Recall the problems:

$$\text{SAT} = \{\langle \phi \rangle \mid \phi \text{ has at least } \textit{one} \text{ satisfying assignments}\}$$

and

$$\text{DOUBLE-SAT} = \{\langle \phi \rangle \mid \phi \text{ has at least } \textit{two} \text{ satisfying assignments}\}$$

(1) For the following Boolean formulae, count how many satisfying assignments they have, then decide if they are in SAT and DOUBLE-SAT?

- $\phi_1 = x \wedge (y \vee \bar{x}) \wedge (z \vee \bar{y})$
- $\phi_2 = (x \vee y) \wedge \big(\overline{(x \vee z)} \vee (\bar{z} \wedge \bar{y})\big)$
- $\phi_3 = (\bar{x} \vee y \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee \bar{z})$

Solution

Truth tables:

x	y	z	ϕ_1	ϕ_2	ϕ_3
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	0	0	1
1	1	1	1	0	1

ϕ_1 has only 1 satisfying assignment so

$$\langle \phi_1 \rangle \in \text{SAT} \quad \text{but} \quad \langle \phi_1 \rangle \notin \text{DOUBLE-SAT}$$

ϕ_2 has 2 satisfying assignments so

$$\langle \phi_2 \rangle \in \text{SAT} \quad \text{and} \quad \langle \phi_2 \rangle \in \text{DOUBLE-SAT}$$

ϕ_3 has 5 satisfying assignments so

$$\langle \phi_3 \rangle \in \text{SAT} \quad \text{and} \quad \langle \phi_3 \rangle \in \text{DOUBLE-SAT}$$

(2) Define

$$\text{TRIPLE-SAT} = \{\langle\phi\rangle \mid \phi \text{ has at least 3 satisfying assignments}\}$$

Complete the proof below to show that TRIPLE-SAT is **NP-complete**.

TRIPLE-SAT is NP-complete

We need to show that TRIPLE-SAT $\in$ **NP** $\boxed{\text{and}}$ SAT $\leq_P$ TRIPLE-SAT.

1/2 TRIPLE-SAT $\in$ **NP**:
On input $\langle\phi(x_1,\dots,x_n)\rangle$, non-deterministically guess $\boxed{3}$ different assignments for the Boolean variables $x_1,\dots,x_n$, and verify whether they all $\boxed{\text{satisfy}}$ ϕ.
The verification step only costs $O(\boxed{n})$.

2/2 SAT $\leq_P$ TRIPLE-SAT:
On input $\langle\phi(x_1,\dots,x_n)\rangle$, introduce 2 new Boolean variables a and b, and output the formula:

$$\Psi(x_1,\dots,x_n,a,b) = \phi(x_1,\dots,x_n) \wedge (a \vee b).$$

- (1/2) If $\langle\phi\rangle \in$ SAT then this means that ϕ has at least $\boxed{\text{one}}$ satisfying assignment, and therefore $\phi \wedge (a \vee b)$ has at least $\boxed{3}$ satisfying assignments because the added clause $(a \vee b)$ can be satisfied in 3 ways:

$$(a,b) \in \{(1,0),(0,1),\boxed{(1,1)}\}.$$

So $\langle\Psi\rangle = \langle\phi \wedge (a \vee b)\rangle \in$ TRIPLE-SAT.

- (2/2) If $\langle\phi\rangle \notin$ SAT then $\phi \wedge (a \vee b)$ $\boxed{\text{cannot}}$ have a satisfying assignment because

$$0 \wedge 0 = 0 \qquad \text{and} \qquad 0 \wedge \boxed{1} = 0.$$

So $\langle\Psi\rangle = \langle\phi \wedge (a \vee b)\rangle \notin$ TRIPLE-SAT.

Therefore, SAT $\leq_P$ TRIPLE-SAT, and hence TRIPLE-SAT is **NP-complete**.

(3) Let $\mathbb{N} = \{1, 2, 3, \ldots\}$ be the set of natural numbers, and define

$$\mathbb{N}_t = \{n \in \mathbb{N} \mid n < t\} \qquad \text{(i.e. } \mathbb{N}_t = \{1, 2, \ldots, t-1\}\text{)}$$

Consider the following restricted version of the **Subset-Sum Problem** (SSP)

$$\text{SSP} = \{\langle \mathcal{S}, t\rangle \mid \mathcal{S} \subsetneq \mathbb{N}_t \text{ is a finite set, and } t \in \mathbb{N}, \text{ such that there is a subset of } \mathcal{S} \text{ whose sum is } t\}$$

Here the notation "$A \subsetneq B$" means that $A \subset B$ but $A \neq B$.

