COMP 170 HW 9

Ayda Aricanli, Cece Crumlish, Liam Strand

November 29, 2022

4SAT is the problem where, given a formula in conjunctive normal form with exactly *four* literals in each clause, you want to know if it can be satisfied by some assignment.

Prove that 4SAT is NP-Hard.

Proof by Mapping Reduction: from 3SAT to 4SAT

Any given 3SAT expression can be converted into an equivalent 4SAT expression as follows:

F on input $\langle\phi\rangle$ where ϕ is a 3SAT formula.

1. Construct ϕ' as follows:
    1. For each clause $(a \lor b \lor c)$ in ϕ, output the extended clause $(a \lor b \lor c \lor \psi)$.
        - We will use ψ, but any novel literal is acceptable.
    2. After all clauses have been extended, insert $(\overline{\psi} \lor \overline{\psi} \lor \overline{\psi} \lor \overline{\psi})$ at the end of ϕ', forming an expression of the form $(a \lor b \lor c \lor \psi) \land (d \lor e \lor f \lor \psi) \land ... \land (\overline{\psi} \lor \overline{\psi} \lor \overline{\psi} \lor \overline{\psi})$.
2. Output ϕ'.

F is a Polynomial Time Function:

Let n be the number of clauses in ϕ. Step 1 iterates n times, processing one clause at a time. We add one variable to each clause, which takes constant time to create. In Step 2 we add one additional clause, which takes constant time. Therefore Step 1 takes $O(n)$ time, and Step 2 takes $O(1)$ time. Thus F computes an equivalent 4SAT expression in $O(n)$ time.

Case Analysis:

$\phi \in$ 3SAT $\implies \phi' \in$ 4SAT
$\rightarrow$ ϕ is satisfiable
$\rightarrow$ there exists some assignment of variables in ϕ such that each clause in ϕ evaluates to true
$\rightarrow$ every clause in ϕ is satisfiable
$\rightarrow$ since the final clause in ϕ' is $(\overline{\psi} \lor \overline{\psi} \lor \overline{\psi} \lor \overline{\psi})$, ϕ' is satisfiable only when ψ is false.
$\rightarrow$ since every clause in ϕ' is constructed by adding ψ to a clause in ϕ using $\lor$, every clause in ϕ' (except for the last one) is satisfiable regardless of ψ's value
$\rightarrow$ ϕ' is satisfiable when ψ is false
$\rightarrow$ ϕ' is satisfiable.

$\phi \notin$ 3SAT $\implies \phi' \notin$ 4SAT
$\rightarrow$ ϕ is not satisfiable
$\rightarrow$ there is no assignment of variables in ϕ such that every clause in ϕ evaluates to true
$\rightarrow$ for every assignment of variables in ϕ, some clause evaluates to false
$\rightarrow$ since the final clause in ϕ' is $(\overline{\psi} \lor \overline{\psi} \lor \overline{\psi} \lor \overline{\psi})$, ϕ' is satisfiable only when ψ is false.
$\rightarrow$ when ψ is true, ϕ' is unsatisfiable
$\rightarrow$ since every clause in ϕ' is constructed by adding ψ to a clause in ϕ using $\lor$, when ψ is false, each clause evaluates to true if and only if its associated clause in ϕ also evaluates to true
$\rightarrow$ since some clause in ϕ evaluates to false, some clause in ϕ' evaluates to false
$\rightarrow$ ϕ' is not satisfiable.

Therefore, 3SAT $\leq_p$ 4SAT. Since 3SAT is NP-Hard, so is 4SAT. $\square$

Prove that DOUBLE-CLIQUE is NP-Complete

$$\text{DOUBLE-CLIQUE} = \{\langle G, k\rangle \ \mid G \text{ has at least 2 cliques, each of size greater than or equal to } k\}$$

Proof by construction: DOUBLE-CLIQUE is in NP

Construction of a deterministic polynomial-time verifier V_{DC} for DOUBLE-CLIQUE.

We begin by constructing a polynomial-time verifier V_C for CLIQUE, which is a language identical to DOUBLE-CLIQUE with the modification that it requires at least 1 clique of size greater than or equal to k, rather than DOUBLE-CLIQUE's 2.

Construction of a verifier for CLIQUE V_C

$V_C\langle\langle G, k\rangle, c\rangle$ is defined as follows.

On input $\langle\langle G, k\rangle, c\rangle$ where c is a list of nodes in G representing a candidate solution to CLIQUE:

1. Test whether c is in G and of size k
2. Test whether G contains edges connecting all the nodes in c
3. If c passes 1 and 2, **accept** otherwise **reject**.

V_C is polynomial-time

Since c is a list on the order of n, checking if every element in c is also in G's vertex list is an n^2 operation (for each element in c we must traverse the vertex list). Checking whether c is of size k is $O(n)$ because we must traverse c to check its size. Thus step 1 is $O(n^2)$

G has a maximum number of edges on the order of n^2, so traversing its edge list is $O(n^2)$. For every pair of nodes in c, we must traverse the edge list of G. There are on the order of n^2 unique pairs of nodes in c. Therefore, for each pair, we must traverse the edge list to see if the pair is connected. We do this n^2 operation n^2 times. Thus step 2 is $O(n^4)$

Checking the output of steps 1 and 2 is $O(1)$, so step 3 is $O(1)$.

Overall, V_C is $O(n^4)$ which is polynomial-time.

