

Recursion

22c:19
Chapter 8
Hantao Zhang

1

Recursion

- Recursion means defining something, such as a function, in terms of itself
 - For example, let $f(x) = x!$
 - We can define $f(x)$ as $f(x) = x * f(x-1)$
- Sequences are functions from natural numbers to reals.

2

Recursion example

- Find $f(1)$, $f(2)$, $f(3)$, and $f(4)$, where $f(0) = 1$
- a) Let $f(n+1) = f(n) + 2$
 - $f(1) = f(0) + 2 = 1 + 2 = 3$
 - $f(2) = f(1) + 2 = 3 + 2 = 5$
 - $f(3) = f(2) + 2 = 5 + 2 = 7$
 - $f(4) = f(3) + 2 = 7 + 2 = 9$
- b) Let $f(n+1) = 3f(n)$
 - $f(1) = 3 * f(0) = 3 * 1 = 3$
 - $f(2) = 3 * f(1) = 3 * 3 = 9$
 - $f(3) = 3 * f(2) = 3 * 9 = 27$
 - $f(4) = 3 * f(3) = 3 * 27 = 81$

3

Recursion example

- Find $f(1)$, $f(2)$, $f(3)$, and $f(4)$, where $f(0) = 1$

c) Let $f(n+1) = 2^{f(n)}$

- $f(1) = 2^{f(0)} = 2^1 = 2$
- $f(2) = 2^{f(1)} = 2^2 = 4$
- $f(3) = 2^{f(2)} = 2^4 = 16$
- $f(4) = 2^{f(3)} = 2^{16} = 65536$

$$f(n) = 2n + 1$$

d) Let $f(n+1) = f(n)^2 + f(n) + 1$

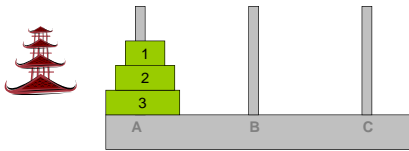
- $f(1) = f(0)^2 + f(0) + 1 = 1^2 + 1 + 1 = 3$
- $f(2) = f(1)^2 + f(1) + 1 = 3^2 + 3 + 1 = 13$
- $f(3) = f(2)^2 + f(2) + 1 = 13^2 + 13 + 1 = 183$
- $f(4) = f(3)^2 + f(3) + 1 = 183^2 + 183 + 1 = 33673$

$$f(n) = 3^n$$

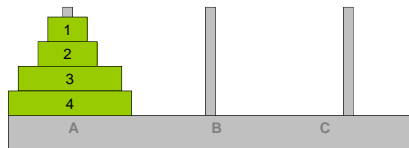
4

Start here - Instructions

1. Transfer all the disks from pole A to pole B.
2. You may move only ONE disk at a time.
3. A large disk may not rest on top of a smaller one at anytime.

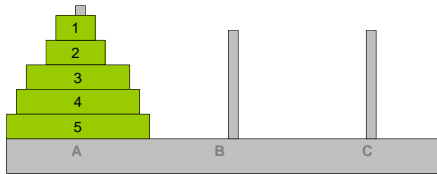


Try this one!



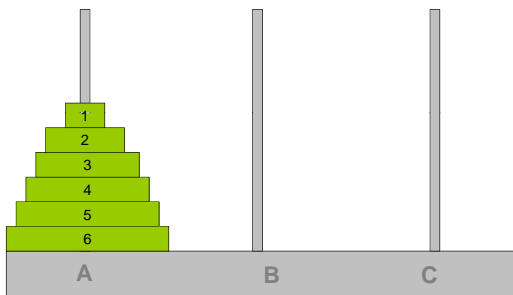
Shortest number of moves??

And this one



Shortest number of moves??

Now try this one!



Shortest number of moves??

How to solve Tower of Hanoi of n disks?

- If $n = 1$, "move disk 1 from A to B", done.
- If $n > 1$,
 1. Solve the Tower of Hanoi of $n-1$ disks, from A to C;
 2. "move disk n from A to B"
 3. Solve the Tower of Hanoi of $n-1$ disks, from C to B.

```
Hanoi (int n, char A, char B, char C) {
  if (n==1) cout << "move disk from " << A << " to " << B;
  else {
    Hanoi(n-1, A, C, B);
    cout << "move disk " << n << " from " << A << " to " << B;
    Hanoi(n-1, C, B, A);
  }
}
```

Counting the moves:
Let $f(n)$ be the number of moves for n disks.
 $f(1) = 1$;
 $f(n) = 2f(n-1) + 1$.