SSP is **NP-complete**.

Define:

$$\text{DOUBLE-SSP} = \{\langle \mathcal{S}, t\rangle \mid \mathcal{S} \subset \mathbb{N}_t \text{ is a finite set, and } t \in \mathbb{N}, \text{ such that there are two distinct subsets of } \mathcal{S} \text{ which both sum to } t\}$$

Complete the proof below to show that DOUBLE-SSP $\in$ **NP-complete**.

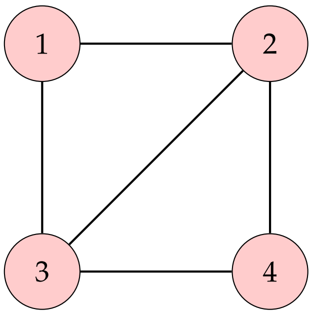
DOUBLE-SSP is NP-complete

1/2 DOUBLE-SSP $\in$ **NP** because we can $\boxed{\text{verify}}$ if 2 given subsets $\mathcal{T}_1, \mathcal{T}_2$ of $\mathcal{S}$ sum to t and check that $\mathcal{T}_1 \neq \mathcal{T}_2$ in time $O(\boxed{n})$, which is polynomial time.

2/2 SSP $\leq_P$ DOUBLE-SSP.
Since $\mathcal{S} \subsetneq \mathbb{N}_t$ then $\mathcal{S}$ $\boxed{\text{misses}}$ at least an element m from $\{1, 2, 3, \ldots, t-1\}$.
Select such an m and build a DOUBLE-SSP instance $\langle \mathcal{S}', t\rangle$ where $\mathcal{S}' = \mathcal{S} \cup \{m, t-m\}$.
• (1/2) Showing that: $\langle \mathcal{S}, t\rangle \in \text{SSP} \implies \langle \mathcal{S}', t\rangle \boxed{\in}$ DOUBLE-SSP.
Since $\langle \mathcal{S}, t\rangle \in$ SSP then there is a subset $\mathcal{T} \subset \mathcal{S}$ which $\boxed{\text{sums}}$ to t.
A second solution is given by $\boxed{\{m, t-m\}}$.

• (2/2) Showing that: $\langle \mathcal{S}', t\rangle \in \text{DOUBLE-SSP} \implies \langle \mathcal{S}, t\rangle \in \boxed{\text{SSP}}$.
If $\langle \mathcal{S}', t\rangle \in$ DOUBLE-SSP then there are two distinct subsets $\mathcal{T}$ and $\mathcal{U}$ of $\mathcal{S}'$ that both sum to t.
$\mathcal{T}$ and $\mathcal{U}$ cannot both be $\{m, t-m\}$ because they must be $\boxed{\text{different}}$. So one of them must be a $\boxed{\text{subset}}$ of $\mathcal{S}$.
So, $\langle \mathcal{S}, t\rangle \in$ SSP.

(4) Consider the following graph:



and recall that

$$\text{CLIQUE} = \{\langle G, k\rangle \mid \text{The graph } G \text{ has a } k\text{-clique}\}$$

- What is the largest k for which this graph satisfies CLIQUE?
- In general, how many edges does a k-clique have (as a function of k)?

Solution

• There are two *triangles* (3-cliques): $\{1,2,3\}$ and $\{2,3,4\}$. However, there is no 4-clique, since there are only 4 nodes, and one edge is missing.

Thus $k = 3$ is the answer.

• All pairs of nodes must have an edge between them, and the number of pairs of k nodes is

$$\binom{k}{2} = \frac{k!}{2!(k-2)!} = \frac{1}{2}k(k-1).$$

(5) A ***vertex cover*** of an undirected graph G is a subset of the vertices where every edge of G touches one of those vertices.

