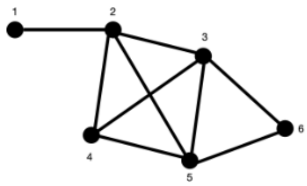
CSCI-570 Homework 7
Due Date: Tuesday, December 2nd, 11:59pm

Name: Sakshi Kuldeep Dhariwal

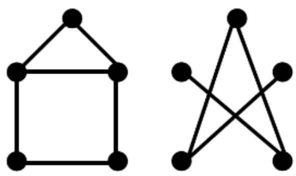
USC ID: 1851490301

Problem 1.
A *clique* in a graph $G = (V, E)$ is a subset of the vertices $C \subseteq V$, where every two vertices are connected by and edge, i.e. for every $u, v \in C$, $(u, v) \in E$. The following decision question can be asked: Given a graph G and an integer k, does G have a clique of size at least k?



For example, in the graph above vertices $\{2, 4, 5\}$ and $\{3, 5, 6\}$ form cliques of size 3 (and there are more of the same size), while the largest clique is of size 4 consisting of $\{2, 3, 4, 5\}$.

The *complement* of a graph $G = (V, E)$ is another graph $\overline{G} = (V, E')$ with the same set of vertices, but in $\overline{G}$ exactly those $u, v \in V$ vertices are connected by a $(u, v) \in E'$ edge, which are not connected in G, that is, $(u, v) \notin E$. (Thus $E' = \{V \times V \setminus E\}$.)



G and its complement $\overline{G}$

Prove the following statements:

(a) Let $G = (V, E)$ be a graph and $\overline{G}$ its complement. A subset $S \subseteq V$ of the vertices is an independent set in G if and only if it forms a clique in $\overline{G}$.

(b) The decision problem *clique* as formulated above is an $\mathcal{NP}$-*complete* problem.

[4+8 points]

Assignment - 7

Q.1a) Statement

Let $G = (V, E)$ be a graph and $\bar{G} = (V, \bar{E})$ its complement (so $(u, v) \in \bar{E}$ iff $(u, v) \notin E$).

To Prove: that a subset $S \subseteq V$ is an independent set in G if and only if S is a clique in $\bar{G}$

Independent set in G: $S \subseteq V$ is independent iff for every pair of distinct vertices $u, v \in S$ $(u, v) \notin E$.

Clique in a graph H: $S \subseteq V(H)$ is a clique iff for every pair of distinct vertices $u, v \in S$, the edge (u, v) is present in H.

Proof:

1) If S is independent in G then S is a clique in $\bar{G}$.

Suppose is independent in G. Then for every distinct $u, v \in S$ $(u, v) \notin E$. By the definition of complement $(u, v) \in \bar{E}$. Thus every pair of vertices in S is adjacent in $\bar{G}$, so S is a clique in $\bar{G}$

2) If S is a clique in $\bar{G}$ then S is independent in G.

Suppose S is a clique in $\bar{G}$. Then for every distinct $u, v \in S$ $(u, v) \in \bar{E}$. By definition of complement, $(u, v) \notin E$. Hence no 2 vertices of S are adjacent in G, so S is independent in G.

Since both directions hold, the equivalence is proved.

Computing the complement graph $\bar{G}$ is a polynomial-time operation.

A graph on n vertices has $\binom{n}{2} = O(n^2)$ possible undirected edges.

To form $\bar{G}$:

For each unordered pair (u, v), check:

If $(u, v) \in E$, then $(u, v) \notin \bar{E}$

else add (u, v) to $\bar{E}$

This is exactly 1 check per pair

Time complexity:

- Total no. of vertex pairs $= O(n^2)$
- Each check takes $O(1)$ if adjacency matrix is used.

(if adjacency list is used, check is still polynomial: $O(\deg u) \leq n$)

So total time $= O(n^2)$ (which is polynomial of in size of the polynomial n^2)

1) b) Given a graph $G = (V, E)$ and integer k,
To decide: if G contains a clique of size at least k.

We need to show:

1. Clique $\in NP$ (certificate + Polynomial time verifier)
2. Clique is NP-hard (we provide a polynomial time reduction from a known NP-complete problem)

1. Clique $\in$ NP

Certificate: A set $S \subseteq V$ of vertices with $|S| \geq k$ (at least k) claimed to form a clique.

