CISC 610-90- O-2018/Late Fall
Assignment 4

a) Compare

Divide and Conquer, Dynamic Programming, and Greedy Approach are all optimization design approaches. They are similar only in the way that they usually divide the problem construction into several stages or sub-problems, but different in optimization purposes, structures, efficiencies, and information needed, etc.

Comparison

Divide and Conquer approach breaks a problem into smaller sub-problems, and we may need to keep dividing the sub-problems into even smaller sub-problems. Until no more division is possible, the smallest sub-problems will be solved individually, most of the time recursively, and the individual solutions will be merged back to make the final solution to the original problem. Ideally, the sub-problems need to be a scaled-down version of the original problem, easier to be solved, and not related to each other, otherwise, the Divide and Conquer approach may not be the best choice. Standard algorithms that are Divide and Conquer algorithms include: Binary Search, Quicksort, Merge Sort, Closest Pair of Points, and Cooley-Tukey Fast Fourier Transform.

When the sub-problems are not solely independent, but over-lapped to each other, the Divide and Conquer approach is not efficient to be used. When the sub-problems are not exactly the same, that means there's one best combination of the sub-problems, and we need to go through all possible combinations to find the best one. instead of using Divide and Conquer approach, computing different sub-problems again and again which is very inefficient, rather, we prefer to remember the results in memory for sub-problems and analyze the results at a higher level. This kind of optimization approach is called Dynamic Programming. This algorithm is best for the optimization problems that can be divided into smaller over-lapped similar sub-problems. Shortest Path Problem, and many String Algorithms (longest common sub-sequence, longest increasing sub-sequence, the longest common sub-string) are typical uses of Dynamic Programming approach.

Like the Divide and Conquer and Dynamic Programming approaches, Greedy algorithm also builds up a solution piece by piece. However, it always chooses the next piece that has to be the most immediate benefit at the moment, without, or not able to consider future steps. The greedy approach is local optimization, without global optimization guarantee. It is best for this kind of problem: we can make a locally best choice at each step, and these choices eventually will lead to the global optimization solution. The advantage is: if one problem can be solved by Greedy approach, it is almost the most efficient approach, not as generally applied as Divide and Conquer or Dynamic Programming approaches of course, but highly efficient. Famous applications include Kruskal's Minimum Spanning Tree, Prim's Minimum Spanning Tree, Dijkstra's Shortest Path, and Huffman Coding.

As a conclusion, we compare the differences and similarities between Divide and Conquer Dynamic Programming, and Greedy approaches in Table. 1.

	Divide and Conquer	Dynamic Programming	Greedy
Optimization level	Global	Global	Local
Structure	Top-down	Bottom-up	Locally only
Most efficient in	Independent sub-problems	Overlapped sub-problems	Local optimization leads to global optimization
Typical form	Recursive	Iterative	Iterative

Table 1: Compare Divide and Conquer, Dynamic Programming, and Greedy approaches

Examples

Divide and Conquer

Eating Pizzas could be one best example for Divide and Conquer approach in daily life. We usually divide a pizza into pieces, and each people in the party finishes one piece. These pieces are obviously independent. Even if one cannot finish his/her piece, he/she can divide the piece into even smaller pieces to share, but not ask you to re-bite.

For computing example, Quick Sort is a good candidate. In Quick Sort, we choose a pivot in the unsorted list, then divide and sort each side recursively. The sort in sub-problems is completed individually and doesn't need to resort in other sub-problems.

Dynamic Programming

Chess is good for dynamic programming. No same strategy can be applied to all steps (comparing to Divide and Conquer), and you need to consider as many steps as possible to make the current decision. Be prepare to regret if you apply Greedy approach in chess.

As computing example, I think the shortest path is the best. Assuming we drive from one city to another. The two cities are the nodes in the graph, the edges connecting the nodes are the distances between each node. All possibilities can be visited and sued to find the shortest path.

Greedy

Doing homework in Dr. Renata McFadden's class should always follow the Greedy approach. That is, I should always try to get a full mark in each assignment, as I don't know how difficult the future assignments are, or how much time I will have. Until I already got enough marks for A, I can save some time on homework, or dismiss the extra assignment.

For computing example, packing should be easy to be understood one. When trying to save all items into as few boxes as possible, we always try to fill the largest box, then way down to the smallest.