THE PLACE OF THE GREEDY METHOD CONCEPT IN THE PROGRESSION OF CONTESTANTS' THINKING

ERDŐSNÉ NÉMETH, Ágnes – ZSAKÓ, László, HU

Abstract: A didactically interesting question is how problem solving can be developed in children's mind while teaching CT for all or CS for competitors and what steps and tasks lead through from the understanding of the idea to its professional usage. In this paper we make an overview of how and in what forms the greedy algorithm – as a special problem-solving strategy – appears in children's informatics studies: from CS unplugged activities through Bebras tasks and national CS competitions to efficient coding at the IOI.

Key words: greedy concept, teaching informatics in primary and secondary schools, preparing for contests, greedy algorithm

1 Overview

"*Computer scientists are lazy and smart, which is a great combination. They learn a bag of tricks, and whenever they encounter a problem, they try to apply one* [or combine more] *of them.*" [Bebras 2015, Beaver Logs problem's solution] This idea makes the special problem-solving concepts important whether we teach computational thinking (CT) for all or computer science (CS) for competitors. In this article the place of greedy idea is examined in the development of children's mind, especially in CS studies.

A greedy concept is one of the most useful and widely used special problem-solving paradigms, which occurs naturally in common problems and it can be used many times at various levels of mathematics and computer science too. The "greedy" name came from the wildlife and the nature where it is very effective usually. Greedy algorithms try to find locally optimal solution which may eventually land in the global optimal solution. But in many cases greedy algorithms do not provide a globally optimized solution, so they have to be used carefully.

For the successful use of greedy algorithms in CS, the problem needs to have two properties:

- Existence of an optimal sub-structure: the optimal solution of the whole problem contains the optimal solutions of the sub-problems.
- The greedy property: if you make a choice that seems like the best at the moment and proceed to solve the remaining sub-problems, you reach the optimal solution for the whole problem. This way you will never have to reconsider your previous choices.

If both of the abovementioned conditions stand then it can be proven that using the greedy strategy will lead to an optimal solution.

It is important that children should be familiar with this useful and powerful concept from the beginning of their CT and CS studies, learning its usefulness and limitations.

2 Place of greedy method in algorithms textbooks

There are a lot of textbooks about algorithms written primarily for university students. These textbooks are useless for primary and secondary school pupils because of the lack of advanced mathematics and informatics knowledge. However, some parts of them may be useful in upper secondary, exclusively for contestants preparing for IOI. It is interesting to

examine where the place of the greedy concept is in the structure of algorithms because it provides general direction where the place of the greedy method should be in the line of teaching certain algorithms at lower level studies.

There are eight classical problems in all of the listed textbooks which can be solved by greedy algorithms: Coin Change (the special case), Load Balancing, Interval Covering, Minimum Spanning Tree, Single-Source Shortest Paths, Huffman Coding, Fractional Knapsack and some Job Scheduling problems. Two more problems are in most of the textbooks: Optimal Merging and Topological Sort. The best-known special greedy algorithms appear in every textbook: red-blue, Kruskal's, Prim's and Dijkstra's algorithm. The greedy concept and the special algorithms are: only in the chapter about graphs [Skiena, 2008; Sedgewick, 2011]; in a separate chapter as fundamental idea [Kleinberg, 2005]; or both [Cormen, 2001; Dasgupta, 2006; Halim, 2013].

3 Teaching greedy algorithms in primary and secondary school generally

Little children use the greedy concept when they use coins for payment. They don't think about it they just use it instinctively. In the mathematics lessons pupils learn greedy concept without using its name when they convert between the number systems at upper primary age. They are familiar with the idea to sort the input first then use greedy property for solving the sub-problems, having the optimal choice for the initial problem – without naming it and developing it to an exact procedure.

We think the stages of teaching greedy concept are:

- get acquainted with simple tasks solvable by this concept – naming and particularizing them
- practice and recognize if the task is of greedy type (simple tasks)
- prove the correctness of using greedy method for the previously occured problems
- teach special algorithms in graph theory
- find counter-examples, something that is not greedy (typical examples of dynamic programming – coin change in another coin systems, knapsack)
- and last stage is learning to look for counterexamples and prove the correctness

Teaching greedy concept is incorporated into teaching other methods and it appears in higher and higher levels from upper-primary-school-age to being a contestant on olympiads.

