CLIQUE, then SPACE COMPLEXITY

Cliques

- A *clique* C in an undirected graph $G = (V, E)$ is a subset of vertices ($C \subseteq V$) such that every two vertices in C are connected by an edge from E.

- In other words, a clique in an undirected graph is a subset of vertices which induces a complete subgraph

- A *k-clique* is just a clique with k vertices

Example

CLIQUE

- CLIQUE is the following language

$$\{(G,k): G\ is\ an\ undir\ \ graph\ with\ a\ k-clique\}$$

- CLIQUE is in NP?

- Yes. As usual, we could think of proving this in two ways:

- Verifier
- NTM

CLIQUE is NP proof 1

- We describe a verifier for CLIQUE

- Input: $((G, k), c)$

(*G*: arbitrary undir graph, *k*: arbit natural number , *c*: arbit set of vertices)

1. Test whether *c* is a set of vertices from *G*
2. Test if for any two vertices in *c*, *G* contains an edge connecting them
3. If 1,2 are a YES, accept. Otherwise reject

CLIQUE is NP proof 2

- We describe a non-deterministic TM which decides CLIQUE

- Input: (G, k)

(*G*: arbitrary undir graph, *k*: arbit natural number)

1. Nondeterministically select a subset *c* from the set of vertices of *G*
2. Test if for any two vertices in *c*, *G* contains an edge connecting them
3. If 1,2 are a YES, accept. Otherwise reject

CLIQUE is NP-complete

- We **describe** a polynomial time reduction of 3SAT to CLIQUE
- In other words, we describe a **ploytime** $TM_{Reduction}$ (program) that takes:

Input: an arbitrary instance of 3SAT (i.e. an arbitrary Boolean 3-CNF formula φ)
Outputs: an instance of CLIQUE (i.e. a pair $\left(G_\varphi, k_\varphi\right)$ of a graph and a natural number)

SUCH THAT: φ is satisfiable iff $\left(G_\varphi, k_\varphi\right)$ is in clique.

In other words: $TM_{CLIQUE}(\left(G_\varphi, k_\varphi\right))$ accepts iff $TM_{3SAT}(\varphi)$ accepts

$TM_{Reduction}$

- Note that it is not a decider
- **Input:** $\varphi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$
- Each of the a,b,c is a literal
- Example: $(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$

$TM_{Reduction}$

- **Output:** an undirected graph G_φ, and a natural number k_φ as follows:

- $k_\varphi = k$ where k is the number of clauses in the input formula φ

- G_φ has $3k$ vertices, k groups of 3 (call them *triples*: $t_1, t_2, \dots, t_k$)

-Each triple corresponds to a clause, and each vertex in a triple corresponds to a literal from the associated clause

- Edges of G_φ : connect every pair of vertices except:

  -If they are in the same triple, or

  -they represent contradictory literals (one is the negation of the other)

Example of a $TM_{Reduction}$ computation

- **Input:** $(x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)$
- **Output:**

$\neg x_1$ $\neg x_2$ $\neg x_2$

x_1 $\neg x_1$

x_1 x_2

x_2 x_2

Running Time of $TM_{Reduction}$

- **Input:** A formula with k clauses

- How many steps are required to construct (compute) the graph G_{φ}, and to compute the number k_{φ} ?

It works

- The input formula φ is satisfiable **iff** the output graph G_φ contains a k_φ-clique

>>>>:

- Suppose φ is satisfiable
- This means it has a satisfying assignment
- Every clause has at least one literal which is TRUE (based on the assignment)

Still proving it works

- Now we want to show that G_φ contains a k_φ-clique

- Recall: $k_\varphi = k$ where k is the number of clauses in φ
- Recall: Every clause has at least one literal which is TRUE

- On the graph G_φ , from every color (triple), choose a vertex (only one) which corresponds to a TRUE literal.

- **That collection of chosen vertices happens to be a *k*-clique, why?**

Because:

- *Every pair of the chosen vertices is connected by an edge from* G_φ

- Edges of G_φ connect every pair of vertices except:
    1. If they are in the same triple, or
    2. they represent contradictory literals (negations of each other)

- Every pair of the vertices we chose is neither 1 or 2

-we chose only one from each triple

-we choose TRUE literals, so no way that any pair represents a variable and its negation

Break

Proof of “It works” is still going

Still proving it works (the other direction)

- <<<<:

- Suppose that G_φ has a k-clique, call it W, and we want to show that φ is satisfiable

- I.e., we want to show that **there exists** a truth assignment that satisfies φ GIVEN THE FACT THAT G_φ has a k-clique

- In that clique W, no two vertices are of the same color/triple (by the construction of G_φ)

- In other words, every vertex represents a single literal from a particular clause, and that clause is unique for the vertex

- Claim: **There exists** an assignment that makes TRUE **all** of those literals labeling the vertices in W

- Why? Because the labels do not contradict each other

- That EXISTENT assignment clearly satisfies our input formula φ

So far

- We built a reduction TM
- We checked it is polytime
- We showed that if the input is satisfiable, then the output has a clique with as many vertices as there are clauses in the input
- We showed that if the output has a clique with as many vertices as there are clauses in the input, then the input must had been satisfiable

DONE!

With CLIQUE

Space Complexity

We discussed time, let’s discuss space!

Space complexity of a TM

- Let *M* be a decider deterministic TM which halts on all inputs
- The space complexity of *M* is the function *f* where $f(n)$ is the maximum number of tape cells that *M* **scans** on any input of length n
- We say *M* runs in space $f(n)$
- If *M* is nondeterministic where all branches halt on all inputs, then its space complexity is the maximum number of tape cells that *M* scans on any branch of its computation given any input of length n

Time Complexity CLASSES

- $SPACE\big(f(n)\big) =$
  $\{L\text{: } L \text{ is a language decidable by an } O\big(f(n)\big) \text{ space determinstic } TM\}$

- $NSPACE\big(f(n)\big) =$
  $\{L\text{: } L \text{ is a language decidable by an } O\big(f(n)\big) \text{ space nondeterminstic } TM\}$

- $PSPACE = \bigcup_{k \in \mathbb{N}} SPACE(n^k)$

- Wondering about NPSPACE?

PSPACE $\supseteq$ NP

- Surprised ?
- You shouldn’t be. Space is reusable
- SAT is in PSPACE
- In fact, it is in SPACE(n), linear, very sweet!

What is known so far:

- P ⊆ NP ⊆ PSPACE (= NPSPACE) ⊆ EXPTIME

- P $\subsetneq \mathrm{EXPTIME}$

- So at least one of the inclusions in the first line must be strict

- $EXPTIME = \bigcup_{k \in \mathbb{N}} TIME(2^{n^k})$

Also known

- EXPTIME $\subseteq$ NEXPTIME $\subseteq$ EXPSPACE
- NP $\subsetneq$ NEXPTIME and PSPACE $\subsetneq$ EXPSPACE
- ALL this is within the class of computable sets ☺