- **SAT**
  - Input: boolean formula f in CNF with n-variables and m-clauses
  - Output: Satisfying assignment S where each clause evaluates to True. Returns 'NO' when no such S exists.
- ***SAT Variations***
  - k-SAT is SAT with an additional input constraint k so that each clause has at most k literals. To be NP-C, k > 2
- **Independent-Set (Search)**
  - Input: Graph G = (V, E) and goal g
  - Output: Subset S in V if there are **no edges** between every vertex in S and |S|>=g. Returns 'NO' when no such S exists.
- **Clique (*Search*)**
  - Input: Undirected Graph G = (V, E) *and goal g*
  - Output: subset S in V where there **are edges** between every pair of vertices in S and |S|>=g. Returns 'NO' when no such S exists.
- **Vertex Cover (Search)**
  - Input: Undirected Graph G = (V, E) and budget, b
  - Output: Subset S in V, if every e=(x,y) has either x in S or y in S and |S|<=b. Returns 'NO' when no such S exists.
- **K-Colorings**
  - Input: Undirected Graph G=(V,E) & integer k > 0
  - Output: assign each vertex a color in {1, 2, ..., k} so that adjacent vertices get different colors and NO if no such k-coloring exists
  - *Note:* for k=2, K-Colorings∈P because it can be solved with modified BFS/DFS to find a bipartite graph
- **Knapsack (Search)**
  - Input: n-items with integer weights: w_1,...,w_n; integer values v_1,...,v+n; capacity B; and goal g
  - Output: subset S items where the total value V is >=g and total weight W is <= B. Returns 'NO' when no such S exists.
- **Subset Sum (Search)**
  - Input: n positive integers a_1, ..., a_n and a target, t
  - Output: Subset S of [1, ..., n] where the sum of a_i for all i in S = t, if such a subset exists. Returns 'NO' when no such S exists.

From Book:

Based on recommendations from course staff (https://edstem.org/us/courses/42665/discussion/3720834?answer=8572355), we should focus on the problems presented in lecture; however, the examples from the book may still prove helpful.

- **Traveling Salesman**
  - Input:

        - n vertices {1, 2, ..., n}
        - n(n-1)/2 pairwise distances between each vertex
        - A budget b for total distance traveled
    - Output: a path that visits every vertex with a total distance <= b. Returns 'NO' when no such path exists.
- **Rudrata Cycle**
    - Input: G = (V, E) (directed or undirected)
    - Output: a cycle c that visits every vertex exactly once. Returns 'NO' when no such S exists.
- **Balanced Cut**
    - Input: G = (V, E) and a budget b,
    - Output: partition the vertices into two sets S and T such that |S|,|T| >= n/3 and such that there are at most b edges between S and T. Returns 'NO' when no such S exists.

Solution Verification Runtimes

Note: additional runtimes can be added to:
https://edstem.org/us/courses/42665/discussion/3844271

- **SAT**: O(nm) to evaluate all literals within each clause
    - k-SAT: O(m) assuming k is a constant
- **Independent Set** (Search): O(n^2) to verify there are no edges between any vertices within S and O(n) to check |S| >= g
- **Clique** (Search): O(n^2) to verify that there are edges between every vertex in S and O(n) to check |S| >= g
- **Vertex Cover** (Search): O(n+m) to *TBD* and O(n) to check |S| <= b
- **K-Colorings**: For every edge, check that color assignment for each paired vertex is different O(m) (or O(n+m) to be safe)
- **Knapsack** (Search): *TBD*
- **Subset Sum** (Search): *TBD*