Polynomial-time verifier:
Given (G, k) and certificate S:

- Check $|S| \geq k$ (Time: $O(|S|)$)
- For each unordered pair $u, v \in S$ (there are $\binom{|S|}{2}$ pairs) check whether $(u, v) \in E$. Each check is $O(1)$ with an adjacency matrix; with ~~most~~ adjacency lists use hashing or sorted lists to keep verification polynomial.
- If all pairs are edges, accept; otherwise reject.

Total verification time is $O(k^2)$, polynomial in input size. Thus Clique is $\in$ NP.

2) Clique is NP-Hard

We reduce from the known NP-complete problem Independent-Set.

Independent-Set: given a graph G and integer k, decide whether G has an independent set of size at least k. This problem is NP-complete.

Reduction (polynomial time): On input (G, k) for Independent-set, compute the complement graph $\bar{G}$ (same vertices; for each unordered pair $\{u, v\}$ include the edge in $\bar{G}$ iff it is not in G). Output $(\bar{G}, k)$ as the clique instance. Building $\bar{G}$ is polynomial-time: for each of the $\binom{|V|}{2}$ pairs decide membership in G and add the complement edge accordingly.

Correctness:

$(G, k) \in$ Independent Set $\iff (\bar{G}, k) \in$ Clique.

Forward direction

- Suppose G has an independent set S of size k
- By definition, no 2 vertices in S are adjacent in G.
- In $\bar{G}$ all pairs in S are adjacent $\to S$ forms a clique of size k.

Backward direction:

- Suppose $\bar{G}$ has a clique S of size k
- By definition all pairs in S are adjacent in $\bar{G} \to$ in G, no two vertices in S are adjacent $\to S$ is an independent set of size k in G

This proves reduction is correct & sensible

Correctness: From problem 1 (a), G has an independent set of size k iff $\bar{G}$ has a clique of size k. Hence $(\bar{G}, k)$ is a yes-instance of clique $\therefore$ the reduction is correct.

Because independent-set is NP-hard, Clique is NP-Hard. Combined with membership in NP, Clique is NP-complete.

Problem 2.

The $Half$-$Clique$ decision problem is as follows: given a graph G with $|V| = n$ vertices, does it contain a *clique* with at least $\frac{n}{2}$ nodes? Show that $Half$-$Clique$ is $\mathcal{NP}$-*complete* using the $\mathcal{NP}$-*completeness* of *Clique*.

[8 points]

Q.2 Given : A graph $G = (V, E)$ with $n = |V|$, decide whether G contains a clique of size at least $n/2$ (i.e at least half the vertices).

We show

- Half-Clique $\in$ NP
- Half-Clique is NP-hard (by a polynomial-time reduction from clique).

1. Half-Clique $\in$ NP.

Certificate : A subset $S \subseteq V$ of vertices with $|S| \geq n/2$ claimed to be a clique.

Verifier (polynomial time) :

- Check $|S| \geq n/2$ (Time : $O(|S|)$)
- For each unordered pair $u, v \in S$ check that $(u, v) \in E$. There are at most $\binom{n}{2}$ such checks ; each check is polynomial
- If all pairs are edges, accept ; otherwise reject.

Thus Half-Clique $\in$ NP.

2. Half-Clique is NP-hard (Reduction from Clique)

We give a polynomial-time reduction from clique (which is NP-complete) to Half-Clique. Let the input to clique be (G, k) with $n = |V(G)|$. We produce a graph G' such th

* G has a clique of size $\geq k \iff G'$ has a clique of size $n/2$

We must handle two cases depending on how k compares to $n/2$

Case A $k \geq n/2$.

Set $s = 2k - n$ (note $s \geq 0$). Construct G' by taking G and adding s isolated vertices (no edges between them & no edges connecting them to original vertices).

- Then $|V(G')| = n + s = 2k$, so $|V(G')|/2 = k$
- Any clique in G of size $\geq k$ remains a

clique in G'.

- Conversely, any clique of size $\geq |V(G')|/2 = R$ in G' cannot contain an isolated vertex (an isolated vertex cannot be adjacent to others), so that clique lies entirely inside the original vertex set and thus is a clique in G of size $\geq R$.

Therefore G has a clique of size $\geq R$ iff G' has a clique of size $\geq |V(G')|/2$.
Construction is polynomial time (adding S vertices).

