SHARE QUESTION

QUESTION TOPICS

- Online Judges
- Dynamic Programming (DP)
- Codeforces
- Sphere Online Judge (SPOJ)
- CodeChef
- TopCoder
- Competitive Programming
- Computer Programmers
- Algorithms

How can one start solving Dynamic Programming problems?

I am facing difficulty in solving DP probs, can someone tell me how to hit such problems and where to start from.

This is follow up to : What are systematic ways to prepare for dynamic programming?

5 ANSWERS

Anonymous

192 upvotes by Shailendra Sason, Ankit Sablok, Mohan Kumar, (more)

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.

BEST QUESTIONS IN CODECHEF

- How do I start competitive programming?
- How should I use the book *Introduction to Algorithms* by Cormen?
- How did St. Petersburg State University prepare for the ACM-ICPC 2014 which they went on to win?
- Anish Shankar is the youngest Indian to qualify ACM ICPC. How did he prepare for it?
- How do I join CodeChef?
- CodeChef: How was your most satisfying experience of getting to the solution?
- In ACM ICPC what is the rationale behind selecting the best team from each

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:

Q. *Given an array, find the largest sum along a contiguous segment of it.*
Simple enough, and you probably *know* the answer, but lets see what the 4 steps 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

college rather than selecting the best teams overall? Is that fair?

★ What are the very basic algorithms that every Computer Science student must be aware of?

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) = { min(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?

Updated 6 Aug, 2013.

Upvote 192 Downvote Comments 8+

Related Questions

How do I recognize a problem as a dynamic programming problem?

What are some one dimensional dynamic programming problems that I can solve on popular programming online judges?

How do I go about solving problems through dynamic programming? I know the concepts but I'm not able to program for new problems!

Ketan Gupta, I don't hate anyone

15 upvotes by Kaushal Kumar, Marcelle Bonterre, Amit Chhabra, (more)

Check out this website - CS 97SI: Introduction to Competitive Programming Contests

Explains it awesomely!

Written 23 Jan.

Upvote 15 Downvote Comment

Pierre Yves Schobbens

4 upvotes by Ankit Gupta, Nalin Savara, Abhra Basak, (more)

I assume you mean DP = Dynamic Programming

See e.g. in the book "Introduction to Algorithms" (Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein)

1. Specify your problem
2. From this, prove (or not) the Optimal substructure property.
3. Check if the recursive calls introduce redundant computation
4. If yes, store the optimal substructures in an array.

Written 31 Jul, 2013.

Upvote 4 Downvote Comment

Anonymous

14 upvotes by Mohan Kumar, Amit Jain, Gaurav Mishra, (more)

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 begin with memoization + recursion (which you will learn during step 2)
After you do a few problems then tabular DP will become easier .

GOOD LUCK

Written 31 Jul, 2013.

Upvote 14 Downvote Comment

Obinna Okechukwu, Autodidact, Illinois Tech. '14

8 upvotes by Swati Jha, Raghu Kanchiraju, Ankur Sao, (more)

Follow the Feynman Algorithm. It's a simple 3 step process.

1. Write down the problem
2. Think really hard
3. Write down the solution

Written 30 Dec, 2013.

Upvote 8 Downvote Comment 1

1 Answer Collapsed

Write an answer

Related Questions

Solve with dynamic programming?

How do I decide if a problem can be solved use Dynamic Programming?

What is a stepwise explanation (beginner to advance) to solve the dynamic programming problem using bitmasking?

What are the very basic problems one needs to solve while starting out with competitive programming?

What is easiest approach to tackle Dynamic programming problems?

What is dynamic programming?

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

Top Stories