

# Undergraduate Complexity Theory

## Lecture 7: SAT

Marcyhm

July 11, 2022

### 1 Lecture Notes

Different SAT problems:

1. CIRCUIT-SAT
2. FORMULA-SAT a.k.a SAT
3. CNF-SAT
4. k-SAT
5. 3-SAT

bigger id number means less generality.

Boolean circuit  $C$ , which is acyclic graph, has  $n$  inputs  $x_1, \dots, x_n$ , and one output. Gates in  $C$  includes  $\vee, \wedge, \neg$  with fan-in 2, 2, 1 respectively. Circuit  $C$  computes a boolean function  $C$  or  $f_C : \{0, 1\}^n \rightarrow \{0, 1\}$ . When input to algo is a circuit  $\langle C \rangle$ , we measure runtime in terms of  $n := \#$  input gates,  $m := \#$  total gates.  $n \leq m \leq |\langle C \rangle| \leq O(m \log m)$ .

**Remark 1.1.** One circuit only operates on strings of a fixed length.

**Definition 1.2** (CIRCUIT-EVAL). Given  $\langle C, x \rangle$ , whether  $C(x)$  evaluates to 1?

$$\text{CIRCUIT-EVAL} = \{ \langle C, x \rangle : C(x) = 1 \}$$

CIRCUIT-EVAL  $\in$  P, and even in time  $O(m)$ , but it's not parallelizable.

**Definition 1.3** (CIRCUIT-SAT). Given  $\langle C \rangle$ , is there an  $x$  s.t.  $C(x) = 1$ ?

$$\text{CIRCUIT-SAT} = \{ \langle C \rangle : \exists x : C(x) = 1 \}$$

Brute force:  $O(2^n \text{poly}(n))$ . CIRCUIT-SAT  $\in$  P  $\iff$  P = NP.

**Definition 1.4** (FORMULA-SAT). “Formula” = Circuit where all fan-outs are 1.

**Definition 1.5** (CNF-SAT). Special case of FORMULA-SAT: formula is a big AND of “clauses”, which is formed by OR of “literals” (variable or negated variable). For CNF,  $m$  is defined as  $\#$  of clauses.

CNF-SAT is in EXP, with a brute force in  $O(2^m)$ . Also NP-complete.

**Definition 1.6** (k-SAT). Special case of CNF-SAT where all clauses have  $\leq k$  literals.

Brute force:  $O(2^n \text{poly}(n))$ . Also NP-complete.

**Theorem 1.7** (Sch '99, MS '10). 3-SAT  $\in$  TIME( $1.34^n$ ).

---

**Algorithm 1** randomized algorithm for 3-SAT

---

```
1: procedure WALKSAT( $\phi$ )
2:   for  $t = 1 \dots (4/3)^n$  do
3:     pick  $x \in \{0, 1\}^n$  at random
4:     for  $u = 1 \dots n^2$  do
5:       if  $x$  satisfies  $\phi$  then
6:         output YES
7:       else
8:         pick a random clause  $c$  not satisfied by  $x$ 
9:         flip  $x$ 's assignment on a random variable in  $c$ 
10:  output NO
```

---

**Theorem 1.8.** *If  $\phi$  is satisfiable, then  $\Pr(\text{algo outputs NO}) \rightarrow 0$  as  $n \rightarrow \infty$ .*

1. 4-SAT  $\in \text{TIME}(1.5^n \text{poly}(n))$
2. 5-SAT  $\in \text{TIME}(1.6^n \text{poly}(n))$
3. 6-SAT  $\in \text{TIME}(1.667^n \text{poly}(n))$
4. 7-SAT  $\in \text{TIME}(1.7143^n \text{poly}(n))$
5. ...
6. k-SAT  $\in \text{TIME}((2 - 2/k)^n \text{poly}(n))$ . In fact today's fastest algo is about  $O(2^{(1 - \pi^2/6k)n})$ .