Here we are creating a new graph G':
- Start with all vertices of G
- Add extra isolated vertices (or universally connected vertices)
- The no of added vertices is chosen so that: $\frac{N}{2} = R \Rightarrow N = 2R$

Sometimes, $N = 2|V(G)| - 2R$.
We add exactly S extra vertices so that the target clique size $N/2$ matches the "R" from Clique Problem.

Adding a known no of S vertices requires
- Creating a new list of vertices $O(n+S)$
- Possibly connecting them with edges: $\leq O((n+S)^2)$

Since:
- S is at most linear in n ($eg: S = 2R - n$ and $R \leq n$)

we have:
$$n + S = O(n)$$
& constructing all edges takes:
$$O((n+S)^2) = O(n^2)$$
which is polynomial in the size of the input graph.

Case B: $R < n/2$

Let $t = n - 2R$ (so $t > 0$). Construct G' as follows:
- Start with G (the original n vertices).
- Add t new vertices $u_1, \ldots, u_t$
- Add all edges among the new vertices (so the new vertices form a clique of size t)
- Connect each new vertex u_i to every original vertex (i.e. each u_i is adjacent to all

vertices of G)

Thus the t new vertices are universal to the original vertices & form a complete subgraph among themselves

Compute $|V(G')| = n + t = n + (n - 2k) = 2n - 2k$

$\therefore \frac{|V(G')|}{2} = n - k$.

Now showing equivalence:

- If G has a clique C with $|C| \geq k$, then in G', the set $C' = C \cup \{u_1, \ldots, u_t\}$ is a clique because every u_i is connected to every vertex in C & to every other u_j. Its size is $|C'| = |C| + t \geq k + t = k + (n - 2k) = n - k = |V(G')|/2$. So G' has a clique of size at least $|V(G')|/2$
- Conversely, suppose G' has a clique S with $|S| \geq |V(G')|/2 = n - k$. Let x be the no. of original vertices (from G) contained in S, & let y be the no. of new vertices contained in S. Then $x \leq n$ and $y \leq t$ and $|S| = x + y \geq n - k$.

Since the new vertices form a clique & are universal to original vertices, every such S is formed by selecting some original vertices plus some new vertices. But note that if the original graph G had no clique of size $\geq k$, then the maximum possible x is at most $k-1$. Thus,
$|S| = x + y \leq (k-1) + t = (k-1) + (n - 2k) = n - k - 1 < n - k$,
contradicting $|S| \geq n - k$. Hence G must have a clique of size at least k.
So again G has a clique of size $\geq k$ iff G' has a clique of size $\geq |V(G')|/2$

In case k, we add t new vertices & $O(nt + t^2)$ edges (connect new vertices to all original vertices & among themselves): still polynomial time as it is $O(t^2)$ which is a polynomial.

$\therefore$ Our mapping $(G, k) \to G'$ is a polynomial time reduction from Clique to Half clique

Half-clique $\in$ NP & is NP-hard (via above reduction), $\therefore$ Half-clique is NP-complete.

Problem 3.

Consider an instance of the *Satisfiability Problem*, specified by clauses $C_1, ..., C_k$ over a set of Boolean variables $x_1, ..., x_n$. We say that the instance is *monotone* if each clause contains only non-negated variables; that is, each term is equal to x_i, for some i, rather than $\overline{x_i}$. Monotone instances of *Satisfiability* are very easy to solve: they are always satisfiable by setting each variable equal to 1.

For example, suppose we have $(x_1 \vee x_2)$ & $(x_1 \vee x_3)$ & $(x_2 \vee x_3)$. This is monotone, and indeed the assignment that sets all three variables to 1 ("True") satisfies all the clauses. But we can observe that this is not the only satisfying assignment; we could also have set x_1 and x_2 to 1, and x_3 to 0. Indeed, for any monotone instance, it is natural to ask how few variables we need to set to 1 in order to satisfy it.

Given a monotone instance of *Satisfiability*, together with a number k, the decision problem *Monotone Satisfiability with Few True Variables* asks: Is there a satisfying assignment for the instance in which at most k variables are set to 1? Prove this problem is $\mathcal{NP}$*-complete*.

[10 points]

Q.3 Given
- A monotone CNF formula $C_1 \wedge C_2 \wedge \dots \wedge C_m$; where each clause contains only non-negated variables.
- An integer k.

