Algorithms in Competitive Programming +7

How can one start solving dynamic programming problems?

https://www.quora.com/What-are-systematic-ways-to-prepare-for-dynamic-programmi…

This question previously had details. They are now in a comment.

Answer Follow · 969 Request 2

22 Answers

Abhishek Verma, knows little bit about coding

Answered Mar 4, 2016

Looking at most of the DP problems, they don't seem to be solve-able using DP on the first site! If you try to solve it by thinking it to be a DP problem, most likely you won't get it.

Instead forget about DP, apply brute-force first! Then, you will most probably land into recursion. Congrats work is almost done :D Now, just see, if you are doing some work repetitively? Can you use some array or table to store value of the recursion calls to use it later? How many parameters are there in recursion call and can you do the memoisation now according to the number of parameters? Most likely, you will now get the problem and solve it :)

Now, let's see how we should begin, if we don't know anything about DP :)

1. First and most important part, take that CLRS book. Open its dynamic programming chapter, read it full and thoroughly understand it. Three basic DP problems- rod cutting, matrix chain multiplication and LCS are given in it. Understand completely how we have approached their solution, why this solution works etc., everything.
2. Now, when you know the basic DP concepts, go to Algorithms GeeksforGeeks and see all the problems here. These are all the most common DP problems and almost every problem you will face, it would be the variation of these problems only.
3. Now, you are done with collecting all your weapons, it's time for war :D Try to solve as many problems you can on online judges like spoj, codechef, hackerearth, hackerrank etc. A good start can be found here A2 Online Judge . Try solving from easiest ones to more difficult ones.
4. First try your level best to solve a particular problem. Don't get disheartened if you are not able to solve any problem. After trying your best, search on internet for its solution, discuss with your peers, try to see how the problem is approached, why and how your thinking was wrong, how it has been solved etc. This is the only way to get better at competitive programming :)
5. Repeat step 3 and 4 :D

All the best :)

Upvote · 184 Share

Related Questions

How does one become good at solving algorithm/programming problems? I am a beginner, and it was suggested that I read the CLRS book to learn a...

How do you start problem solving on programming?

What is the best way to learn dynamic programming?

How do you develop recurrence relationships for dynamic programming problems?

What are some real-world problems that have been solved with dynamic programming?

What kind of programming problems should I solve to improve my problem solving and thinking capabilities?

Dynamic Programming (DP): How do I solve Candy on LeetCode?

Is problem solving the worst part of programming?

What is your favourite dynamic programming problem?

How do I solve dynamic programming problems without fear?

+ Ask New Question

More Related Questions

Question Stats

968 Public Followers

179,259 Views

Last Asked Jun 30, 2017

1 Merged Question

Edits

Is this answer still relevant and up to date?

Add a comment... Recommended All

Bhavit Sharma, Just entered the arena.
Updated Dec 14, 2015

Originally Answered: How can I start solving Dynamic Programming problems?

To get the basic idea behind solving DP problems, go to HackerRank and in algorithms domain, select dynamic programming, and solve all the problems in the easy and medium section.

after that, go to A2 Online Judge.

Make your account and in the problem categories, select the dynamic programming. It contains a comprehensive list of DP problems(with it's source) ranging from easy to very difficult, and solve them.

Also, as a side reference you can checkout GeeksforGeeks | A computer science portal for geeks and you can choose the category DP which contains hell lot of articles of DP which contains theory as well as enormous number of problems.

After you're fed with the above things, I recommended you to solve every previous contest of any competitive programming website, they contains lot of DP problems which are worth looking at.

12.7k Views · View Upvoters

Upvote · 23 Share

Add a comment... Recommended All

Anonymous
Answered Jul 31, 2013

I was facing same problem as you . The following strategy was useful to me , though i am not an expert in DP but now i am able to solve old div 1 500/1000 dp solutions.
following is my strategy :

1. Start with recursion because recursion builds the basic intuition for DP problems . You can start from here : TopCoder Statistics - Problem Archive

if you a are a beginner to programming you can start from here :
TopCoder Statistics - Problem Archive

2. After you have undergone first step you have a basic platform to start DP . So read this by vortys TopCoder Feature Articles

NOTE : In the beginning you may find the tabular DP hard so I will advice you to-

GOOD LUCK

16.7k Views · View Upvoters

Anonymous
Updated Feb 10, 2015 · Upvoted by Nithin Chandy, 5 years as competitive programmer

Firstly, let me put forth my own thought process for solving DP problems (since its short), and then refer you to other sources.

NOTE: All DPs can be (re)formulated as recursion. The extra effort you put in in finding out what is the underlying recursion will go a long way in helping you in future DP problems.

STEP1: Imagine you are GOD. Or as such, you are a third-person overseer of the problem.
STEP2: As God, you need to ***decide*** what choice to make. ***Ask a decision question***.
STEP3: In order to make an informed choice, you need to ask "what ***variables*** would help me make my informed choice?". This is an important step and you may have to ask "but this is not enough info, so what more do I need" a few times.
STEP4: Make the choice that gives you your best result.

