We use the notation $\mathcal{O}^*(f(n))$ to hide factors polynomial in the input length, e.g., $\mathcal{O}(2^n n^2) = \mathcal{O}^*(2^n)$.

# 1 Method: Branching

## 1.1 Vertex Cover

Recall the definition from day 1.

**Definition 1.** *A* vertex cover *of a graph $G = (V, E)$ is a set of vertices $X \subseteq V$ such that for every edge $e \in E$, $e = \{u, v\}$, at least one endpoint of $e$ lies in $X$ (that is, $u \in X$ or $v \in X$).*

| Vertex Cover |
|---|
| **Input:** Graph $G = (V, E)$ |
| **Question:** Find a minimum vertex cover of $G$ |

*Given in class:* Vertex Cover can be solved in time $\mathcal{O}^*(1.3803^n)$ (recurrence $T(n) = T(n-1) + T(n-4)$).

## 1.2 $k$-SAT

**Definition 2.** *A* $k$-CNF formula *is a boolean formula $\mathbb{F} = C_1 \wedge \ldots \wedge C_m$, where each $C_i$ is a* $k$-clause $(l_1 \vee \ldots \vee l'_k)$, *$k' \leq k$, and each $l_i$ is either $x$ or $\neg x$ for some variable $x$. (Example: $(a \vee b \vee c) \wedge (\neg a \vee \neg b) \wedge (\neg c \vee d)$ is a 3-CNF formula.)*

| 3-SAT |
|---|
| **Input:** A 3-CNF formula. |
| **Question:** Does the formula have a satisfying assignment? |

*Given in class:* $k$-SAT can be solved in time $\mathcal{O}^*(c_k^n)$ where $c_k < 2$ for every fixed $k$.

**Problem 1.** *(Repeated from pre-course exercises.) 2-SAT can be solved in polynomial time.*

## 1.3 Exact Hitting Set

| Exact Hitting Set |
|---|
| **Input:** A set system: A set $U$, and a set $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of $U$. |
| **Question:** Find a set $X \subseteq U$ which intersects every set $S_i \in \mathcal{S}$ *exactly* once (if one exists). |

**Problem 2.** *Solve* Exact Hitting Set *in time $\mathcal{O}^*(1.4656^n)$ (recurrence $T(n) = T(n-1) + T(n-3)$).*

**Remark 1.** *The* Hitting Set *problem (accidentally called* Set Cover *on the approximation problem set) seems to be much harder for exact algorithms than* Exact Hitting Set*; no algorithm with running time $\mathcal{O}^*(c^n)$ for $c < 2$ is known for* Hitting Set *(and some researchers conjecture that none exists).*

# 2 Method: Dynamic Programming

## 2.1 Subset Sum

| Subset Sum |
|---|
| **Input:** A set of integers $x_1, \ldots, x_n$; a target integer $T$. |
| **Question:** Is there a subset of the integers that sum to $T$? |

You may assume that the integers are non-negative (it makes no difference, but it might be easier to think about). In day 1, we saw that Subset Sum can be solved via dynamic programming in time $\mathcal{O}^*(T)$ (note that $T$ can be exponentially large in the size of the input, since writing down $T$ only takes $\log T$ bits). Here, we ask for a different direction.

**Problem 3.** *Solve* Subset Sum *in $\mathcal{O}^*(2^{n/2})$ time.*

## 2.2 Chromatic Number

**Definition 3.** *A* $k$-coloring *of a graph* $G = (V, E)$ *is a labeling* $f : V \to \{1, \ldots, k\}$ *of the vertices, such that for every edge* $\{u, v\} \in E$*, we have* $f(u) \neq f(v)$*.*

| Graph $k$-coloring |
|---|
| **Input:** A graph $G$, an integer $k$. |
| **Question:** Does $G$ have a $k$-coloring? |

**Problem 4.** *Solve* Graph $k$-coloring *in time* $\mathcal{O}^*(c^n)$*, preferrably* $c = 3$*. (Note that the size of the search space is* $k^n$*.)*

**Remark 2.** *Via a method known as* principle of inclusion-exclusion *it is possible to do this in time* $\mathcal{O}^*(2^n)$ *and exponential space, or in time* $\mathcal{O}^*(c^n)$*,* $2 < c < 3$*, with* polynomial *space.*