Let $f(n)$ be the number of moves for n disks.

$$f(1) = 1;$$
$$f(n) = 2f(n-1) + 1.$$

Number of Disks	Number of Moves
1	$f(1) = 1$
2	$f(2) = 2 \cdot 1 + 1 = 3$
3	$f(3) = 2 \cdot 3 + 1 = 7$
4	$f(4) = 2 \cdot 7 + 1 = 15$
5	$f(5) = 2 \cdot 15 + 1 = 31$
6	$f(6) = 2 \cdot 31 + 1 = 63$

Let $f(n)$ be the number of moves for n disks.

$$f(1) = 1;$$
$$f(n) = 2f(n-1) + 1.$$

Prove: $f(n) = 2^n - 1.$

By induction.

- Base step: $n = 1.$
 - Left = $f(1) = 1;$
 - Right = $2^1 - 1 = 1$
- Induction hypothesis: $f(n-1) = 2^{n-1} - 1.$
- Inductive step:
 - Left = $f(n) = 2f(n-1) + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 1.$
 - Right = $2^n - 1.$

11



Fascinating fact

So the formula for finding the number of steps it takes to transfer n disks from post A to post C is:

$$2^n - 1$$

- If $n = 64$, the number of moves of single disks is 2 to the 64th minus 1, or 18,446,744,073,709,551,615 moves! If one worked day and night, making one move every second it would take slightly more than 580 billion years to accomplish the job! - far, far longer than some scientists estimate the solar system will last.

Fibonacci sequence

- Definition of the Fibonacci sequence

– Non-recursive:
$$F(n) = \frac{(1 + \sqrt{5})^n - (1 - \sqrt{5})^n}{\sqrt{5} \cdot 2^n}$$

– Recursive:
or:
$$F(n) = F(n-1) + F(n-2)$$
$$F(n+1) = F(n) + F(n-1)$$

- Must always specify base case(s)!
 - $F(1) = 1, F(2) = 1$
 - Note that some will use $F(0) = 1, F(1) = 1$

13

Fibonacci sequence in Java

```
long Fibonacci (int n) {
    if ( (n == 1) || (n == 2) )
        return 1;
    else
        return Fibonacci (n-1) + Fibonacci (n-2);
}

long Fibonacci 2 (int n) {
    return (long) ((Math.pow((1.0+Math.sqrt(5.0)), n) -
        Math.pow((1.0-Math.sqrt(5.0)), n)) /
        (Math.sqrt(5) * Math.pow(2, n)));
}
```

14

Recursion pros

- Easy to program
- Easy to understand

15

Recursion cons

- Consider the recursive Fibonacci generator
- How many recursive calls does it make?
 - $F(1)$: 1
 - $F(2)$: 1
 - $F(3)$: 3
 - $F(4)$: 5
 - $F(5)$: 9
 - $F(10)$: 109
 - $F(20)$: 13,529
 - $F(30)$: 1,664,079
 - $F(40)$: 204,668,309
 - $F(50)$: 25,172,538,049
 - $F(100)$: 708,449,696,358,523,830,149 $\approx 7 * 10^{20}$
 - At 1 billion recursive calls per second (generous), this would take over 22,000 years
 - But that would also take well over 10^{12} Gb of memory!

16

Bad recursive definitions

- Consider:
 - $f(0) = 1$
 - $f(n) = 1 + f(n-2)$
 - What is $f(1)$?
- Consider:
 - $f(0) = 1$
 - $f(n) = 1 + f(-n)$
 - What is $f(1)$?