In the above, the *variables* alluded to in Step3 are what is generally called the "state" of your DP. The *decision* in Step2 is thought of as "from my current state, what all states does it depend upon?"

Trust me: I've solved loads of TC problems on DP just using the above methodology. It took me about max 2 months to imbibe this methodology, so unlike everyone's "Keep practicing" advice (which I would liken to Brownian motion), I'm suggesting the above 'intuition'.

Btw, if you can identify me by my advice, good for you. The point of going anonymous is mainly so that people don't *blindly* upvote when they see "*X* added an answer to *Algorithms*: ..."

Now for some examples:

amount to.
Step1: I am an overseer (just a psychological step)
Step2: I go through the array, and ask, "*does the largest sum begin at this point?*"
Step3: I need to know what is the largest sum that begins at the next point, in order to decide if it can be extended or not.
Step4: I either extend it to the next point, or I cut it off here: f(i) = max(arr[i], arr[i] + f(i+1)).

The above particular example was demonstrated to me by a friend and prior to his demonstration (of how easy DP is), I was completely baffled by DPs myself.

Q. *I have a set of N jobs, and two machines A and B. Job i takes A[i] time on machine A, and B[i] time on machine B. What is the minimum amount of time I need to finish all the jobs?*
Step1: I am the overseer (this problem lends itself more naturally to being "God").
Step2: I go through jobs from 1 to N, and have to decide on which machine to schedule job *i*.
Step3: If I schedule the job on machine *A*, then I then have a load of A[i] + best way to schedule the remaining jobs. If its on B, then I have a load of B[i] + best way to schedule the remaining jobs.
But wait! "best way to schedule remaining jobs" doesn't account for the fact that the i'th job will potentially interfere. Thus, I need to infact also pass on the "total load" on each machine in order to make my informed choice.
Therefore, I need to modify my question to: "**Given that the current load on A is *a*, and on B is *b*, and I have to schedule jobs i to N, what is the best way to do so?**"
Step4: f(a, b, i) = { max(a, b) : i == N+1, else min(f(a + A[i], b, i+1), f(a, b + B[i], i+1): i <= N}. Final answer is in f(0, 0, 1).

Now, enough of examples (answer is becoming too long). If you have any DP question you'd like me to work my magic on, post it as a comment :)

Please have a look at Mimino's answers on DP: What are systematic ways to prepare for dynamic programming?

55.4k Views · View Upvoters

Upvote · 565 Share

Shreyash Srivastava
A very good answer! I would like you to throw some light on this one: Problem Statem...


Answered Oct 28, 2016

DP (Dynamic Programming) is a Conceptual topic which is always a trending topic in any Competitive Programming Contest. Now the question arises how to solve the questions of it as there is always one/two questions of it in any CP contest(or may be not!!!).

Lets break it in to points and form strategy how to solve it:-

1. **dynamic programming** (also known as **dynamic optimization**) is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions . (Source:-[1] Wikipedia) , So Basically it says DP consists of two parts ,First , Breaking a given problem into smaller part (subproblem) and Second Remembering the result of the smaller subproblem to get the desired Result.
2. Now , Yeah definition is always very easy to understand and follow up But how to solve the Problem related to it in Contest??? For that I would Personally recommended Practically DP as much as possible ;)
3. But , First in order to solve problem you much first get basic reference to how to solve such problem . For that there are various tutorial sites such as:-

- Algorithms - GeeksforGeeks
- Tutorial for Dynamic Programming
- Dynamic Programming - From Novice to Advanced

Now , Since you may have learned and watch few tutorials of it , now what to do next??

Just start coding , NO ONE IN THE WORLD COULD TEACH YOU CODING UNLESS AND UNTIL YOU DON'T START PRACTICING IT!!!!

So, I think best way to start coding is search by tags for DP, or hackerrank provide separate category for it !!!

- Solve Algorithms Code Challenges

PS :- Happy Coding!!!

Footnotes

[1] Dynamic programming - Wikipedia

2.4k Views · View Upvoters

Upvote · 3 Share


Add a comment... Recommended All


Rishabh Thukral, supercool276 at competitive platforms.
Answered Mar 3, 2016

If you have gone through tutorials on topcoder or codeforces then you might

You can refer to any book if you want, I would recommend Introduction to algorithms by Cormen.

For more questions you can goto A2OJ.com where every question across all platforms are categorized .

Use the tag #dynamic-programming on spoj and you can read comments to get the idea of problem's difficulty.

Hope this helps.

Happy Coding.

2.9k Views · View Upvoters

Upvote · 2 Share

Add a comment... Recommended All

Mansi Gupta, The purpose has always been clear, just figuring out the right way.

Updated May 25

Hey,

It is really cool that you are trying to learn DP.

