0.1 Set cover

- *Algorithm:* Greedy Approach (algo. 1)
- *Input:* A universe of elements $V = \{e_1, ..., e_n\}$, and a list of sets $\{S_i \subset V\}_{i=1}^m$
- *Complexity:* $\mathcal{O}(mn \cdot \min(n, m))$
- *Data structure compatibility:* set
- *Common applications:* facility location selection in industry

Problem. Set cover

Give A universe of elements $V = \{e_1, ..., e_n\}$, and a list of sets $\{S_i \subset V\}_{i=1}^m$, we want to cover all the elements in universe V with minimal number of sets from the set list.

Description

Detailed definition of the problem. Assume we have a universe of elements $V = \{e_1, ..., e_n\}$, we assume there are a list of non-empty sets $\{S_i \subset V\}_{i=1}^m$, we would like to find out a number of sets $\{T_i\}$ out of the universe V, such that the universe V is equal to the union of them, namely $V = \cup_{i=1}^j T_i$. There are many variations of this problem, we can also set cost to each set(a specific number rather than 1). However, the idea does not change much, we only need to modify the greedy algorithm a little bit. Set cover problem is also related to max coverage and vertex coverage problems.

Set cover is a classic NP problem in the field of mathematics and operations research. Unless we can prove $N = NP$, we can not derive an algorithm that can find the exact solution to the problem within polynomial time[1]. So a natural trade-off will lie between time complexity and accuracy. In our case, the time complexity is more critical. To deal with the time complexity, we have to sacrifice the accuracy and use approximate algorithm.

Another perspective is the usage of greedy algorithm. In algorithm design, the big picture is that greedy algorithm, local search, and LP-relaxation are three main strategies that deal with most NP-hard problems[2]. So brute force these three methods before using fancy technique. To enjoy more about approximate algorithms, read *the Design of Approximate Algorithms*[3] and you will gain a lot.

The key idea in this algorithm is to be greedy, trying to choose the set that covers largest number of elements. Here we assume $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$, and we can tell $H_n \approx \ln n$, which is taught in basic calculus course. Our algorithm is an H_n- approximate algorithm that solves the set cover problem. The detailed proof is beyond the scope of this course(sadly engineers does not share the beauty of mathematics).

Also we have to consider the time complexity for our algorithm. For this simple version of greedy algorithm. We can eliminate at least one element from the set U or we can take it as one set from the set list $\{S_i \subset V\}_{i=1}^m$, so there are at most $\min(n, m)$ loops for the while(not less because of the case that each set S_i has exactly one element). Within each loop, there are $\mathcal{O}(mn)$ operations for the argmax operation. So the total time complexity is $\mathcal{O}(mn \cdot \min(n, m))$. In the exercise problem in Introduction to Algorithm, we can modify the algorithm to make it linear for argmax.

Algorithm 1: Greedy Approach

Input : A universe of elements $V = \{e_1, ..., e_n\}$, a list of sets $\{S_i \subset V\}_{i=1}^{m}$
Output: minimal number of sets $\{T_j\}$

Assume U is the set of uncovered elements.
Set $U = \{e_1, ..., e_n\}$.
while $U \neq \emptyset$ **do**
    pick $T_j = argmax_{j=1,...,m} \mid S_j \cap U \mid$
    U $\leftarrow U \setminus T_j$
end while
return

References.

[1] Manuel. *VE477 – Introduction to Algorithms (lecture slides)*. 2020 (cit. on p. 1).

[2] Cong Shi. *VG441 – Supply Chain (lecture slides)*. 2020 (cit. on p. 1).

[3] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011 (cit. on p. 1).