Advanced Counting Techniques

CS1200, CSE IIT Madras

Meghana Nasre

April 7, 2020

- Principle of Inclusion-Exclusion ✓
- Recurrences and its applications
- Solving Recurrences

Recurrences

Recursive definitions play an important role in CS.
Some examples:

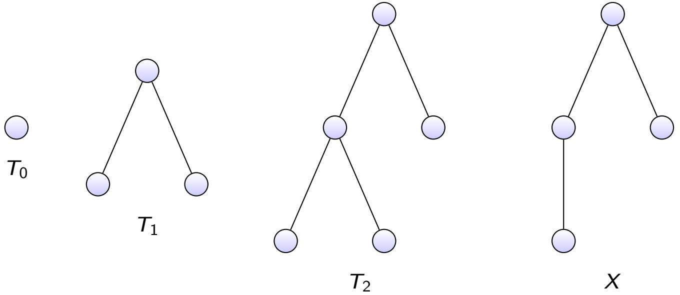
- Fibonacci sequence
- Towers of Hanoi (reading exercise Example 2, Chapter 8)

Today's class: Modeling a variety of problems having the same solution.

Example 1: Full binary trees with n internal nodes

Full binary tree: A single node is a full binary tree. If T_1 and T_2 are disjoint full binary trees then we can get a full binary tree with root r together with r connecting to the roots of left subtree T_1 and right subtree T_2.

Recall: Every node in a full binary tree has either two children or zero children.



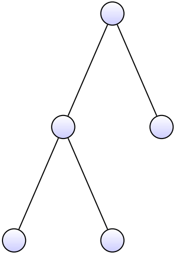
- T_i denotes a full binary tree with i internal nodes.
- X is not a full binary tree.

Example 1: Full binary trees with n internal nodes

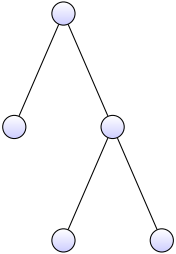
Goal: Count the number of full binary trees with n internal nodes. Call it $f(n)$

Ex: Construct explicitly and count the values for $n = 0, 1, 2$.

Verify that $f(0) = f(1) = 1$ and $f(2) = 2$.



T_2^1



T_2^2

Example 1: Full binary trees with n internal nodes

Goal: Count the number of full binary trees with n internal nodes. Call it $f(n)$

A recursive formulation. Clearly, $f(0) = f(1) = 1$.

A tree with n internal nodes has one internal node as root. We are left with $n-1$ internal nodes. These can be distributed into the left subtree and the right subtree as:

- 0 internal nodes in left subtree and $n-1$ internal nodes in right subtree. This gives $f(0) \cdot f(n-1)$ ways of constructing trees with n internal nodes.

OR

- 1 internal node in left subtree and $n-2$ internal nodes in right subtree. This gives $f(1) \cdot f(n-2)$ ways of constructing trees with n internal nodes.

OR

- 2 internal nodes in left subtree and $n-3$ internal nodes in right subtree. This gives $f(2) \cdot f(n-3)$ ways of constructing trees with n internal nodes.

$\vdots$

- $n-1$ internal nodes in left subtree and 0 internal nodes in right subtree. This gives $f(n-1) \cdot f(0)$ ways of constructing trees with n internal nodes.

Example 1: Full binary trees with n internal nodes

Goal: Count the number of full binary trees with n internal nodes. Call it $f(n)$

A recursive formulation. Clearly, $f(0) = f(1) = 1$.

A tree with n internal nodes has one internal node as root. We are left with $n-1$ internal nodes. These can be distributed into the left subtree and the right subtree as:

$$\begin{aligned} f(n) &= f(0) \cdot f(n-1) + f(1) \cdot f(n-2) + \ldots + f(n-2) \cdot f(1) + f(n-1) \cdot f(0) \\ &= \sum_{i=0}^{n-1} f(i) \cdot f(n-1-i) \end{aligned}$$

- Why is adding the terms valid?
- What is a closed form expression for this recurrence?

