Polynomial time and reductions

1

Set Cover

Given a set D and n subsets $S_1, S_2, \cdots, S_n \subseteq D$. We want to choose k distinct indices $1 \leq a_1, a_2, \cdots a_k \leq n$ such that $S_{a_1} \cup S_{a_2} \cup \cdots \cup S_{a_k} = D$

- **Decisiton:** Given k, can we do it with at most k indices?
- **Evaluation:** What is the minimal k to satisfy?
- **Search:** What are the indices $a_1, a_2, \cdots a_k$ to satisfy the problem where k is minimal.

TSP

Given an undirected weighted complete graph of n vertices and a vertex u.

- **Decisiton:** Given k, can we find a hamiltonian circuit from vertex u with weight $\leq k$?
- **Evaluation:** What is the minimal weight of hamiltonian circuit from vertex u?
- **Search:** What is a hamiltonian circuit from vertex u with the minimal length?

0-1 Knapsack

Given a knapsack of capacity W and n items, with weights $w_1, w_2, \cdots, w_n$ and values $v_1, v_2, \cdots, v_n$. We want to choose k distinct indices $1 \leq a_1, a_2, \cdots, a_k \leq n$ such that $w_{a_1} + w_{a_2} + \cdots + w_{a_k} \leq W$. The total value of this knapsack is $V = v_{a_1} + v_{a_2} + \cdots + v_{a_k}$.

- **Decisiton:** Given v, can we do it so that the total value of the knapsack, $V \geq v$?
- **Evaluation:** What is the maximal V?
- **Search:** What are the indices $a_1, a_2, \cdots, a_k$ satisfying the constraint and giving us the maximal V?

2

We can solve Knapsack problem for our items with values $v_i = a_i$, weights $w_i = a_i$ and knapsack capacity $W = M$. Now if we can have a total value of (at least) M, then the answer for Subset-Sum problem is `true`. The answer is `false` otherwise.

3

If we have n subsets. First find the minimum number of sets that we need to cover, let's call it k_0. For each $1 \leq i \leq n$ let's remove S_i from our subsets and calculate the minimum number of sets needed to cover again, let's call it k_i, now if $k_i > k_0$ then S_i should be in our set cover, So we'll keep S_i as part of the answer, and remove elements of it from all the other subsets and also the set D. Now we repeat the process for the remaining $n-1$ subsets and the modified D until $D = \varnothing$. The answer to the search variant are those S_i's that we found during the process.

If the evaluation variant can be solved in $T(n)$, the search variant can be solved in $O(T(n) \cdot n^2)$. Because each step of the process is $O(n)$ and we repeat the process at most n times ($k_0 \leq n$ times).

Obviously if the evaluation solver gave us ∞, it means that the problem does not have an answer, and we stop immediately.

Pseudo-Code:

```
answer = []
SetCoverSearch(D, S[1..n]):
    k0 = SetCoverEvaluate(D, S[1..n])
    for i in 1..n:
        if SetCoverEvaluate(D, S[1..i-1, i+1..n]) > k0:
            answer.push(S[i])
            for j in 1..n where j != i:
                S[j] -= S[i]
            D -= S[i]
            SetCoverSearch(D, S[1..i-1, i+1..n])
            return
```