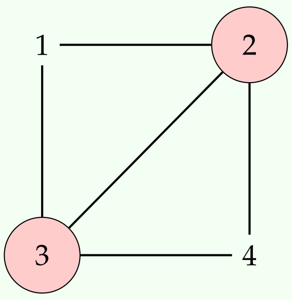
The VERTEX-COVER problem asks whether a graph contains a vertex cover of a specified size:

$$\text{VERTEX-COVER} = \{\langle G, k\rangle \mid G \text{ is an undirected graph that has a } k\text{-vertex cover}\}.$$

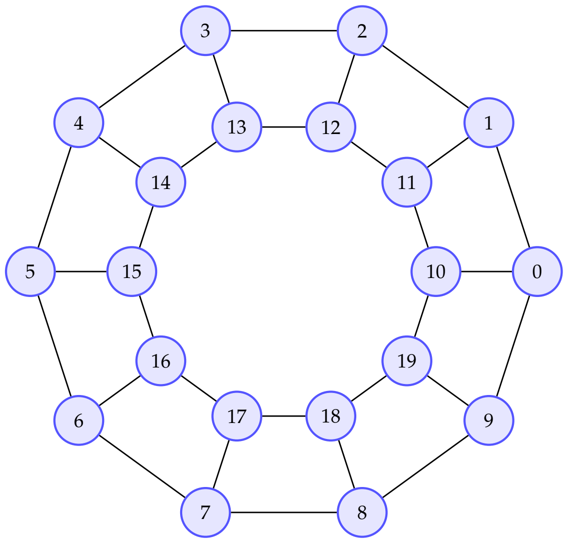
What is the smallest k for which the graph from the previous question satisfies VERTEX-COVER?

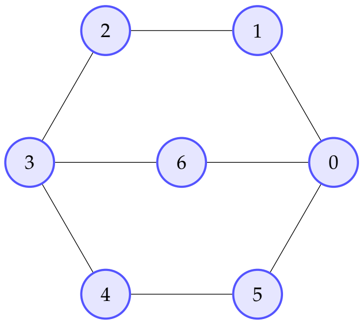
Solution

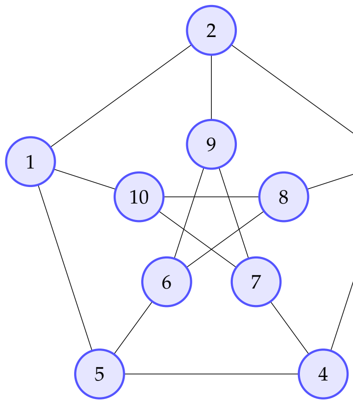
$k = 2$



(6) Do the following graphs have Hamiltonian circuits?







Solution

Yes for the first one, e.g.

$$0 - 1 - 11 - 12 - 2 - 3 - 13 - 14 - 4 - 5 - 15 -$$

$$16 - 6 - 7 - 17 - 18 - 8 - 9 - 19 - 10 - 0.$$

No for the second. To visit 6 we must have 0 and 3 as its immediate neighbours. This forces the inclusion of the path $0 - 6 - 3$ and makes it impossible to visit the upper vertices 1 & 3 and the lower ones 4 & 5 without revisiting 0 or 3.

Another way to also see it is to realise that we must have the paths $0 - 1 - 2 - 3$ and $0 - 5 - 4 - 3$. This makes it impossible to create a hamiltonian cycle that also includes 6.

No for the third. (The proof is a little more involved... [and you were not asked for proofs])

(7) Let $x_1, x_2, \dots, x_n$ be Boolean variables, and let ϕ be a Boolean formula written in 3-cnf (Conjunctive Normal Form, like ϕ_3 in the first excercise) given by

$$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_\ell,$$

where each clause c_m has the form $\alpha \vee \beta \vee \gamma$, where each of α, β, γ is a literal: a variable x_i or its negation $\bar{x}_i$.

The 3SAT problem is **NP-complete**, and asks if a given 3-cnf formula is satifiable.

We showed in the lecture that the Subset-Sum Problem (SSP) is in **P**. Now, show that SSP is **NP-complete** by reducing 3SAT to it, i.e. show that

$$\text{3SAT} \leq_P \text{SSP}$$

You may find it easier if you study the discussion at https://saravananthirumuruganathan.wordpress.com/2011/02/07/detailed-discussion-on-np-completeness-of-subset-sum/ then the proof given in the textbook.