DP is one of the most interesting paradigms in algorithmic coding. Its awesomeness is reflected in the fact that it reduces the time complexity from exponential to polynomial.

I would suggest you to start with DP tutorial on topcoder and move on to solving DP problems on SPOJ,start with most solved. Then try the DP ladder at codeforces. Slowly and gradually you would start getting comfortable with the working of DP and be able to visualize it. But yeah it would happen over a few months, be patient, don't give up, once you get hold of it, you would love it.

All the best!

3.6k Views · View Upvoters

Upvote · 4 Share

Add a comment... Recommended All

Prasoon Goyal, PhD student in AI at UT Austin

Answered Jun 29, 2017

I'll give some techniques by explaining how you come up with a DP solution to some classic DP problems:

can do this, and just considering all possibilities gives you the solution.

Let's look at some examples:

- Coin change problem : Given a value N, if we want to make change for N cents, and we have infinite supply of each of $S = S_1, S_2, . . , S_m$ valued coins, how many ways can we make the change? The order of coins doesn't matter.

How do you proceed? Clearly, there are two dimensions to the problem — the value N and the types of coins. If N was small like 1, or there was just one type of coin, the answer would be trivial. So your subproblems will either have a smaller N or fewer types of coins, or both.

How do you know which one is it? Let's consider the case of smaller N. Say the given N was 10. Now if you knew how many ways you can make change for a smaller value, say 8, can you *efficiently* find the no. of ways to make change for 10? Note that the operative word here is "efficiently" — if you can't do it efficiently, DP is useless. Now, you extend the above question — if you knew how many ways you can make change for *all values* of N smaller than 10, can you efficiently find the no. of ways to make change for 10? One way you think you can proceed is to add the no. of ways to make change for $10 - S_1$, no. of ways to make change for $10 - S_2$, ..., because adding one coin to each of these ways would give you a sum of 10. However, this is incorrect — because the order of coins is not important, so you will be overcounting. In particular, if $S_1 = 1$ and $S_2 = 2$, then $10 - S_1$ include ways that include coins of denomination 2, and you'll add 1 to these to get 10. Similarly, $10 - S_2$ include ways that include coins of denomination 1, and you'll add 2 to these to get 10. So there is repeated counting.

Let's go to the next dimension — the types of coins. Suppose you knew how many ways there were to make change of N using coins of denominations $S_1, \dots, S_{m-1}$. Can you use this to generate the no. of ways to make change of N using $S_1, \dots, S_m$? Clearly not.

But, if you knew how to make change for *any value* below N using coins $S_1, \dots, S_{m-1}$, you can get the answer. In particular, no. of ways to make change for N using coins $S_1, \dots, S_m$ = no. of ways to make change for $N - S_m$ using coins $S_1, \dots, S_m$ + no. of ways to make change for N using coins $S_1, \dots, S_{m-1}$. Note that the above two terms are mutually exclusive — the first term corresponds to the cases where you use at least one coin of denomination S_m, while the second case includes no coins of denominations S_m.

Now, from this point, you just have to perform table-filling to get the answer.

- All-pair shortest path problem : Given a graph, find the shortest path from every vertex to every other vertex.

What can you reduce to make the problem size smaller? The graph size — so find shortest path from vertex i to j using subgraph $v_1, \dots, v_k$. And that gives you the Floyd-Warshall algorithm.

given set S with sum equal to N.

Again, there are two dimensions — the size of the set and N. Proceeding similar to coin-change problem, you need to use subproblems along both dimensions — there is a way to make N if there is a way to make $N - s_m$ using $s_1, \ldots, s_{m-1}$, or if there is a way to make N using $s_1, \ldots, s_m$. This has the same interpretation as above — either s_m is included in the sum or it is not.

Often, people start by thinking about these mutually exclusive cases like inclusion and exclusion, which makes it tricky because different problems have different criteria to generate mutually exclusive sets. The correct way is to start thinking in terms of dimensions, which is much more consistent across problems.

Finally, I would recommend Algorithm Design book. It has one chapter each on various paradigms of algorithms, like greedy algorithms, dynamic programming, divide and conquer, etc. The end-of-chapter exercises are doable in number, and cover basic to advanced problems.

6.6k Views · View Upvoters

Upvote · 29 Share

Adarsh Jois
I remember you recommending me Tardos and Kleinberg as well, its easily one of my ...

Vivek Srivastava
Answered Feb 28, 2017

Dynamic programming is very important topic and one just cannot neglect the topic. If you understand recursion well, you can solve DP problems using Memoization else you can use Tabulation which is iterative in nature. I would recommend below link for DP problems coded in C++.

Dynamic Programming Interview Questions

But remember one cannot master DP in a day or two. It will require time and patience, but most importantly it requires confidence. Start with basic problems, understand them first and then after you get comfortable, try to solve problems by yourself without looking at proposed solutions.

3.3k Views · View Upvoters

Upvote · 4 Share

Add a comment... Recommended All

Top Stories from Your Feed