Monash University
Faculty of Information Technology

FIT2014 Theory of Computation

Lecture 28
NP-completeness: reductions to SATISFIABILITY

slides by Graham Farr

Overview

- Modelling languages using logic: general advice
- Extended example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY
- Sketchy overview of Cook's Theorem

Reducing to SATISFIABILITY

To understand how *any* language in NP can be polynomial-time reduced to SAT, we will look at some specific languages.

Reducing to SATISFIABILITY

Suppose we have a language which we want to reduce to SAT.

Approach:

1. Introduce Boolean variables to describe the parts of the certificate.
2. (Possibly introduce other variables to help describe the conditions under which a certificate is valid.)
3. Translate the **rules of the language** into CNF.
4. Put it all together as an algorithm.

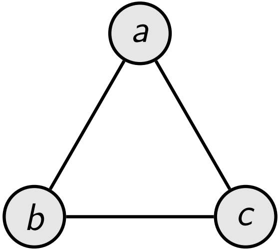
Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

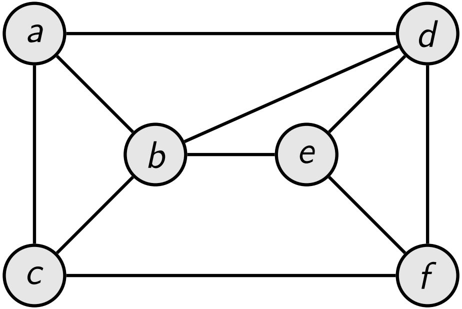
Let's do it for the following language.

PARTITION INTO TRIANGLES:
the set of graphs G such that the vertex set of G can be partitioned into 3-sets (i.e., sets of size 3) such that each of these 3-sets induces a triangle in G.

Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

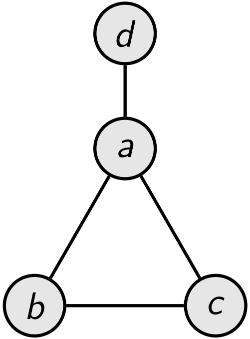
Some members of PARTITION INTO TRIANGLES:

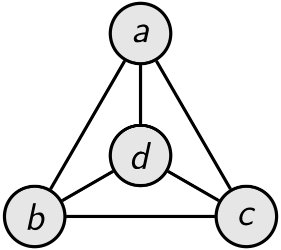


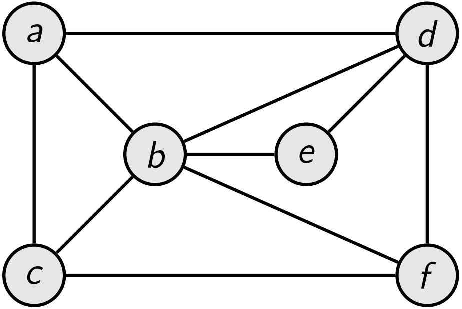


Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

Some <u>non-members</u> of PARTITION INTO TRIANGLES:

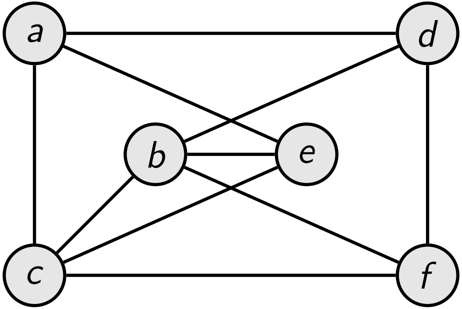


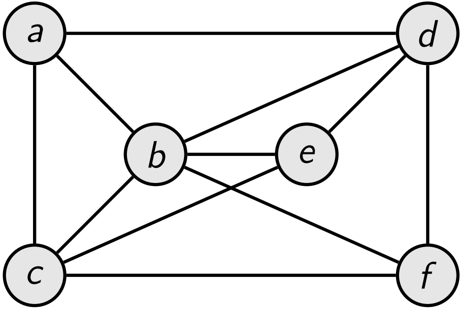




Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

What about these?





Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

PARTITION INTO TRIANGLES

Certificate:

State which triangles, in the graph, are in the partition.