17

Defining sets via recursion

- Same as mathematical induction:
 - Base case (or basis step)
 - Recursive step
- Example: the set of positive integers
 - Basis step: $1 \in S$
 - Recursive step: if $x \in S$, then $x+1 \in S$

18

Defining sets via recursion

- Give recursive definitions for:
 - a) The set of odd positive integers
 - $1 \in S$
 - If $x \in S$, then $x+2 \in S$
 - b) The set of positive integer powers of 3
 - $3 \in S$
 - If $x \in S$, then $3 \cdot x \in S$
 - c) The set of polynomials with integer coefficients
 - $0 \in S$
 - If $p(x) \in S$, then $p(x) + cx^n \in S$
 - $c \in \mathbb{Z}, n \in \mathbb{Z}$ and $n \geq 0$

19

Defining strings via recursion

- Terminology
 - λ is the empty string: ""
 - Σ is the set of all letters: $\{a, b, c, \dots, z\}$
 - The set of letters can change depending on the problem
- We can define a set of strings Σ^* as follows
 - Base step: $\lambda \in \Sigma^*$
 - If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$
 - Thus, Σ^* is the set of all the possible strings that can be generated with the alphabet

20

Defining strings via recursion

- Let $\Sigma = \{0, 1\}$
- Thus, Σ^* is the set of all binary strings
 - Or all possible computer files

21

String length via recursion

- How to define string length recursively?
 - Basis step: $len(\lambda) = 0$
 - Recursive step: $len(wx) = len(w) + 1$ if $w \in \Sigma^*$ and $x \in \Sigma$
- Example: $len("aaa")$
 - $len("aaa") = len("aa") + 1$
 - $len("aa") = len("a") + 1$
 - $len("a") = len("") + 1$
 - $len("") = 0$
 - Result: 3

22

Strings via recursion example

- Given a string $x = a_1a_2\dots a_n$ x^R stands for its reversal: $x^R = a_n a_{n-1} \dots a_1$. Eg. $x = abc$, $x^R = cba$.
- A string x is a palindrome if $x = x^R$. Eg. $x = aba$.
- Give a recursive definition for the set of string that are palindromes
 - We will define set P , which is the set of all palindromes
- **Basis step:** $\lambda \in P$
 - Second basis step: $x \in P$ when $x \in \Sigma$
- **Recursive step:** $xpx \in P$ if $x \in \Sigma$ and $p \in P$

23

How many binary strings of length n that do not contain the pattern 11?

- $n = 0$: 1 string (λ , the empty string)
 - $n = 1$: 2 strings (0 and 1)
 - $n = 2$: 3 strings (00, 01, 10)
 - $n = 3$: 5 strings (000, 001, 010, 100, 101)
 - $n = 4$: 8 strings (0000, 0001, 0010, 0100, 1000, 0101, 1001, 1010)
-
- Any pattern?
 - A Fibonacci sequence!

24

How many binary strings of length n that do not contain the pattern 11?

- $n = 2$: 3 strings (00, 01, 10)
- $n = 3$: 5 strings (000, 001, 010, 100, 101)
- $n = 4$: 8 strings (0000, 0001, 0010, 0100, 1000, 0101, 1001, 1010)
- The strings of $n=4$ can be divided into two classes:
 - $X = \{ 0000, 0001, 0010, 0100, 0101 \}$ and
 - $Y = \{ 1000, 1001, 1010 \}$
 - X can be obtained from $n = 3$: adding a leading 0
 - Y can be obtained from $n = 2$: adding leading 10.

25

How many binary strings of length n that do not contain the pattern 11?

- Let S_n be the set of binary strings of length n that do not contain the pattern 11.
- For any string x in S_{n-1} , $y = 0x$ is a string of S_n .
- For any string x in S_{n-2} , $z = 10x$ is a string of S_n .
- Any string of S_n is either y or z .
- Hence $|S_n| = |S_{n-1}| + |S_{n-2}|$
- From $|S_0| = 1$, $|S_2| = 2$, we can compute any $|S_n|$.

26

Stirling Numbers of the Second Kind

- You have 3 students in a class, and you want to divide them up into 2 groups (perhaps groups of 1) to work on a project. In how many ways can you do this?
- Answer: 3 ($\{a, b\}\{c\}$, $\{a, c\}\{b\}$, $\{a\}\{b, c\}$).
- You have 4 students and want to divide them up into 2 groups?
- Answer: 7 ($\{a, b, c\}\{d\}$, $\{a, b, d\}\{c\}$, $\{a, c, d\}\{b\}$, $\{a\}\{b, c, d\}$, $\{a, b\}\{c, d\}$, $\{a, c\}\{b, d\}$, $\{a, d\}\{b, c\}$).
- You have 5 students and want to divide them up into 2 groups?
- Answer: $S_2(5,2)=15$
- You have 5 students and want to divide them up into 3 groups?
- Answer: $S_2(5,3)=25$
- You have 8 students and want to divide them up into 5 groups?
- Answer: $S_2(8,5)=1050$.

