Competitive Programming and Mathematics Society

ICPC Workshop #7

Introduction to the ICPC and Dynamic Programming

Ryan and Patrick

Dynamic Programming

Definition

Dynamic Programming is am optimisation technique that prevents repeating calculation for equal sub-problems.

For most dynamic programming problems, we need to define two things:

- Base Cases: The initial (often the most simple) version of the problem.
- Recursive Cases: Larger problems defined generally in terms of some dynamic programming "state" which break down into smaller cases

Solving Fibonacci with Dynamic Programming

Given an integer, N, output the N-th Fibonacci number. Each Fibonacci number is defined as the sum of the two previous numbers, where $F_0 = 0$ and $F_1 = 1$.

Solving Fibonacci with Dynamic Programming

Given an integer, N, output the N-th Fibonacci number. Each Fibonacci number is defined as the sum of the two previous numbers, where $F_0 = 0$ and $F_1 = 1$.

Can we define base cases?

Solving Fibonacci with Dynamic Programming

Given an integer, N, output the N-th Fibonacci number. Each Fibonacci number is defined as the sum of the two previous numbers, where $F_0 = 0$ and $F_1 = 1$.

Can we define base cases?
$F(0) = 0$ and $F(1) = 1$

Solving Fibonacci with Dynamic Programming

Given an integer, N, output the N-th Fibonacci number. Each Fibonacci number is defined as the sum of the two previous numbers, where $F_0 = 0$ and $F_1 = 1$.

Can we define base cases?

$F(0) = 0$ and $F(1) = 1$

Can we define the recursive cases, in terms of N?

Solving Fibonacci with Dynamic Programming

CPMSOC

ARC CLUBS SUPPORTED BY Arc INDEPENDENTLY RUN

Given an integer, N, output the N-th Fibonacci number. Each Fibonacci number is defined as the sum of the two previous numbers, where $F_0 = 0$ and $F_1 = 1$.

Can we define base cases?

$F(0) = 0$ and $F(1) = 1$

Can we define the recursive cases, in terms of N?

$F(N) = F(N-1) + F(N-2)$

Solving Fibonacci with Dynamic Programming

Given an integer, N, output the N-th Fibonacci number. Each Fibonacci number is defined as the sum of the two previous numbers, where $F_0 = 0$ and $F_1 = 1$.

Can we define base cases?
$F(0) = 0$ and $F(1) = 1$
Can we define the recursive cases, in terms of N?
$F(N) = F(N-1) + F(N-2)$

When we implement this as a standard recursive function, we find that we often call the function with the same value of N, we repeat the calculation, but ultimately return the same answer. The idea of dynamic programming is to prevent us from repeating that calculation, and just returning the answer we calculated earlier.

This improves the time complexity from $O(2^N)$ to $O(N)$.

Iterative vs Recursive Dynamic Programming

CPMSOC

Arc CLUBS SUPPORTED BY Arc INDEPENDENTLY RUN

There are two general approaches to standard DP problems: Iterative and Recursive.

Recursive Dynamic Programming:

- Start with the big problem, and break it down into smaller problems.
- Often implemented as recursive functions which take the DP "state" as arguments and returns the answer for that subproblem.
- You must be careful when making lots of recursive calls not to create memory issues or have a function call itself.

Iterative Dynamic Programming:

- Start with the smaller problems. Calculate the answer to those problems and build up to the final answer.
- Often implemented as as for loops from the small case to the large case.
- Depending on the problem, the looping behaviour can make this a constant factor more efficient than an equivalent recursive dynamic programming algorithm.

Dynamic Programming in 2 Dimensions

Problem: IOI '99 Little Shop of Flowers

https://dmoj.ca/problem/ioi99p1

Can we define a DP state in terms of two variables?

Dynamic Programming in 2 Dimensions

Problem: IOI '99 Little Shop of Flowers

https://dmoj.ca/problem/ioi99p1

Can we define a DP state in terms of two variables?

Can we define a recursive function for the maximum aesthetics given that we begin at vase N, bunch F?

Can we define base cases for our function?

What is the time complexity of a DP solution to this problem?

Dynamic Programming on a Rooted Tree

CPMSOC

Problem: Snurgle Holders. https://orac2.info/problem/aiio07snurgle/

You have a factory with a construction process defined in N *steps*. Each step is required for 1 other step, except for step N. You can place inspectors on any number of steps, as long as no two are on adjacent steps (adjacent defined as one step leading to the other). What is the maximum number of inspectors you can place in the factory?

Dynamic Programming on a Rooted Tree

CPMSOC

Problem: Snurgle Holders. https://orac2.info/problem/aiio07snurgle/

You have a factory with a construction process defined in N *steps*. Each step is required for 1 other step, except for step N. You can place inspectors on any number of steps, as long as no two are on adjacent steps (adjacent defined as one step leading to the other). What is the maximum number of inspectors you can place in the factory?

Suppose we are free to place an inspector at step i. Can we break down the answer into two cases?

Dynamic Programming on a Rooted Tree

Problem: Snurgle Holders. https://orac2.info/problem/aiio07snurgle/

You have a factory with a construction process defined in N *steps*. Each step is required for 1 other step, except for step N. You can place inspectors on any number of steps, as long as no two are on adjacent steps (adjacent defined as one step leading to the other). What is the maximum number of inspectors you can place in the factory?

Suppose we are free to place an inspector at step i. Can we break down the answer into two cases?

What is the time complexity of a recursive dynamic programming solution to this?

Attendance form :D

https://forms.gle/opFYJL3zUyrBN3Cx8

Wrap Up

- Tommorow: Annual Programming Competition!
- Next Wed: AGM
- Next Sat: ANZAC 7
- Week 4 Tue: Maths workshop