Example 2: Number of ways to parenthesize

We are given $n + 1$ integers: $x_0, x_1, x_2, \ldots x_n$ and we wish to parenthesize them to compute their product.

Goal: How many ways are there to parenthesize? Denote this by $g(n)$.

$g(n)$ denotes the number of ways to parenthesize $n + 1$ integers and not n integers!

Example: Say $n = 2$, that is, we have 3 integers x_0, x_1, x_2

- We multiply $x_0 x_1$ first followed by multiplying the product by x_2.

$$((x_0 \cdot x_1) \cdot x_2)$$

- We multiply $x_1 x_2$ first followed by multiplying the product by x_0.

$$(x_0 \cdot (x_1 \cdot x_2))$$

Are there other ways?
No! Note that $((x_0 \cdot x_2) \cdot x_1)$ is not a valid way to parenthesize x_0, x_1, x_2
The last "$\cdot$" (multiplication symbol) has to appear between two integers. There are only two possible places first, between x_1 and x_2. This gives the first way. Second between x_0 and x_1. This gives the second way to multiply.
Base cases: $g(0) = 1$, $g(1) = 1$.

Example 2: Number of ways to parenthesize

We are given $n+1$ integers: $x_0, x_1, x_2, \dots x_n$ and we wish to parenthesize them to compute their product.

Goal: How many ways are there to parenthesize? Denote this by $g(n)$.

$g(n)$ denotes the number of ways to parenthesize $n+1$ integers and not n integers!

Let the last "$\cdot$" appear between x_k and x_{k+1}

- On the left side of the last "$\cdot$" we have $k+1$ integers $x_0, \dots x_k$. These can be parenthesized in $g(k)$ ways.
- On the right side of the last "$\cdot$" we have $n-k$ integers $x_{k+1}, \dots, x_n$. These can be parenthesized in $g(n-k-1)$ ways. Note that it is not $g(n-k)$ ways.
- If last "$\cdot$" appears between x_k and x_{k+1} then number of ways is $g(k) \cdot g(n-k-1)$ ways.

Since k can take values between 0 and $n-1$, we have

$$g(n) = g(0) \cdot g(n-1) + g(1) \cdot g(n-2) + \dots + g(n-2) \cdot g(1) + g(n-1) \cdot g(0)$$

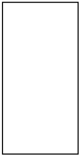
Compare it with the recurrence for $f(n)$ earlier.

Other Examples

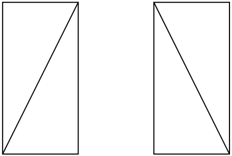
Goal: Determine the number of balanced strings of parenthesis of length $2n$.

- An empty string is a balanced parenthesis.
- (()) and ()() are two balanced parenthesis of length $2 * 2$.
- Every balanced parenthesis of length $2n$ contains n open and n close parenthesis. In addition, every prefix of the string has as many open parenthesis as many close parenthesis.

Goal: Given $n + 2$ side convex polygon, in how many ways can triangulate it?



Input



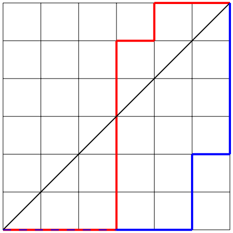
Two ways to triangulate it

Ex: Work out the recursive formulation for both the examples.

Example 3: Diagonal Avoiding Grid Paths

Goal: Input is a $n \times n$ grid. To compute number of paths from $(0, 0)$ to (n, n).
Constraints:

- Use steps of one unit and go right or up at each step.
- Each path contains n right steps (R) and n up (U) steps.
  we have already solved this when there were no more constraints
- Additional condition: Paths should not cross the main diagonal.



- Two paths – one red and another blue.
- Red path is invalid. (RRRUUUUURURR)
- Blue path is valid. (RRRRRUURUUUU)

Goal: Compute total number of valid paths, that is, diagonal avoiding paths.

A first guess it to count all paths from $(0, 0)$ to (n, n) which is $\binom{2n}{n}$ and then claim that half of the total paths are diagonal avoiding.

