Algorithms Analysis and Design

Week 5 - Diary

Ayan Agrawal (2020101034)

Lecture 8 : Set Cover and Approximation Algorithms

Problems for the class :

In Lecture 8, we progressed our discussion on the Greedy algorithms towards how to realize an approximate solution to NP-complete problem (close to optimal solution by a factor of $logn$).

Points discussed were :

- Set cover problem
- Reason Greedy method fails in this problem
- How close greedy method has came to the optimal solution.

1. Set - Cover Problem

Problem statement :

Let's try to understand the problem statement for Set cover problem:

Input : Set B containing elements and a set of subsets $S_1, S_2, \dots S_n$ in B.

Output : Selected S_i such that union of all S_i is set B.

Work : Number of Sets selected.

- This is an NP- complete problem meaning there is no polynomial time solution that this problem (that is know algorithms are exponential in n) but we can check the solution in Polynomial time .
- Because of the nature of this problem(NP-complete) , it becomes important to study this problem because getting a polynomial time solution to this problem would mean we can solve all NP-Complete problems and this can be one of the biggest revolutions in human history.

Here, Minimum set cover is defined as minimum cost incurred, now let's see different ways to solve it,

1. Naive Solution

- It will involve most basic solution of checking all 2^n subsets of S_i, and see which all will give B on union. We can store cost incurred along with it and see the minimum cost for the same.
- This would give the most appropriate solution but we can't always perform such an expensive method (exponential solution), so it is better to find a less expensive method which gives accurate or near to accurate solution.

2. Greedy Approach

- As seen previously, if we can prove that our first step in a Greedy algorithm is always bound to be correct, we can recurse this step and by the **optimum substructure property**, we would get the correct result always.
- In this problem, we see that the greedy algorithm does not work as determining the first step itself is inaccurate/ time consuming.Though we can try to take a first step which is partially correct, and for all practical purposes, it would give us a near accurate result if not the best in very less time.

- For this, Let's use approximation algo theory, a greedy approach for this problem is as follows :

Pick Set S_i with largest number of non-included elements in set B.

But, this solution isn't correct always, let's take a counter example,

Suppose $B = \{1, 2, 3, 4, 5, 6\}, S_1 = \{1, 2, 3, 4\}, S_2 = \{1, 3, 5\}, S_3 = \{2, 4, 6\}$.

We will choose S_1 then S_2, S_3 in any order but work done will be 3.

However, choosing just S_2, S_3 will just select B and do the work of 2.

Therefore, Greedy method fails and we determine that the set cover problem is a NP-complete problem. Lets try to understand just how accurate is the greedy strategy is with respect to the actual solution.

Accuracy of the Greedy Strategy :

Claim:

Suppose B contains n elements and the optimal set contains k subsets. Greedy algorithm will use at most $kln(n)$ sets.

Proof :

Let's say we have n_i elements left after i iterations of the algorithm.
So, $n_0 = n$
One key observation in the proof is that at an iteration i , if we have n_i elements left, then as there are k sets at best. One set should at least have n_i/kelements in it by Pigeon hole principle.

Therefore, inequality acquired is

$$n_{i+1} \leq n_i - n_i/k$$

Now, recursing over i from 0 to t, we get

$$n_i \leq n_0(1 - 1/k)^t.$$

Also,

$$e^{-x} \geq 1 - x.$$

Thus, final inequality is :

$$n_i \leq n(e^{-i/k}).$$

When we hit a point where n_i is lesser than 1 , we are sure to have found a solution. Therefore, at $i = kln(n)$, the RHS approaches 1 . At every iteration we have assigned one set and thus total sets would be equal to total iterations. Thus we get a bound of $kln(n)$.

So, we call this an $O(ln(n))$ approximate solution.