Constructive Algorithm

Alex Tung

27/2/2016

CODEFORCES β
Sponsored by Telegram

HOME CONTESTS GYM PROBLEMSET GROUPS RATING API HELP TESTLIB 8VC VENTURECUP

PROBLEMS SUBMIT STATUS STANDINGS CUSTOM TEST

Constructive algorithms

#	Name		Solved
629A	Far Relative's Birthday Cake	brute force, constructive algorithms, implementation	x3520
626B	Cards	constructive algorithms, dfs and similar, dp	x2870
622D	Optimal Number Permutation	constructive algorithms	x875
625D	Finals in arithmetic	constructive algorithms, implementation	x278
625C	K-special Tables	constructive algorithms, implementation	x3039
625B	War of the Corporations	constructive algorithms, strings	x3359
623A	Graph and String	constructive algorithms, graphs	x2140
618F	Double Knapsack	constructive algorithms, two pointers	x146
618B	Guess the Permutation	constructive algorithms	x3934
617D	Polyline	constructive algorithms	x2229
613C	Necklace	constructive algorithms	x357
611H	New Year and Forgotten Tree	constructive algorithms, flows	x49
610C	Harmony Analysis	constructive algorithms	x1764
612E	Square Root of Permutation	combinatorics, constructive algorithms, dfs and similar, graphs	x338

Constructive Algorithm

- Gives an answer by explicit construction of one
- Designing a constructive algorithm usually requires a lot of rough work and/or insight and/or intuition

Construction Problems in OI

- Construction problems are problems solvable using a constructive algorithm
- Usually more interesting and less “standard” (Hence popular in OI)
- Many difficult constructive problems are brilliant :D

- Personal experience (in OI/ACM):
  - Theorem 1. It feels awesome to solve a nontrivial construction problem in contest.
  - Theorem 2. If I am able to solve a construction problem in contest, it is trivial.

Examples from HKOI 2015/16

- Junior Q1 (Model Answer)
  - Find a model answer to make Alice pass and others fail
- Senior Q3 (Arithmetic Sequence)
  - Find a sequence without length-3 arithmetic subsequence

Example from Math

- Q1. Prove that for all positive integers N, there exists N consecutive composite numbers.

- Hint: consider (N+1)!

- A1. (N+1)! + 2, (N+1)! + 3, …, (N+1)! + N, (N+1)! + (N+1) are N consecutive composite numbers.

Non-example from Math

- Q2. (Bertrand's postulate) Prove that for all positive integers N, there exists a prime number in the interval [N, 2N].

- Do you think a constructive proof exists for Q2?

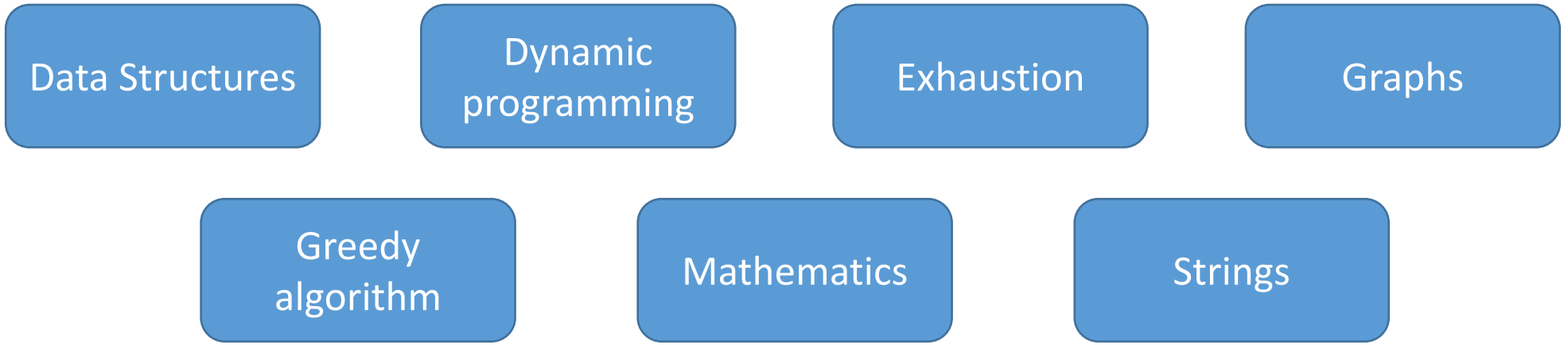
- A2. For those interested: https://en.wikipedia.org/wiki/Proof_of_Bertrand's_postulate

How to solve a construction problem

- Step 1: Identify it as a construction problem
- Step 2a: Look for concepts related to this problem
- Step 2b: Try small examples, observe a pattern
- Step 3: Make a guess
- Step 4: Convince yourself that it is correct
- Step 5: Code!
  - If AC, congratulations ☺
  - Otherwise, debug/go back to step 3

Step 1: Identify a construction problem

- “ad-hoc”, i.e. cannot be solved using general methods/algorithms
- Most do not *directly* fall into areas below
  - But knowledge in these areas may be helpful



- Usually requires contestants to “find a solution”

Step 2a: Look for related concepts

- Sequence (Finding general terms)
  - Triangular numbers: 1, 3, 6, 10, 15, …
  - Fibonacci numbers: 1, 1, 2, 3, 5, 8, …
- Graph
  - Look for special features
  - Bipartite? Planar? No three points (on plane) collinear? Etc.
- Greedy
  - Construct the answer in a greedy manner

Step 2b: Try small examples and observe

- Say, you are to find the minimal number of steps required to do sth
  - If f(1) = 2, f(2) = 4, f(3) = 6, f(4) = 8, guess f(n) = 2n
  - If g(1, 1) = 1, g(1, 2) = 2, g(2, 2) = 4, g(7, 8) = 56, guess g(a, b) = ab
- With the general formula, it is easier to come up with the actual method
- Options: by pen(cil) and paper / by computer program

Step 3: Make a guess

- Would this construction method work?
- Would f(n) = 2n be the number of steps required?

Step 4: Convince yourself that it is correct

- Test on machine
- Create cases to test your idea
- Write mathematical arguments (if time allows)

Step 5: Code!

- Typically, constructive algorithms are rather easy to code
- Step 4 (verification) is much more important

So, what now?

- Unlike in other algorithm lectures, this general flow of solving construction problems does not help much ☹

- What helps?
  - Learning discrete mathematics (remember to attend “Math in OI”!)
  - Practicing (e.g. CF has many problems tagged “constructive algorithms”)
  - Having a good sense/intuition

Activity

- Form groups
  - Members in each group will solve several construction problems together
  - Each group will be assigned one problem; after solving it, the group can attempt other problems
- Machines allowed
- After time is up, present your solution to the assigned problem

- Time allowed: 45 minutes
  - http://www.online-stopwatch.com/

Solution Session

Further Practice

- HKOJ
  - J151 Inverse Problem
  - S132 Safe Storage
  - S134 Unfair Santa Claus
  - I1413 Game
  - T032 Trishade
    - Challenging. Note that non-constructive solutions exist.

Further Practice

- Codeforces
  - 297C Splitting the Uniqueness (by gagguy)
  - 610C Harmony Analysis
  - 625C K-special Tables
  - …and many others; see “constructive algorithm” tag

The End

- Thank you