4 CSUnplugged

There are many activities for greedy concept in the book of CSUnplugged for elementary and lower secondary school-age pupils.

The first is for 7+ age children: it is named *Binary*, and the task is to change a number into the binary number system. It could be tried with other number systems, too. *The Muddy City* activity is a simple variation of minimal spanning tree, illustrating Kruskal's algorithm. *The orange game* (Routing and deadlock in networks) is a counter-example against the greedy concept. *Tourist town* is a greedy type about dominating sets, while *Ice roads* is one of Steiner trees for elementary and up. We could play with *Coin Change special cases* with children to practice greedy method without a computer.

5 BEBRAS

There are a lot of Bebras challenges where you can use the greedy idea from upper primary hard problems until secondary hard ones. Some examples are: *Bebras City I* (2009), *Upstream* (Flussaufwärts 2014), *Stagecoaches* (Postkutschen 2012), *Bank construction* (Dammbau 2015), *Bebras City II* (2009) and *Meeting point* (2014). The average rate is one task (and its variations) per year.

There was a problem in 2014 that looked like greedy but the right solution was DP: *Loading Lisas* for children of upper-primary-age. It is very important for children to see problems, which look like a greedy but actually are not, at early stage of their studies – the greedy concept is not universal, sometimes it fails.

6 Tasks for contestants of upper-primary-school-age

At this age the real life problems, the CSUnplugged activities, the BEBRAS tasks and mathematics problems are the first problems through which pupils become acquainted with greedy method like:

- the cheapest lunch possible works by getting the cheapest meat, fruit, vegetable, etc. respectively
- given a real coin system make change for a given amount with a minimal number of coins
- convert a given number into another number system format
- and others from the earlier list (CSUnplugged, BEBRAS).

After that, we can code these problems discussing the criteria for greediness: for an optimization problem, we are given a set of constraints and an optimization function. In a greedy method we attempt to construct an optimal solution in stages.

- At each stage we make a decision that appears to be the best (under some criterion) at the time.
- A decision made at one stage is not changed in the later stages, so each decision should satisfy the constraints.
- In every greedy-solvable problem we have to make an ordering first.

If the pupils are familiar with coding the known problems we could go further with variety of wording the machine scheduling problems. The initial problem is the following: we have a number of tasks to be done on a minimum number of machines, while each task has a start and an end time. This problem could be worded with photographers, jobs, burns of potteries or watching films.

In this age the problem-solving method of greedy type is very simple:

- sorting based on appropriate property and
- deciding whether the next element will be in the result set or will not.

Practicing tasks are on every online practice site, for example on *codeforces.com* (Team, Twins, Chat room, Business trip, Dragons, Slightly Decreasing Permutations, Buggy Sorting, Unlucky Ticket, Squares, Building Permutation, Magic Numbers, Roadside Trees, Ciel and Dancing, Yaroslav and Permutations, Permutation, Little Pigs and Wolves, Two Bags of Potatoes, Painting Eggs, Increase and Decrease, Boys and Girls, Combination, Ternary Logic, Multithreading, Sockets, Sail, Sale, Roma and Changing Signs, Little Elephant and Bits, Before an Exam, Vasya and the Bus, Snow Footprints, Coins, LLPS, Olympic Medal, Ilya and Matrix, Playing Cubes, Paper Work, Photographer); on *uva.onlinejudge.org* (Minimal coverage, Watering Grass, Station Balance, Coin Collector); on

topcoder.com (FanFailure, GroceryBagger, Boxing, SchoolAssembly, PlayGame, Apothecary, RockStar, Unblur); on *acm.timus.ru* (Minimal coverage); *on spoj.com* (I AM VERY BUSY, Wine trading in Gergovia, Encode Integer).

