6651 Comments on lecture 3

Yaqiao Li

Janunary to April, 2023

You don't have to do all of the following, these are NOT assignments. But doing some of them MAYBE, hopefully, helpful to you. Also, I encourage you working together with your classmates. Try brainstorming, it will be more fun.

1. In class, for fractional knapsack problem, we order the items according to value per unit weight $X[i]/W[i]$, and then successively greedily add the item (or a fraction of the item if only some fraction fits the knapsack) with the largest value per unit weight. This algorithm, call it $\mathcal{A}$, is optimal.

   Consider the following greedy strategy $\mathcal{A}'$: simply successively greedily add the item (or a fraction of the item if only some fraction fits the knapsack) with the largest value $X[i]$ into your knapsack. Create some examples on your own, try this greedy algorithm on your examples. Then run $\mathcal{A}$ to get optimal solutions on your examples. Compare the results. Does $\mathcal{A}'$ give you optimal solutions? If in your example, the solution of $\mathcal{A}'$ is not optimal, then try to create one example for which $\mathcal{A}'$ is optimal, i.e., $\mathcal{A}'$ give the same solution as $\mathcal{A}$.

2. We gave several different greedy strategies for the coding problem. Can you give some other greedy strategy? It does not need to be optimal, as long as it is "greedy".

3. For the interval scheduling problem, give one greedy strategy different from the ones discussed in class, then create some examples to verify your strategy. Is your greedy algorithm always optimal?

4. We've introduced the Horn formula satisfiability problem (check the slides if you haven't fully understood the problem), but we haven't discussed any greedy algorithm to solve it. DO NOT LOOK UP THE SOLUTION IN SLIDES. TRY come out of at least two different greedy strategies for this problem. Compare your algorithms on some examples and check their performances. What are their time complexities? Do they always output correct answers?

5. In general, what do you feel about greedy strategy? Are they simple to understand, or not? Is it easy to implement, or not? Are they efficient, or not? Are they always optimal, or not? Do you feel it's easy to come out of *some* (not necessarily optimal) greedy algorithm for a problem? Do you feel it's easy to come out of the *optimal* greedy algorithm for a problem?

6. Try design a greedy algorithm (need not be optimal) for the MaxSubarray problem. Your algorithm should have the spirit of being "greedy", but don't care about whether it really outputs the maximum subarray. Now, run your greedy algorithm on the example in Lecture 2, or on some examples you create by your own. What is the time complexity of your algorithm? Does it output the maximum subarray? Compare and discuss your solutions with classmates.

7. Try come out of some divide-and-conquer algorithm for the coding problem. Run your algorithm on the example in lecture slides. Discuss and compare your algorithms with classmates (e.g., the time complexity? is it simple to understand? Is the algorithm intuitive? Does it always output optimal solution? . Are there any difficulties? What are they? etc).