Furkan Ayas

Lecturer / Research Supervisor : Zerrin Işık

15 May 2019

Optimal Solution Algoritms

Optimal Solutions Algorithms: Report Outline

1 - What is the Greedy Algorithm?

Formal: A **greedy algorithm** is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage[1] with the intent of finding a global optimum. In many problems, a greedy strategy does not usually produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time. (https://en.wikipedia.org/wiki/Greedy_algorithm)

Without considering the consequences that may arise in the future, the approach which best advocates that the best choice should be made is called greedy algorithm. The reason for this Greed name may be the search for a short-term solution. The basic principle is the idea that the solution in the local (at that step) can work globally (sometimes). Optimization is to choose the best among possible solutions. The greedy algorithm is used to solve optimization problems.

In the greedy algorithm, which is a problem-solving approach, an optimal solution is found at each step. Such an approach does not always give a definitive solution, but because it is simple, it does not require complicated calculations and at least it solves the problem in a certain time.

-Greedy algorithm chooses the best-looking algorithm at the moment (local best)
-Selects the best algorithm that is local (at every step), assuming that this choice will be the best solution globally
-Even if it doesn't always give the best solution, sometimes it can
-It's a simple algorithm and gives results at a certain time

2 - What is the Dynamic Programming?

Formal: **Dynamic programming** is both a mathematical optimization method and a computer programming method. The method was developed by Richard Bellman in the 1950s and has found applications in numerous fields, from aerospace engineering to economics. In both contexts it refers to simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner. While some decision problems cannot be taken apart this way, decisions that span several points in time do often break apart recursively. Likewise, in computer science, if a problem can be solved optimally by breaking it into sub-problems and then recursively finding the optimal solutions to the sub-problems, then it is said to have optimal substructure.
Bir problem tahlil ve çözüm yöntemi olan dinamik programlama yapı olarak parçala fethet yöntemine benzer. Tek farkı problemi parçalara böldükten sonra aynı problemin tekrarı olan parçaları bir kerede çözüp her tekrar için ayrı bir çözüm yapmamasıdır (https://en.wikipedia.org/wiki/Dynamic_programming)

The approach that records the results obtained in the previous steps and uses these records to obtain new results is called dynamic programming (dynamic programming).

At Computer Science area, there is different algorithms and some of them has long running times, these called Expensive functions. We store that types algorithm's data for next computations for make operation faster. This operation needs extra storage and we call this technique, Caching or Memoization.

Dynamic programming is based on recursion. It store every result at every recursion step. It merge sub problems results for solving whole problem. It using for optimization problems. It finds best optimal result.

It has 4 steps:
-Characterize the structure of an optimal solution
-Define optimal solution recursively
-Calculates the most appropriate solution as a bottom-top
-Creates an optimal solution from calculated information

3 - How we can perform reading the external file?

Firstly; convert to .xlsx file to .txt file for make reading operation more easy. Then at the code, we called the split method and use “(space)”for operator. Table has various types data so at the first step we create different types arrays and we make type parsing than we add this data at the new players. On the last step add this last player to our player ArrayList. With FileReader and BufferedReader I/O packages.

4 - Greedy & DP

Every Computer Science area problems has different solutions at it always makes dilemmas. For instance; in sorting algorithms, you must decide between speed and extra storage (cost). At this point if you choose Greedy your algorithm will be more simple and more writable also readable but it always doesn’t give perfect solution. On the other hand you can choose DP approach, it gives you perfect solution but it coding process is more harder and longer. It also needs more storage area for keep before step’s solution.

Greedy	Dynamic programming
-Make a choice at each step.	-Make a choice at each step.
-Make the choice before	-Choice depends on knowing optimal solutions to subproblems.
-Solving the subproblems.	-Solve subproblems first.
-Solve top-down.	- Solve bottom-up.

Runtime Complexity:
Greedy Θ(n)

LCS Θ(mn)   Brute Force Θ(n2m).
Dynamic programming doesn't have a time complexity, because it is not a specific algorithm. (https://www.quora.com/What-is-the-time-complexity-of-dynamic-programming)

References

https://www.quora.com/What-is-the-time-complexity-of-dynamic-programming

https://en.wikipedia.org/wiki/Dynamic_programming

https://en.wikipedia.org/wiki/Greedy_algorithm

http://bilgioloji.com/pages/yazilim/kod/program/algoritma/tur/dinamik-programlama-dynamic-programming-yontemi-nedir/

http://bilgioloji.com/pages/yazilim/kod/program/algoritma/tur/acgozlu-algoritma-greedy-algorithm-nedir/

https://stackoverflow.com/questions/7707586/what-is-the-best-complexity-of-a-greedy-algorithm