Week 5, Lecture 1

Algorithm Analysis and Design

Pratyay Suvarnapathaki, 2020111016

Set Cover Problem

The problem statement is quite simple. Given a set B and its subsets $s_1, \cdots, s_m$, the subsets whose union covers B are to be found. Here, the cost of the solution is determined by the number of subsets chosen. Naturally, lesser the cost, the better is the solution.

The Greedy Solution

Intuitively, this problem seems to be solvable if we take the greedy approach. Applying the basic principles of greedy design:

- **The greedy choice property** Can we know for sure what a good first-step towards the solution is? Well yes, it *seems* like the set which by itself covers the largest position of B would be a good pick.
- **The optimal substructure property** Can we now re-state the remainder of the problem post the first step? i.e. After ascertaining the requisite first set, can we find the next 'best' set from among the remaining subsets? Well yes, it would naturally seem like the next best pick would be the subset that covers the most of the remainder of B, post choosing the first set.

To summarise the greedy solution: **'Until all elements of B are covered, keep picking the S_i which contains the maximum number of yet-uncovered elements of B.'**
Example:
$B = \{a, d, e, h, i, l, n, o, r, s, t, u\}$
Subsets = `arid, dash, drain, heard, lost, nose, shun, slate, snare, thread, lid, roast`
(Note: the subsets have not been written in proper set notation, but rather as words for the sake of readability.)

Pick_Num	Picked_Subset	Yet_Uncovered
1	`thread`	$\{i, l, n, o, s, u\}$
2	`lost`	$\{i, n, u\}$
3	`drain`	$\{u\}$
4	`shun`	$\varnothing$

Cost=4. This solution is clearly optimal.
However, surprisingly, the greedy approach may **not** result in the optimal solution every time.
Example:
$B = a, b, c, d, e, f$, Subsets= `abcd,ace,bdf`
The greedy approach first picks `abcd`, then `ace` and `bdf` resulting in a total cost of 3.
However, clearly, the optimal solution should cost just 2, and comprises of only `ace` and `bdf`.

No Cigar, But Still Quite Close

As demonstrated above, the greedy solution will not *always* result in the optimal solution.
However, it does get pretty close. To be precise, the greedy solution is $O(lnn)$-approximate.
i.e.
Claim: If $|B| = n$ and the optimal solution costs k, then the greedy solution costs at most $kln(n)$
Proof:
Let n_t be the number of subsets not yet covered by the greedy solution after iteration t.
Now, since the k optimal sets cover all the remaining elements, by the pigeonhole principle, there should be atleast one set with atleast n_t/k of them. Thus, due to the greedy approach, we can state the following about the next iteration:
$n_{t+1} \leq n_t - \frac{n_t}{k} = n_t\left(1 - \frac{1}{k}\right) \implies n_t \leq n_0(1 - 1/k)^t$ ($\because n_0 = n$, by the definition of n_t)
Now,
Using the inequality: $1 - x \leq e^{-x}$, equality at $x = 0$
We have:
$n_t \leq n_0\left(1 - \frac{1}{k}\right)^t < n_0\left(e^{-1/k}\right)^t = ne^{-t/k}$
Thus,
at $t = kln(n)$, $n_t < ne^{-ln(n)} = 1$, $\implies$ all elements have been covered.
Hence, proved.

Thus, this maximum factor $ln(n)$ is called the **approximation factor** of our greedy algorithm.

Now that we have established that the greedy approach is not optimal, the natural question is: *can we do better?*
Well, the answer, surprisingly, seems to be **'no'**. Under certain usual assumptions, there exists no polynomial time algorithm with a provably smaller approximation factor.