27

Stirling Numbers of the Second Kind

- The numbers $S_2(n, k)$ are called the Stirling Numbers of the Second Kind.
- $S_2(n, k)$ = the number of ways of dividing n objects into k **non-empty** subsets.
- You can find a nice page about them at <http://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html>.

28

Stirling Numbers of the Second Kind

- Now evaluate $S_2(4,3)$.
- Let $X = \{x, y, z, w\}$ and $Y = \{x, y, z\}$.
- How many ways to divide Y into two non-empty subsets?
 - Answer: $S_2(3,2) = 3$ ($\{x,y\}\{z\}$, $\{x,z\}\{y\}$, $\{x\}\{y,z\}$)
 - Adding $\{w\}$ into either solution of $S_2(3,2)$ will produce a solution of $S_2(4,3)$.
- How many ways to divide Y into three non-empty subsets?
 - Answer: $S_2(3,3) = 1$ ($\{x\}\{y\}\{z\}$)
 - Adding w into either set of ($\{x\}\{y\}\{z\}$) will produce a solution of $S_2(4,3)$.
- Hence, $S_2(4,3) = S_2(3,2) + 3S_2(3,3) = 3 + 3*1 = 6$.

29

Stirling Numbers of the Second Kind

- In general, let $X_1 = \{x_1\}$ and $X_n = X_{n-1} \cup \{x_n\}$.
- (A) For any solution of dividing X_{n-1} into $k-1$ non-empty sets, adding $\{x_n\}$ will produce a solution of dividing X_n into k non-empty sets .
- (B) For any solution of dividing X_{n-1} into k non-empty sets, adding x_n into one of k sets will produce a solution of dividing X_n into k non-empty sets.
- For any solution of dividing X_n into k non-empty sets, if x_n is in a singleton set, it is a solution from (A); otherwise, it's solution from (B).
- Hence, $S_2(n,k) = S_2(n-1,k-1) + kS_2(n-1,k)$.

30

Stirling Numbers of the Second Kind

- Theorem: For $n > k > 1$, $S_2(n, k) = S_2(n-1, k-1) + k \cdot S_2(n-1, k)$.
- Proof: Count the partitions of X_n into k non-empty subsets according to whether x_n is in a subset by itself. If so, then there are $S_2(n-1, k-1)$ ways to partition the remaining elements of X_n . If not, then there are $S_2(n-1, k)$ ways to partition the remaining elements of X_n and then k ways to choose the subset to place x_n in.

31

Stirling Numbers of the Second Kind

- $S_2(n, k) = S_2(n-1, k-1) + kS_2(n-1, k)$.
- Intuitively $S_2(0, 0) = 1$.
- If $n > 0$, then $S_2(n, 0) = 0$
- $S_2(n, 1) = 1$
- $S_2(n, n) = 1$
- If $n < k$, then $S_2(n, k) = 0$.
- $S_2(n, 2) = 2^{n-1} - 1$
- Note: There are 2^n subsets of X_n . A subset and its complement consist of a solution of $S_2(n, 2)$, except the empty set.

32

Stirling Numbers of the Second Kind

- There is also a formula for $S_2(n, k)$, namely

$$S_2(n, k) = \frac{1}{k!} \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} j^n$$

but for many purposes the recurrence relation is easier to use.

33

Stirling Numbers of the Second Kind

	k										
S ₂ (n,k)	0	1	2	3	4	5	6	7	8	9	10
0	1										
1	1										
2	1	1									
3	1	3	1								
4	1	7	6	1							
n	5	1	15	25	10	1					
	6	1	31	90	65	15	1				
	7	1	63	301	350	140	21	1			
	8	1	127	966	1701	1050	266	28	1		
	9	1	255	3025	7770	6951	2646	462	36	1	
	10	1	511	9330	34105	42525	22827	5880	750	45	1

34

Stirling Numbers of the Second Kind

- How many onto functions are there from a 5-set to a 4-set?
- Answer: There are $S_2(5,4)$ ways to divide 5 elements into four non-empty subsets.
- For each division, there are $4!$ ways to map the four subsets to the codomain.
- The total number of onto functions is $4!S_2(5,4)=24 \cdot 10 = 240$.
- In general, the number of onto functions from a m -set to a n -set is $n!S_2(m, n)$.

35
