WEEK 5, LECTURE 8 ON 15 SEPTEMBER 2021 CS1.301.M21 ALGORITHM ANALYSIS AND DESIGN

REVIEW

- *Minimum spanning tree* problem showed the overarching method of writing greedy algorithms.
- Find a local property that gives a local answer
- Use induction to extend it into a global solution
- With the *activity selection* problem, we selected the activity with the least finish time and that gave a local optimum since it will always be a part of the final solution
- Huffman codes also were examined

SET COVER PROBLEM

Input: A set of elements B; and sets $S_1, \ldots S_m \subseteq B$

Output: A selection of the S_i sets whose union is B.

Cost: Minimize the Number of sets picked

Example

Consider

$$\{\text{arid}, \text{dash}, \text{drain}, \text{heard}, \text{lost}, \text{nose}, \text{shun}, \text{slate}, \text{snare}, \text{thread}, \text{lid}, \text{roast}\}$$

to cover the set $B =$

$$\{a, d, e, h, i, l, n, o, r, s, t, u\}$$

> Set cover problem is a problem that pops up in many places, it is often the underlying sub problem in many other problems.
>
> If you solve set cover, we can also solve other problems like *protein folding*.

A possible greedy algorithm

```
Repeat until all elements of B are covered:
    Pick the set S[i] with the highest number of elements that are yet to be covered.
```

First S_i selected will be of the set with the highest cardinality i.e. in our case thread

> note that in our case, we are using words that can have repeated elements, but sets normally will not have non-unique elements

Then subsequently follow the algorithm.

Analyzing the example

shun is a must pick since it is the only set with u.

- Uncovered: {a,d,e,i,l,o,r,t}
- At least 3 more picks since e & i aren't together and picking arid/drain or lid (for covering i) leaves {e,l,o,t} or {a,e,o,r,t} requiring two more picks!

Counter-Example

Suppose $B = \{1, 2, 3, 4, 5, 6\}$

- Set Family: {1,2,3,4}, {1,3,5}, {2,4,6}
- Greedy Solution picks all three
- Optimum Solution: {1,3,5} and {2,4,6}

So, the question remains whether the greedy solution compares to the optimum solution

It has been shown that *set cover* is NP complete.

Greedy sol is O(ln n)

Claim: Suppose $|B| = n$ and the optimal cover has k sets. Then the greedy algorithm will use at most $k \ln n$ sets.

Proof:

Let n_t be the number of uncovered elements after t iterations

So $n_0 = n, n_1 < n, \ldots$

Our greedy algorithm stops at iteration t when $n_t < 1$

Some sets of the k sets can cover all of the n_t elements (since they cover all the elements in fact). And by the Pigeonhole Principle, we can say that there is one of the k sets that cover at least $\frac{n_t}{k}$ of the n_t elements.

So,

$$n_{t+1} \leq n_t - \frac{n_t}{k} = n_t(1 - \frac{1}{k})$$
$$\therefore n_t \leq n_0(1 - \frac{1}{k})^t$$
$$\text{it is known that,} 1 - x \leq e^{-x} \text{ equal only if } x = 0$$
$$n_0(1 - \frac{1}{k})^t < n_0(e^{-1/k})^t = ne^{-t/k}$$
$$\text{at } t = k \ln n, n_t < ne^{-\ln n} = 1$$

> here greedy algorithm isn't always optimal, but we actually have a bound on the cost of the solution of the greedy algorithm, so it becomes better than whatever heuristic we can come up with