7 Tasks in lower-secondary-school-age

At this age children have to learn – preparing for competitions – graph theory: directed and undirected graphs, trees and connected components, breadth-first search and depth-first search algorithms. In addition, they could study some of the standard greedy algorithms to grasp the concept better with graphs: finding shortest paths with Dijkstra's algorithm, finding minimal spanning trees with Prim's algorithm and with Kruskal's algorithm. At this age the greedy type problems are mostly about graph theory because of the well-known special algorithms.

Beside the greedy graph algorithms the children have to think and implement greedy algorithms based on number theory, mathematical knowledge, advanced sorting and constructive algorithms, too. They should be able to distinguish the greedy type tasks from dynamic type problems.

In these problems "*the pupils designed greedy solutions, which they only intuitively verified*" [Ginat, 2003] and they step into the greedy trap. Proving the correctness of the greedy method is a must at this age and they also have to seek counter-evidence to their solutions.

There is a very large variety in the wording of the abovementioned problem types on every online practice site, for example on *uva.onlinejudge.org* (The Postal Worker Rings Once, Pick up sticks, XYZZY); on *topcoder.com* (Crossroads, TCSocks, HeatDeath, BioScore, Rationalization); on *spoj.com* (Assembly line, Lucius Dungeon, Milk Scheduling, Mining Camps, Digo plays with Numbers, Draw Skyline Graph, Expedition, Operators); and on *codeforces.com* (T-decomposition, Ilya and Two Numbers, Prime Permutation, Dispute, Text Editor, Little Elephant and Sorting, Black and White Tree).

8 National Olympiads

Contestants preparing for National Olympiads have to learn other standard greedy problems, like Fractional Knapsack Problem, Huffman Coding, Optimal Merging and Topological Sort. They meet a very large amount of tasks based on standard problems and problems mixed with other types of concepts.

The path getting to know the problem begins at this age with understanding examples, then selecting the appropriate method, proving that the asymptotic complexity yields to an acceptable running time with that method and looking for counterexamples. An exact proof of the selected greedy method's correctness is mandatory before implementing it. They must be able to choose between complete search, dynamic programming approach and greedy method for the problem, taking the appropriate running time and memory limit into consideration.

Sometimes the use of the greedy method depends on the pupil's knowledge about advanced data structures and powerful programming languages. In this level the use of advanced data structures, like priority queue or modifiable priority queue is necessary and it expands the range of tasks solvable wholly or partially by greedy method.

Example tasks on online practice sites are on: *spoj.com* (Help R2-D2!, Save Area 11); *topcoder.com* (GoldMine, RearrangeFurniture, MLBRecord, WorldPeace); *codeforces.com* (in every round in Div 1 and Div 2 are problems solvable by greedy or dynamic programming, Little Elephant and Bits, Polo the Penguin and Strings, k-Multiple Free Set, Lucky Conversion, Magical Boxes, Fish Weight, Testing Pants for Sadness, Lexicographically

Maximum Subsequence, Petya and Inequiations, Secrets, Maxim and Discounts, Purification, Lawnmower, Newspaper Headline, Homework, Zero Tree, Chips, Permutations, Cycles, Help General, Naughty Stone Piles, Game, Robbery, Main Sequence, Zero-One, Digits Permutations, Demonstration, Russian Roulette, The table, End of Exams, E-reader Display, Dima and Horses, Minimum Modular, Merging Two Decks, Track, Hamming Distance, Grocer's Problem, Sweets for Everyone!, The Next Good String, Metro Scheme, Road Repairs, Heaven Tour, Fire and Ice).

9 Regional and International Olympiads

There is greedy – usually variation of a classical one combined with other methods – on the International Olympiad in Informatics (IOI) in almost every year and one of the six problems is greedy or combined greedy on Regional Olympiads – like Central-European Olympiad in Informatics (CEOI), Baltic Olympiad in Informatics (BOI), Balkan Olympiad in Informatics (BOI), American Computer Science League (ACSL). So it is very important for contestants to be familiar with this concept's power and limitation as well.