To check: If there exists a satisfying assignment in which at most k variables are set to 1 (True).

Certificate:
A valid certificate is:
- A set $S \subseteq \{x_1, \dots, x_n\}$ with $|S| \leq k$, representing the variables to be set to 1.
- All other variables are set to 0.

Verification Algorithm:
Given the set S, the verifier:
~~1 Check the set S, the verifier~~:
1. Check $|S| \leq k$.
2. For each clause C_i, verifies that at least one variable in the clause is in S. Because the formula is monotone, a clause is satisfied iff atleast one variable in it is true.

Each step takes polynomial time in input size.

Checking $|S| \le k$ is polynomial time:
- S contains at most n variables (where n = no of variables in the formula).
- Counting the size of a set takes $O(n)$
- Comparing 2 integers (size of S and k) takes $O(1)$

Thus this runs in $O(n)$ time.

Checking each clause is satisfied in polynomial time:

Let:
m = no of clauses
l_i = no of variables in clause C_i
total formula size $= \sum_i l_i$
The verifier must check
for each clause C_i, does it contain at least one variable from certificate S?

For each clause C_i:
- You scan the list of variables: $O(l_i)$ time
- For each variable x_j, you check whether it is in S

If you store S in a hash table or boolean array, lookup is $O(1)$

Thus checking a single clause C_i takes $O(l_i)$
For all clauses:
$O(l_1 + l_2 + l_3 + \dots + l_m) = O(\text{size of input formula})$
This is linear in formula size, which is polynomial.

∴ Total Time complexity
Adding up steps:
- Check size of S: $O(n)$
- Checking all clauses: $O(\text{size of formula})$

$O(n + \text{size of formula}) = O(\text{input size})$
This is polynomial in the size of input.

We reduce from Vertex Cover, which is a classic NP-complete problem.

Vertex Cover:
Instance: A graph $G=(V,E)$ and integer k.

Is there a set of vertices $S \subseteq V$, $|S| \leq k$, such that every edge has at least one endpoint in S?

Reduction: VC $\rightarrow$ Monotone-SAT-few-true
Given a graph $G=(V,E)$, where $V=\{v_1, v_2, \ldots, v_n\}$ and $E=\{e_1, e_2, \ldots, e_m\}$
we construct a monotone CNF formula F as follows

Construction:
1. Create one boolean variable for each vertex.
For each vertex $v_i \in V$, create a variable x_i.
2. Create one clause for each edge.
Let an edge be $e=(v_i, v_j)$.
Create the clause
$(x_i \vee x_j)$
This clause is monotone because variables are not negated.

3. The parameter k stays the same.
We ask whether the formula can be satisfied with at most the same number k of variables set to 1.

Example of Reduction:
A graph with edges
$(v_1, v_2), (v_1, v_3), (v_2, v_3)$ becomes
$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3)$
This is a monotone CNF formula

Proof the Reduction is Correct:

We must show
G has a vertex cover of size $\leq k \Leftrightarrow$
F has a satisfying assignment with $\leq k$ variables set to 1.

If G has vertex cover S of size $\leq k$, then F is satisfiable with $\leq k$ true variables

Let S be a vertex cover
Set variable $x_i = 1$ iff $v_i \in S$
All other variables $= 0$.

Consider any clause $(x_i \vee x_j)$
This corresponds to edge (v_i, v_j)
Since S is a vertex cover, at least one of v_i or v_j is in S, so at least one of x_i or x_j is set to 1
Thus all clauses are satisfied, using $\leq k$ true variables

- If f has a satisfying assignment with $\leq k$ true variables, then G has a vertex cover of size $\leq k$

Let T be the set of variables set to 1
Construct a set of vertices:

$$S = \{ v_i : x_i \in T \}$$

$|S| = |T| \leq k$
Take any edge (v_i, v_j)
Its corresponding clause is $(x_i \vee x_j)$
Since the formula is satisfied, at least one of x_i or x_j is 1
Thus at least one of v_i, v_j is in S
So S is a vertex cover.

The reduction is polynomial time:
We create one variable per vertex: $O(|V|)$
We create one clause per edge: $O(|E|)$
No step involves more than polynomial time.

Thus the reduction is polynomial.

$\therefore$ The problem is NP. We used a polynomial time reduction from Vertex Cover. The reduction is correct in both directions
$\therefore$ Monotone satisfiability with few true variables is NP complete.