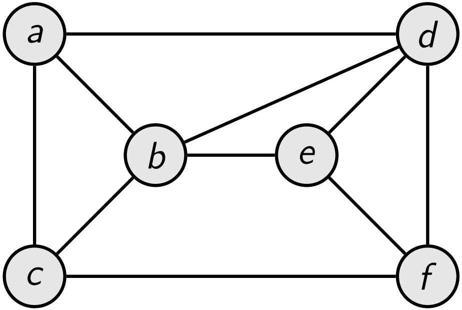
Using Boolean variables:
for each triangle T,

$$x_T = \begin{cases} \text{True}, & \text{if } T \text{ is in the partition;} \\ \text{False}, & \text{otherwise.} \end{cases}$$

Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

Variables for this graph:

x_{abc}
x_{abd}
x_{bde}
x_{def}



In general,
$\#$ triangles $= O(n^3)$

Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

PARTITION INTO TRIANGLES

Rules of the language:

The set of triangles must form a partition of V(G).
i.e.:

- every vertex belongs to at least one triangle, *and*
- no vertex belongs to more than one triangle.

Let's look at each of these in turn.

Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

"Every vertex belongs to at least one triangle."

For each vertex: at least one of the triangles at that vertex must be included in the partition.

Express this in terms of our variables.

For each vertex, use a clause

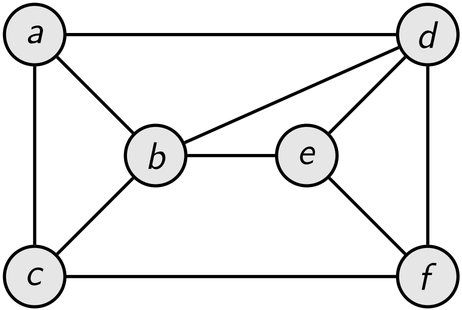
$$x_{T_1} \vee x_{T_2} \vee \ldots \vee x_{T_k}$$

where $T_1, T_2, \ldots, T_k$ are the triangles at the vertex.

# of these clauses $= n$ (i.e., # vertices of graph)

Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

Vertex a: $x_{abc} \vee x_{abd}$
Vertex b: $x_{abc} \vee x_{abd} \vee x_{bde}$
Vertex c: x_{abc}
Vertex d: $x_{abd} \vee x_{bde} \vee x_{def}$
Vertex e: $x_{bde} \vee x_{def}$
Vertex f: x_{def}



Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

"No vertex belongs to more than one triangle."

For each vertex, and each pair of triangles at that vertex:

at least one of the triangles in this pair must not be included in the partition.

Express this in terms of our variables.

For each vertex, and each pair of triangles T_i, T_j at the vertex, use a clause

$$\neg x_{T_i} \vee \neg x_{T_j}$$

$$\begin{aligned} \#\text{ of these clauses} &\leq n \cdot (d(d-1)/2)^2 \quad (\text{where } d = \text{degree}) \\ &= O(n^5) \end{aligned}$$

Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

Vertex a: $\neg x_{abc} \vee \neg x_{abd}$

Vertex b: $\neg x_{abc} \vee \neg x_{bde}$

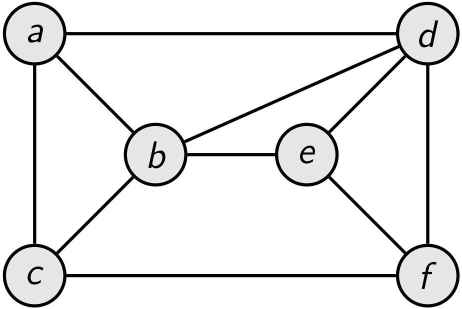
$\neg x_{abd} \vee \neg x_{bde}$

$\neg x_{abc} \vee \neg x_{abd}$

... (included earlier)

Vertex d: $\neg x_{abd} \vee \neg x_{def}$

$\neg x_{bde} \vee \neg x_{def}$



Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

We now have clauses for:

- every vertex belongs to at least one triangle, *and*
- no vertex belongs to more than one triangle.