(8) Prove that CLIQUE is **NP-complete** by reducing the VERTEX-COVER problem to CLIQUE. (VERTEX-COVER is **NP-complete**)

You can use the following reduction from VERTEX-COVER to CLIQUE:

> Suppose we are given an instance $\langle G, k\rangle$ of VERTEX-COVER. Construct the instance $\langle G', n-k\rangle$ of CLIQUE, where n is the total number of nodes of G, and G' is G with the set of edges complemented (i.e. G' has an edge if and only if G does not have that edge).

Solution

We must show that G has a node cover of size k if and only if G' has a clique of size $n-k$.

First, let C be a node cover of G of size k. We claim that C', the complement of the nodes in C, is a clique in G' of size $n-k$. Surely C' is of size $n-k$. Suppose it is not a clique. Then there is a pair of nodes (u, v) that do not have an edge in G'. Thus this edge is in G. But neither u nor v is in C, contradicting the assumption that is is a node cover.

Conversely, if C' is a clique of size $n-k$ in G', then we claim that C the complement of C', is a node cover of size k in G. The argument is similar: if (u, v) is an edge of G not covered by C, then both u and v are in C', but the edge (u, v) is not in G', contradicting the assumption that C' is a clique.

(9) Given a graph G with an even number of vertices n, does G have an $n/2$-clique?

Hint: Reduce CLIQUE to HALF-CLIQUE. You need to figure out how to add vertices to adjust the size of the largest clique depending on whether $k = n/2$ or $k > n/2$ or $k < n/2$ (e.g., if $k = n/2$, just produce G).

Solution

Let (G, k) be an instance of CLIQUE, and suppose G has n vertices. We produce an instance of the HALF-CLIQUE, as follows:

- If $k = n/2$, just produce G.

  Note that G has a half-clique if and only if it has a clique of size k in this case.

- If $k > n/2$, add $2k - n$ isolated vertices (vertices with no incident edges).

  The resulting graph has a half-clique (whose size must be $(n + (2k - n))/2 = 2k$, if and only if G has a clique of size k.

- If $k < n/2$, add $n - 2k$ vertices, and connect them in all possible ways to each other and to the original vertices of G.

  The new graph thus has $2(n - k)$ vertices. The new vertices, plus a clique of size k in G form a clique of size $(n - 2k) + k = n - k$, which is half the number of vertices in the new graph.

  Conversely, if the new graph has a half-clique, then it must include at least $(n - k) - (n - 2k) = k$ vertices of the graph G, implying that G has a clique of size k.

These steps complete a reduction of CLIQUE to HALF-CLIQUE. It is evidently performable in polynomial time, since the number of new vertices and edges is at most the square of the original number of vertices, and the rules for adding vertices and edges are simple to carry out.

(10) Show that the class **P** is closed under:

- Union.
- Concatenation.
- Complementation.

Solution

Union. Test for membership in one language and then, if the input is not in the first, test for membership in the second.

The time taken is no more than the sum of the times taken to recognize each language. Since both ar in **P**, then each can be recognized in polynomial time, and the sum of polynomials is a polynomial. Thus, their union is in **P**.

Concatenation. Let L_1 and L_2 be languages in **P**, and suppose we want to recognize their concatenation.

The idea is to try all the possible ways in which we can split the string into a concatenation of 2 strings from the given languages.

Input: A string w of length n.
Output: *accept* if $w \in L_1L_2$; *reject* otherwise.

```
1: for $i \leftarrow 0, \dots, n$ do
2:     if the the first $i$ symbols of $w$ is in $L_1$ and the remainder of $w$ is in $L_2$. then
3:         return accept                                  ▷ $w$ is in $L_1L_2$
4:     end if
5: end for
6: return reject
```

The running time of this test is at most n times the sum of the running times of the recognizers for L_1 and L_2. Since the latter are both polynomials, so is the running time for the TM just described.

Complementation. Given a polynomial-time TM M for L, we construct a TM M' that decides $\overline{L}$ (the complement of L) as follows:

> On input w:
> Run M on w.
> If M accepts, *reject*; if M rejects, *accept*

Since M' does the opposite of whatever M does then it decides $\overline{L}$.

Furthermore, M' also runs in polynomial time because it only needs one extra step after M halts.

(Alternative approach: Simply swap the accept and reject state. The running time of the modified machine does not change.)