Week 5 - lecture 8

Introduction:

In this lecture , we extended our discussion on the Greedy Algorithms and tried to see how we can realise an approximate solution to a NP-complete problem which is close to the optimal solution by a factor of log n. So we discussed the following ideas.

- What is the set-cover problem.
- Why Greedy algorithm fails in this problem.
- How close is our Greedy solution to the optimal solution.

Set-Cover problem:

The problem states that we are given the elements of some set S.

In addition to that , we are also given m subsets of the set S say S_1,S_2,S_3 , etc . We need to choose minimum no of subsets from these m sets so that their union includes all the elements of set S.

n → Cardinality of set S

So formally we need to minimise the cost which is equal to the total number of subsets picked or tell if it's not possible.

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.

Naive solution:

We can try all the 2^m subsets of the given sets and find the one which covers all the elements of S and has the minimum no of such subsets .

But the time complexity of this solution is exponential and even for a small input size of 100 , the time to compute the solution would be more than the age of the universe , therefore it becomes extremely necessary that we come up with better solutions to this problem .

Greedy Approach :

The Greedy approach can be explained in merely two lines:

- Take an empty set S' . Now while S' cardinality is not equal to S , repeat the following →
- Pick the subset with maximum elements all remaining subsets and take the union of the elements of the set with S'.
- Remove the chosen subset from the set of given subsets and repeat.

So in the greedy approach , we always take the first step as one in which we pick the set with the maximum elements but this is not accurate as we will see later . So this is an approximate solution and not a correct solution .

Why Greedy fails:

We can prove the Greedy strategy is not correct by giving a counter example :

Say S = { 1, 2,3,4,5,6} so n =6 and also 3 subsets are given →

S_1 → {1,2,3,4}

S_2 → {1,2,5}

S_3 → {3,4,6}

So following the Greedy strategy we would pick all three subsets { S_1 followed by S_2 and at last S_3 since S_1 has the maximum size} but we can clearly observe that it is optimal to pick just 2 subsets S_2 and S_3 since their union is indeed S and we are using only two subsets.

Hence Greedy strategy is just an accurate solution and not fully correct.

Now let us see how accurate our Greedy Strategy is :

How close is our Greedy solution to optimal solution :

Let us first make a statement and then try to prove it :

> 💡 **Suppose B contains n elements and that the optimal set contains k sets . Then Greedy algorithm will use at most $kln(n)$ sets .**

Proof:

Let us assume that the optimal solution would require k sets . Now

suppose we will be left with n_t elements after some t iterations of our Greedy Algorithm .

So $n_0 = n$ since at the beginning we have all the n elements to cover as we have not chosen any subsets.

Now we can observe that after any t 'th iteration , we have n_t elements left and best solution has k sets . So by pigeon hole principle , the maximum set of the remaining sets must contain at least n_i/k elements in it .

So we would get the following inequality :

$n_{i+1} <= n_i - n_i/k$

We can write this as:

$n_{i+1} <= n_i(1 - 1/k)$

Now we can recurse over all values of i from 0 to t and get the following inequality :

$n_t <= n_0(1 - 1/k)^t$

We can also prove using calculus that $x^{-i} >= 1 - x$.

So we can rewrite the equation as :

$n_t <= n(e^{-t/k})$

We need to reach the point where n_t is smaller than 1 since that would ensure that we have picked all n elements and have got our required solution . So we can observe that at $t = kln(n)$, the value of n_t is equal to 1 and after that it would become less than 1 . Hence we have proved that $kln(n)$ is the upper bound for the number of iterations if the optimal solution has k sets .

So our Greedy approach is close to the optimal solution by a factor of $ln(n)$.

So in this lecture we say how Greedy approach can help us get a somewhat close solution to an otherwise unsolvable problem!