"*Programming contests rarely involve the purely canonical versions of classical problems. The problem authors try to set the input bounds of problems that allow for Greedy strategies to be in an ambiguous range so that contestants cannot use the input size to quickly determine the required algorithm either Complete Search or Dynamic Programming approaches or greedy at all.*" [Halim, 2013]

Some tasks are partially solvable by greedy method or the solution is partially based on greedy approach in the tasks of IOI: in 1996: Job processing, in 2000: Car parking, in 2007: Sails, in 2008: Teleporters [Verhoeff, 2009], in 2011: Elephants, in 2014: Friend, in 2015: Teams, Boxes. One of the most complicated problem was SAILS in 2007 which solution required two different greedy ideas one after another in the same task.

On each CEOI, BOI, ACSL contest we can find a task for practicing. The greedy concept on these contests mostly appears in connection with graphs.

Conclusion

When children learn the basics of programming there are some problems from the real world where a greedy algorithm provides the solution. Greedy algorithms are easy to think of and usually easy to implement in the beginning. Later the search for counterexamples makes this type of problems interesting. The concept is infamous for being tricky; missing even a very small detail can be fatal. Children get to know examples and counterexamples and decide if the problem is solvable by greedy.

Preparing for contests, proving their correctness comes into view, it may require rigorous mathematical proofs, which are usually hard to retrieve. They have to identify, when one of the other concepts is the right solution. There is no general template on how to apply the greedy method to a given problem; the specification might give a good insight about it. In other cases, a hard problem may hide an ingenious greedy shortcut.

With a lot of practice, the basic concepts come to the fore again and again spirally – the greedy method is one of them – children will have enough sense and experience to use the basic concepts for the tasks on contests.

Bibliography

[1] *Bebras–International Contest on Informatics and Computer Fluency (2007-2015)* http://bebras.org; http://www.beaver-comp.org.uk/; http://informatik-biber.de/ [Accessed on: June, 2016]

[2] CORMEN, T.H., LEISERSON, C.E., RIVEST, R.L., STEIN, C. (2001). *Introduction to Algorithms.* MIT Press, 2nd edition.

[3] DAGIENE, V., STUPURIENE, G. (2016) *Bebras - a Sustainable Community Building Model for the Concept based Learning of Informatics and Computational Thinking.* Informatics in Education, VOL. 15, No 1, PP. 25-44.

[4] DASGUPTA, S., PAPADIMITRIOU, C.H., VAZIRANI, U.V. (2006). *Algorithms.* McGraw-Hill

[5] GINAT, D. (2003): *The Greedy Trap and Learning From Mistakes.* SIGSCE 2013, pp. 11-15.

[6] HALIM, S., HALIM, F. (2014) *Competitive Programming 3. The New Lower Bound of Programming Contests.* http://cpbook.net/

[7] Jagadish, M., Sridhar Iyer (2012): A Mehod to Construct Counterexamples for Greedy Algorithms. ITiSCE 2012, pp. 238-243

[8] KLEINBERG, J., TARDOS, É. (2006*). Algorithm Design.* Addison-Wesley

[9] Joan M. Lucas (2015): *Teaching greedy algorithms using a single problem domain*. Journal of Computing Sciences in Colleges archive, Volume 30 Issue 6, June 2015, pp. 89-96.

[10] SEDGEWICK, R., WAYNE, K. (2011). *Algorithms,* Fourth Edition. Addison-Wesley.

[11] SKIENA, S.S. (2008*). The Algorithm Design Manual.* Springer-Verlag, 2nd edition

[12] VERHOEFF, T. (2009) *20 Years of IOI Competition Tasks*, Olympiads in Informatics, Issue 3, pp.149-166

Lectured by: dr. Gyula Horváth.

Contact address:

Ágnes Erdősné Németh
Doctoral School, Faculty of Informatics, Eötvös Loránd University, H-1117 Budapest, Pázmány P. sétány 1/C, Hungary,
Batthyány Lajos High School, H-880 Nagykanizsa, Rozgonyi u. 23.
phone: +36-30-4745107 , e-mail: erdosne@blg.hu

László Zsakó, Doc. dr. hab. Ph.D,
Department of Media and Educational Informatics, Faculty of Informatics, Eötvös Loránd University, H-1117 Budapest, Pázmány P. sétány 1/C, Hungary,
phone: +36-1-372-2500 , e-mail: zsako@caesar.elte.hu