s' is a "yes" instance of A then s is a "yes" instance of B

Clique Cover Problem

- Precondition:
  - Graph $G = (V, E)$
  - $V = \{1, 2, \ldots, n\}$
  - K
- Postcondition:
  - $V_1, V_2, \ldots, V_K$
    - $V_1 \cup V_2 \cup \ldots \cup V_n = \{1, \ldots, n\}$
    - $V_i \cap V_j = \emptyset$ for $i \neq j$
  - Each of V_i is a clique in G.

3-Coloring $\leq_P$ Clique Cover

- So we have a solver for clique cover problem.
- If we want to solve 3-Coloring problem we need to find three sets of vertices (V_1, V_2, V_3) such that
  - $V_1 \cup V_2 \cup V_3 = \{1, \ldots, n\}$
  - $V_i \cap V_j = \emptyset$ for $i \neq j$
  - There is no edge between any vertices in V_i
  - Then we can color V_1 as "blue", V_2 as "red", and V_3 as "green".
- But Clique Cover with $K = 3$ can find 3 sets where each of them are cliques in graph.

3-Coloring $\leq_P$ Clique Cover

- Complement of a graph G =(V ,E) is shown with $\bar{G}$ and it is a graph with vertices V, and there is an edge between two node (a, b) if and only if there is not an edge between a and b in G.
- Now if we give $\bar{G}$ and 3 to clique cover it will find 3 sets of vertices in $\bar{G}$ that they form 3 cliques.
- Now if we consider those three sets in G, they are 3 sets of vertices where there is no edge between the vertices of one set.

3-Coloring $\leq_P$ Clique Cover (Transforming Inputs)

- Inputs of 3-Coloring:
  - G = (V, E)
- Inputs of Clique Cover:
  - $\bar{G}$
  - K = 3

```
transformToCliqueCover(G = (V, E)) {  //inputs of 3-Coloring
   GBar = (V, E'={})
   for each two nodes a and b:
       if (a, b) is not an edge in E
           add (a, b) to E'
   return (GBar, 3) //K = 3
}
```

3-Coloring $\leq_P$ Clique Cover (Transforming Inputs)

```
transformToCliqueCover(G = (V, E)) {  //inputs of 3-Coloring
   GBar = (V, E'={})
   for each two nodes a and b:
       if (a, b) is not an edge in E
           add (a, b) to E'
   return (GBar, 3) //K = 3
}
```

- Runtime:
  - Initializing GBar takes O(|V|) and then checking for each two nodes takes O(|V|^2 |E|)
  - This runtime is polynomial in the input size.

3-Coloring $\le_P$ Clique Cover (Correctness)

```
transformToCliqueCover(G = (V, E)) {  //inputs of 3-Coloring
   GBar = (V, E'={})
   for each two nodes a and b:
       if (a, b) is not an edge in E
           add (a, b) to E'
   return (GBar, 3) //K = 3
}
```

- Let s be an input to 3-Coloring and s’ the transformed input to Clique Cover.
- We want to prove that if s is a “yes” instance of 3-Coloring then s’ is also a “yes” instance of Clique Cover.
- If s is a “yes” instance of 3-Coloring there there are 3 sets of nodes (V_1, V_2, V_3) that there is no edge between the nodes of one set.
- Then in GBar V_1, V_2, V_3 are cliques.
- Then CliqueCover(GBar, 3) returns “yes”

3-Coloring $\leq_P$ Clique Cover (Correctness)

```
transformToCliqueCover(G = (V, E)) {  //inputs of 3-Coloring
   GBar = (V, E'={})
   for each two nodes a and b:
       if (a, b) is not an edge in E
           add (a, b) to E'
   return (GBar, 3) //K = 3
}
```

- Let s be an input to 3-Coloring and s' the transformed input to Clique Cover.
- We want to prove that if s' is a "yes" instance of Clique Cover then s is also a "yes" instance of 3-Coloring.
- If s' is a "yes" instance of Clique Cover there are 3 sets of nodes (V_1, V_2, V_3) in GBar that each of them are cliques.
- Then in G there is no edge between the nodes of one of the sets V_i
- Then we can color V_1 in "blue", V_2 in "red", and V_3 in "green".
- Then s ia "yes" instance of 3-Coloring .