Case analysis of V_C on CLIQUE

c is a solution to CLIQUE on $\langle G, k\rangle$
$\rightarrow$ every vertex in c is in G's vertex list and c is of length k
$\wedge$ every pair of elements in c is connected by an edge in G's edge list
$\rightarrow$ c passes steps 1 and 2 of V_C
$\rightarrow$ V_C accepts c.

c is not a solution to CLIQUE on $\langle G, k\rangle$
$\rightarrow$ some vertex in c is not in G's vertex list or c is not of length k
$\vee$ some pair of elements in c is not connected by an edge in G's edge list
$\rightarrow$ c does not pass steps 1 and 2 of V_C
$\rightarrow$ V_C rejects c.

Construction of a verifier for DOUBLE-CLIQUE V_{DC}

$V_{DC}\langle\langle G, k\rangle, c_1, c_2\rangle$ is defined as follows.

1. If c_1 and c_2 contain the same vertices, **reject**
2. Run V_C on $\langle\langle G, k\rangle, c_1\rangle$. If V_C rejects, **reject**.
3. Run V_C on $\langle\langle G, k\rangle, c_2\rangle$. If V_C rejects, **reject**.
4. Otherwise, **accept**

V_{DC} is polynomial-time

Since the lists c_1 and c_2 are both on the order of n, checking their set equivalency can be performed in $O(n^2)$. For each element in c_1, search for that element in c_2. Then do the reverse.

Running V_C on c_1 takes $O(n^4)$ as described above.

Running V_C on c_2 takes $O(n^4)$ as described above.

Assuming V_{DC} has not yet rejected, accepting is $O(1)$.

Taken together, V_{DC} is $O(n^4)$, and is therefore polynomial-time.

Case analysis of V_{DC} on DOUBLE-CLIQUE

(c_1, c_2) is a solution to DOUBLE-CLIQUE on $\langle G, k\rangle$
$\rightarrow$ c_1 and c_2 are distinct cliques of size k in G
$\rightarrow$ since c_1 and c_2 contain distinct sets of nodes in G, we do not reject on step 1
$\wedge$ since c_1 is a solution to CLIQUE on G, V_C accepts c_1
$\wedge$ since c_2 is a solution to CLIQUE on G, V_C accepts c_2
$\rightarrow$ since we have not rejected, V_{DC} accepts (c_1, c_2).

(c_1, c_2) is not a solution to DOUBLE-CLIQUE on $\langle G, k\rangle$
$\rightarrow$ c_1 and c_2 are not distinct cliques of size k in G
$\rightarrow$ since c_1 and c_2 contain the same set of nodes in G, we reject on step 1
$\vee$ since c_1 is not a solution to CLIQUE on G, V_C rejects c_1
$\vee$ since c_2 is not a solution to CLIQUE on G, V_C rejects c_2
$\rightarrow$ since one of the above cases must have rejected, V_{DC} rejects (c_1, c_2).

Proof by reduction: DOUBLE-CLIQUE is in NP-Hard

CLIQUE $\leq_p$ DOUBLE-CLIQUE.

For any CLIQUE problem $\langle G, k\rangle$, compute $\langle G', k\rangle$ using the following function f.

f = On input $\langle G, k\rangle$ where G is a graph and k is an clique size:

1. Construct G' as follows:
    1. Copy G into G'
    2. Append vertices $c_1 ... c_k$ to the vertex list of G'
    3. Append edges $(c_1, c_2), (c_1, c_3), ..., (c_1, c_k), (c_2, c_3), ..., (c_{k-1}, c_k)$ to the edge list of G'
        - This adds an additional clique of size k to G' that is disconnected from every vertex in G' that was copied from G.
2. Output $\langle G', k\rangle$.

f is polynomial-time

Since there are n vertices in G, there are order of n^2 edges in G. Therefore, copying G into G' is an $O(n^2)$ operation.

Appending vertices $c_1, ..., c_k$ to the vertex list of G', is $O(k)$, but k is on the order of n, so this operation is $O(n)$.

Since we are adding exactly $\frac{k\times(k-1)}{2}$ edges to the edge list of G', this step takes $O(n^2)$ time, since n^2 is an upper bound for $\frac{k\times(k-1)}{2}$.

Overall, this takes $O(n^2)$ time, which is polynomial.

Case analysis of f on CLIQUE

$\langle G, k\rangle \in \text{CLIQUE} \implies \langle G', k\rangle \in \text{DOUBLE-CLIQUE}$
$\rightarrow$ G contains at least one clique of size k
$\rightarrow$ since G' contains one more clique of size k than G, G' contains at least two cliques of size k.
$\rightarrow$ $\langle G', k\rangle$ is in DOUBLE-CLIQUE

$\langle G, k\rangle \notin \text{CLIQUE} \implies \langle G', k\rangle \notin \text{DOUBLE-CLIQUE}$
$\rightarrow$ G contains no cliques of size k
$\rightarrow$ since G' contains one more clique of size k than G, G' contains exactly one clique of size k.
$\rightarrow$ $\langle G', k\rangle$ is not in DOUBLE-CLIQUE

We have constructed a polynomial-time function that maps CLIQUE problems to DOUBLE-CLIQUE problems. Thus, CLIQUE $\leq_p$ DOUBLE-CLIQUE. Therefore, DOUBLE-CLIQUE is in NP-Hard.

Finally, with triumph...

We have shown that DOUBLE-CLIQUE is in NP via the construction of a polynomial-time verifier and that DOUBLE-CLIQUE is in NP-Hard by reduction to CLIQUE.

Therefore, DOUBLE-CLIQUE is NP-Complete. $\square$