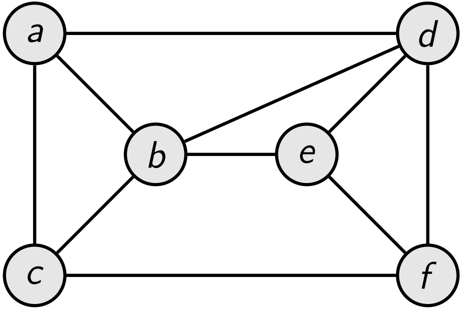
Take the conjunction of all clauses.

This gives a Boolean formula φ
which is satisfiable if and only if the original graph has a partition into triangles.

Specify the mapping from G to φ as an algorithm,
and show it's a polynomial-time reduction.

Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

$$
\begin{aligned}
\varphi \quad = \quad & (x_{abc} \vee x_{abd}) \\
\wedge\ & (x_{abc} \vee x_{abd} \vee x_{bde}) \\
\wedge\ & (x_{abc}) \\
\wedge\ & (x_{abd} \vee x_{bde} \vee x_{def}) \\
\wedge\ & (x_{bde} \vee x_{def}) \\
\wedge\ & (x_{def}) \\
\wedge\ & (\neg x_{abc} \vee \neg x_{abd}) \\
\wedge\ & (\neg x_{abc} \vee \neg x_{bde}) \\
\wedge\ & (\neg x_{abd} \vee \neg x_{bde}) \\
\wedge\ & (\neg x_{abd} \vee \neg x_{def}) \\
\wedge\ & (\neg x_{bde} \vee \neg x_{def})
\end{aligned}
$$



Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

Describe the reduction as an algorithm.

Input: Graph G

1. For each triangle T of G:
   Create new variable x_T.
2. For each vertex v of G:
   Let $T_1, T_2, \ldots, T_k$ be the triangles at v.
   Make the clause $x_{T_1} \vee x_{T_2} \vee \ldots \vee x_{T_k}$.
3. For each pair of triangles T_i, T_j which share a vertex:
   Make a clause $\neg x_{T_i} \vee \neg x_{T_j}$
4. φ := conjunction of all these clauses.
5. Output φ

Example: PARTITION INTO TRIANGLES $\leq_P$ SATISFIABILITY

Time complexity?

Main factor: $\#$ pairs of triangles that share a vertex $= O(n^5)$.

For each such pair, a small amount of work needs doing ...

Looks like time $O(1)$ or $O(n)$ for each ...

So the time complexity is polynomial.

Reducing to SATISFIABILITY

Other languages can be polynomial-time reduced to SAT by a similar approach.

Some good exercises of this type:

- 3-COLOURABILITY
- CUBIC SUBGRAPH:
  the set of graphs with a subgraph consisting entirely of vertices of degree 3
- HAMILTONIAN CIRCUIT

SAT Solvers:

- programs for solving SAT.
- can be used to solve other problems that are $\leq_P$ SAT.

Reducing to SATISFIABILITY

Experience with reducing to SAT suggests that any language in NP can be polynomial-time reduced to SAT.
(This is the Cook-Levin Theorem.)

But how to prove it?

Need a generic way of reducing from any specific language in NP to SAT.

Reducing to SATISFIABILITY

Very sketchy overview of proof of Cook-Levin Theorem:

Given L in NP, let M be a polynomial-time verifier for L.

Create variables for the certificate (considered as a binary string),
and more variables to describe all the components of M at all possible timesteps.

Make clauses to capture the working of the verifier.
These express the allowed transitions of the machine, and forbid any others.
(See Exercise Sheet 7, Q9.)

Show that the construction is indeed the desired polynomial-time reduction.

Revision

Things to think about:

- Try doing one of the other polynomial-time reductions we've mentioned, e.g.,

$$\text{CUBIC SUBGRAPH} \ \leq_P \ \text{SATISFIABILITY}$$

- Try doing a polynomial-time reduction from this problem to SATISFIABILITY:

  FA-Nonempty $:=$ $\{\, A : A$ is a Finite Automation that accepts at least one string$\,\}$.

  Try doing it in such a way that the logical expression you construct models the execution of the FA.
  Use Boolean variables to represent the letters in each position of the input string
  ...

Reading: Sipser, section 7.4, pp. 299–304.