Example 3: Diagonal Avoiding Grid Paths

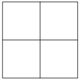
Input: An $n \times n$ grid.

Goal: To compute number of diagonal avoiding paths from $(0,0)$ to (n,n), denote it by $h(n)$.

$n = 1$

$h(n) = 1$

$n = 2$



$h(n) = 2$

- Note that $h(2) \neq \frac{1}{2}\binom{2\times 2}{2}$
- Thus, $h(n) \neq \frac{1}{2}\binom{2n}{n}$

An invalid path has some prefix in which there are more Us than Rs.

In contrast every valid path satisfies the property that every prefix has at least as many Rs as the number of Us. Thus, every diagonal avoiding grid path is in one-to-one correspondence with a string of balanced parenthesis of length $2n$.

This is simply obtained by replacing every "(" by R and every ")" by U.

Example 3: Diagonal Avoiding Grid Paths

Claim: The number of diagonal avoiding grid paths in an $n \times n$ grid are:

$$h(n) = \frac{1}{n+1}\binom{2n}{n}$$

Plan of the proof:

- Count total number of paths from $(0, 0)$ to (n, n).
- Subtract the number of invalid paths. That is, count invalid paths.
- However, since that is not straightforward, convert invalid paths into another counting problem!

How does an invalid path look like?

RRRUUUUURURR

- It has a prefix in which there are more Us than Rs. (that is why it crosses the diagonal!)
- Select the smallest such prefix. (RRRUUUU – in the above).

Example 3: Diagonal Avoiding Grid Paths

Claim: The number of diagonal avoiding grid paths in an $n \times n$ grid are:

$$h(n) = \frac{1}{n+1}\binom{2n}{n}$$

Every invalid path has:

- Exactly n Rs and n Us. (since it does reach n, n).
- Has a prefix (the smallest one) in which there is one more U than R. That is, it has x Rs and $x+1$ Us.
- The remaining part of the path has $n-x$ Rs and $n-(x+1)$ Us.

A clever trick

- Keep the prefix of the path as it is and in the remaining path flip the Us and Rs. Call this the modified path.

RRRUUUURURR → RRRUUUURURUU

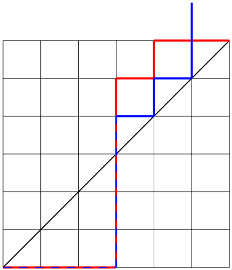
Every modified path has:

- $x + n - (x+1) = n-1$ many Rs. (why?)
- $x + 1 + n - x = n+1$ many Us. (why?)

Example 3: Diagonal Avoiding Grid Paths

The trick in action.

invalid path (RRRUUUUURURR) → modified path RRRUUUURURUU



- Note that the modified blue path follows the red path for the prefix part, and then flips it.

How does it help?

- Every modified path always ends in $(n-1, n+1)$.
  revisit the previous slide and construct a proof!
- One to one correspondence between invalid paths and paths (no more constraints of diagonal avoiding) from $(0, 0)$ to $(n-1, n+1)$.
  establish this correspondence!

Example 3: Diagonal Avoiding Grid Paths

Claim: The number of diagonal avoiding grid paths in an $n \times n$ grid are:

$$h(n) = \frac{1}{n+1}\binom{2n}{n}$$

Plan of the proof:

- Count total number of paths from $(0,0)$ to (n,n).
- Subtract the number of invalid paths. That is, count invalid paths.
- However, since that is not straightforward, convert invalid paths into another counting problem! ✓

- Total number of paths from $(0,0)$ to (n,n) is: $\binom{2n}{n}$.
- Number of invalid paths is: $\binom{2n}{n+1}$

Number of valid paths $= h(n) =$

$$\binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1}\binom{2n}{n}.$$

Catalan Number

The number $\frac{1}{n+1}\binom{2n}{n}$ is called the n-th Catalan number.

- Gives a closed form solution to several examples (seen today).
- Many more applications.
- References Section 8.1[KR].