15-455: UCT **K. Sutner**

Assignment 4 Due: **February 18**, 2022.

1. Easy SAT Versions (30)

Background

Here are a few "easy" variants of SAT. The first, 2−SAT, is in CNF and has two literals per clause. The next, HornSAT, is again in CNF, and this time each clause contains at most one unnegated variable (possibly none). Typical example:

$$\{\overline{x}, \overline{y}, z\}, \{\overline{y}, \overline{z}\}, \{x\}$$

It is more useful to write the clauses in implicational notation:

$$x \wedge y \Rightarrow z,\ y \wedge z \Rightarrow \bot,\ x$$

Lastly, XorSAT, is also based on a kind of CNF, but this time the clauses are required to be xors of literals. So you can write things such as

$$(x \oplus \overline{y} \oplus z) \wedge (\overline{x} \oplus \overline{z}) \wedge (x \oplus z)$$

Task

A. Show that 2−SAT can be solved in polynomial time.

B. Show that HornSAT can be solved in polynomial time.

C. Show that XorSAT can be solved in polynomial time.

Comment

You might wish to recall at this point that xor is associative and commutative.

2. Minimization Problems (30)

Background

A minimization problem consists of $\mathfrak{M} = \langle I, \mathsf{sol}, \mathsf{cost}\rangle$ where $I \subseteq \Sigma^\star$ is a collection of instances, $\mathsf{sol} : I \to \mathfrak{P}_0(\Sigma^\star)$ (codomain is all finite subsets of $\Sigma^\star$) is a solution function, and $\mathsf{cost} : \bigcup_{x\in I} \mathsf{sol}(x) \to \mathbb{N}$ is a cost function. Quite a few combinatorial problems can be interpreted as minimization problems. For example, Vertex Cover, I is the set of all undirected graphs, $\mathsf{sol}(x)$ is the collection of all vertex covers of graph x, and $\mathsf{cost}(w)$ is the cardinality of the vertex cover w.

We can associate three variants with $\mathfrak{M}$: the decision version $\mathfrak{M}_D$, the "counting" version $\mathfrak{M}_C$, and the search version $\mathfrak{M}_S$. We will ignore the function version, it requires some additional ordering on the solutions so we can talk about the "first solution."

Problem: $\mathfrak{M}_D$

Instance: $x \in I$, $k \in \mathbb{N}$.
Question: Is there a $w \in \mathsf{sol}(x)$ such that $\mathsf{cost}(w) \leq k$?

Problem: $\mathfrak{M}_C$
Instance: $x \in I$.
Solution: Determine the minimal cost of any $w \in \mathsf{sol}(x)$.

Problem: $\mathfrak{M}_S$
Instance: $x \in I$.
Question: Determine a $w \in \mathsf{sol}(x)$ of minimal cost.

Task

A. What does all this mean in the case of Independent Set and Clique?

B. Come up with one more example of a natural minimization problem.

C. We would like $\mathfrak{M}_D$, $\mathfrak{M}_C$ and $\mathfrak{M}_S$ to be polynomial time Turing equivalent, as is indeed the case of Vertex Cover. What conditions on $\mathfrak{M}$ must you impose that make this equivalence work?

Comment

Make sure your answer to part (C) covers standard examples like Vertex Cover, Independent Set and Clique. Try to be a bit more general, but don't worry about finding the most general conditions possible, it's fairly messy.

3. BDDs (40)

Background

By BDD we always mean ROBDD, using the standard variable ordering

$$x_1 < x_2 < \ldots < x_n < 0, 1$$

Assume the most basic standard implementation (two terminal nodes, no negation tricks, ...) We refer to the number of nodes in the BDD as its size.

Task

A. What is the size of the BDD for $x_1 \wedge x_2 \wedge \ldots \wedge x_n$.

B. What is the size of the BDD for $x_1 \oplus x_2 \oplus \ldots \oplus x_n$.

C. Assume n is even. What is the size of the BDD that checks

$$(x_1, x_3, \ldots, x_{n-1}) = (x_2, x_4, \ldots, x_n)$$

D. Let $n = 2k$. What happens when we are testing for

$$(x_1, x_2, \ldots, x_k) = (x_{k+1}, x_{k+2}, \ldots, x_n)$$

instead? No exact count needed, just explain the crucial difference to the previous part.

E. What is the size of the BDD that computes addition on two k-bit numbers? For simplicity, assume the output is also k-bit, we simply truncate if there is overflow. Here $n = 3k$ and the bits are grouped as follows:

$$\begin{aligned} x_1 x_4 \ldots x_{3k-2} &= u \\ x_2 x_5 \ldots x_{3k-1} &= v \\ x_3 x_6 \ldots x_{3k} &= w \end{aligned}$$

We want $u + v = w$, ignoring overflow.

Comment

For all of these, start with small values of n, and try to figure out what happens when n increases. If you cannot get an exact count, at least indicate the order of magnitude.