Dynamic Programming Optimization

Date	16 November 2022
Team ID	PNT2022TMID17187
Project Name	IoT Based Safety Gadget for Child Safety Monitoring and Notification

Dynamic Programming:

- We saw that greedy algorithms are efficient solutions to certain optimization problems. However, there are optimization problems for which no greedy algorithm exists. In this chapter, we will examine a more general technique, known as dynamic programming, for solving optimization problems.
- Dynamic programming is a technique of implementing a top-down solution using bottom-up computation. We have already seen several examples of how top-down solutions can be implemented bottom-up. Dynamic programming extends this idea by saving the results of many sub-problems in order to solve the desired problem.
- As a result, dynamic programming algorithms tend to be more costly, in terms of both time and space, than greedy algorithms. On the other hand, they are often much more efficient than straightforward recursive implementations of the top-down solution

Making Change:

- Suppose we wish to produce a specific value $n \in N$ from a given set of coin denominations $d1 < d2 < \cdots < dk$, each of which is a positive integer. Our goal is to achieve a value of exactly n using a minimum number of coins.
- To ensure that it is always possible to achieve a value of exactly n, we assume that $d1 = 1$ and that we have as many coins in each denomination as we need.

OPTIMIZATION II: DYNAMIC PROGRAMMING:

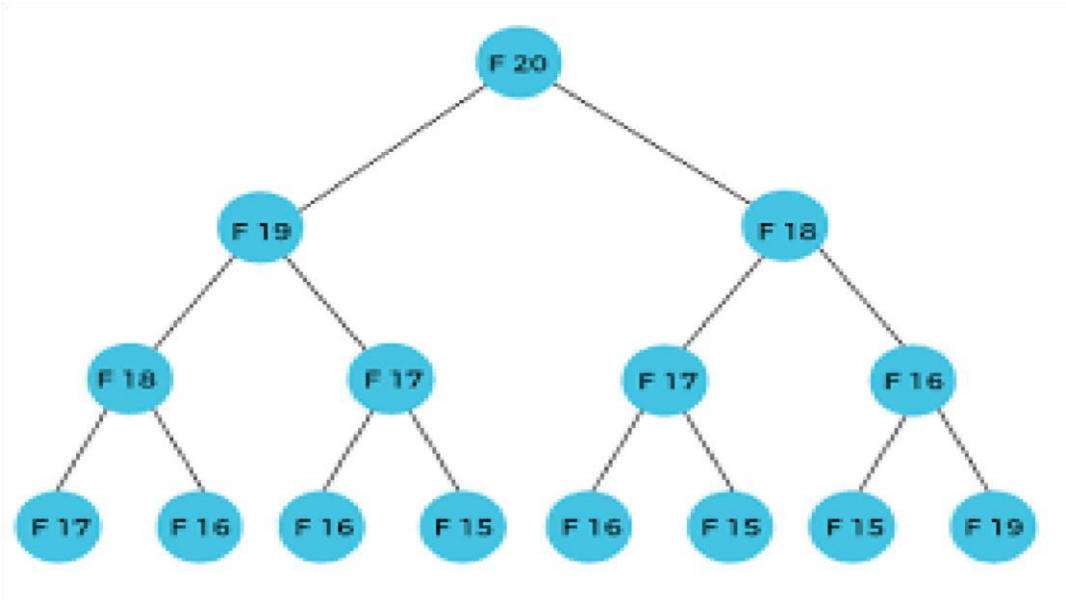
- An obvious greedy strategy is to choose at each step the largest coin that does not cause the total to exceed n. For some sets of coin denominations, this strategy will result in the minimum number of coins for any n.
- However, suppose $n = 30$, $d1 = 1$, $d2 = 10$, and $d3 = 25$.
  The greedy strategy first takes 25.
- At this point, the only denomination that does not cause the total to exceed n is 1. The greedy strategy therefore gives a total of six coins: one 25 and five 1s. This solution is not optimal, however, as we can produce 30 with three 10s.
- Let us consider a more direct top-down solution. If $k = 1$, then $dk = 1$, so the only solution contains n coins. Otherwise, if $dk > n$, we can

reduce the size of the problem by removing dk from the set of denominations, and the solution to the resulting problem is the solution to the original problem.

Finally, suppose dk ≤ n. There are now two possibilities: the optimal solution either contains dk or it does not.

All-Pairs Shortest Paths:

- We discussed the single-source shortest paths problem for directed graphs. In this section, we generalize the problem to all pairs of vertices; i.e., we wish to find, for each pair of vertices u and v, a shortest path from u to v.
- An obvious solution is to apply Dijkstra's algorithm n times, each time using a different vertex as the source



Dynamic programming approach work:

- It breaks down the complex problem into simpler subproblems.
- It finds the optimal solution to these sub-problems.
- It stores the results of subproblems (memoization). The process of storing the results of subproblems is known as memorization.
- It reuses them so that same sub-problem is calculated more than once.
- Finally, calculate the result of the complex problem.