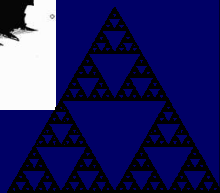


Recursion and Divide & Conquer Revisited



Recursion: when a function calls itself

The factorial function:
 $f(n) = n! = n*(n-1)*(n-2)* \dots *2*1$

Examples: $1! = 1$, $2! = 2$, $3! = 3*2*1 = 6$
 $4! = 4*3*2*1 = 4*3! = 4*6 = 24$

In general: $n! = n * (n-1)!$
 That is: $f(n) = n*f(n-1)$ **Recursion**
 $f(1) = 1$

More recursion

Adding two n digit numbers:
 $1234 + 8988 = ?$

$1204 = 1200+4 = 12*100 + 4$,
 $8988 = 8900 +88 = 89*100+88$
 $1204+8988 = (12+89)*100 + (4+88)$

$Add(1204, 8988) =$ **Recursion**
 $Add(12, 89)*100 + Add(4, 88)$

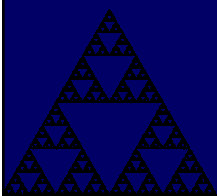
Even more recursion



Recursion

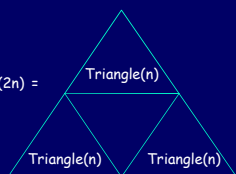
$People(Group) = 1 + People(SmallerGroup1) + People(SmallerGroup2)$
 $People(SmallestGroup) = 1$

Infinite recursion



Sierpinski Triangle

$Triangle(2n) =$



No smallest triangle

In computer science we only have finite recursion!
 We need to specify what the smallest case looks like.

Even more recursion



Recursion

$People(Group) = 1 + People(SmallerGroup1) + People(SmallerGroup2)$
 $People(SmallestGroup) = 1$

Divide and Conquer Method

Factorial: $f(n) = n * f(n-1)$ with $f(1) = 1$

Factorial(n):
 If $n = 1$, return 1 First check if we have smallest case
 Else
 $x = \text{Factorial}(n-1)$ Solve recursive pieces one by one
 Return $n * x$ Combine results from pieces


Runtime?

TIME vs INPUT SIZE


For any algorithm, define
INPUT SIZE = # of bits/digits to specify inputs,

Define
 $T(n)$ = the worst-case amount of time used on inputs of size n .

We Often Ask:
 What is the **GROWTH RATE** of $T(n)$?



Recall: Time complexity of grade school addition

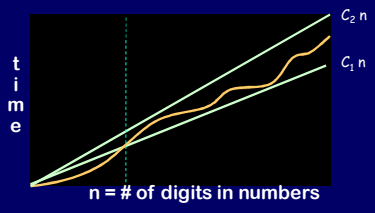


On any reasonable computer adding 3 digits can be done in **constant** time.

Fixed time, specific to the processor

$T(n)$ = The amount of time grade school addition uses to add two n digit numbers
 = $\theta(n)$ = linear time.

$f = \theta(n)$ means that f can be sandwiched between two lines for all large n



Runtime of Factorial

Runtime(Factorial(n)) =
 Runtime(Factorial(n-1)) + 1

Factorial(n):
 If $n = 1$, return 1
 Else
 $x = \text{Factorial}(n-1)$
 Return $n * x$

In terms of the number: Time(1) = 1,
 Time(n) = 1 + Time(n-1) =
 1 + (1 + (1 + ...)) = n

In terms of the input size:
 The number n has $d = \lfloor \log_{10} n \rfloor + 1$ digits.
 $T(d) = n \geq 10^{d-1}$ **EXPONENTIAL!!!**

Find the Minimum among n numbers

0x1, return 0

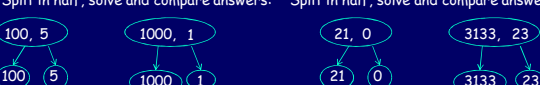
100, 5, 1000, 1, 21, 0, 3133, 23

First half Second half

An instance of the same problem of half the size!

An instance of the same problem of half the size!

Split in half, solve and compare answers: Split in half, solve and compare answers:



Adding two n digit numbers 2

Any n digit number X can be represented as:

$X = a \cdot 10^{n-1} + B$,
where a has 1 digit and B has n-1 digits

Example: $123456 = 1 \cdot 10^{6-1} + 23456$

Add: Divide and Conquer Method 2

Adding two n digit numbers: *Split off first digit*

$$X + Y = [a \cdot 10^{n-1} + B] + [c \cdot 10^{n-1} + D] = ?$$

$$\text{Add2}(X, Y) = (a + c) \cdot 10^{n-1} + \text{Add2}(B, D)$$

$\text{Add2}(X, Y)$:

If X and Y have 1 digit: First check if we have smallest case
Return X+Y

Else

Split into pieces

$X = a \cdot 10^{n-1} + B$ and $Y = c \cdot 10^{n-1} + D$ Solve recursive pieces one by one

$R1 = \text{Add2}(B, D)$, $R2 = a + c$,

Return $R2 \cdot 10^{n-1} + R1$ ← Combine results

Running time of Add2

$$\begin{aligned} \text{Time}(\text{Add2}(\text{two } n \text{ digit numbers})) = & \\ & \text{Time}(\text{Add2}(\text{two } n-1 \text{ digit numbers})) + \\ & \text{Time}(\text{add two } 1 \text{ digit numbers}) + \\ & \text{Time}(R2 \cdot 10^{n-1} + R1) \end{aligned}$$

(R2+carry from R1) followed by (last n-1 digits of R1) at most two digits!
constant time!

Example: $X=19=1 \cdot 10+9$, $Y=22=2 \cdot 10+2$

$R2=1+2=3$, $R1=9+2=11$

$R2 \cdot 10 + R1 = 30 + 11 = (3+1)$ followed by $1 = 41$

Running time of Add2

$$\begin{aligned} \text{Time}(\text{Add2}(\text{two } n \text{ digit numbers})) = & \\ & \text{Time}(\text{Add2}(\text{two } n-1 \text{ digit numbers})) + \\ & \text{Time}(\text{add two } 1 \text{ digit numbers}) + \\ & \text{Time}(R2 \cdot 10^{n-1} + R1) \end{aligned}$$

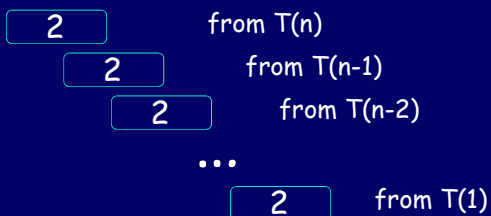
(R2+carry from R1) followed by (last n-1 digits of R1) Takes constant time!

In terms of input size:

$$T(n) = T(n-1) + 1 + 1 = T(n-1) + 2$$

$$T(1) = 1$$

Runtime of Add2: $T(n) = 2 + T(n-1)$



$$= n * 2 = \theta(n)$$

Add2 is Grade School Addition!

```

Add2(X, Y):
If X and Y have 1 digit:
  Return X+Y
Else
  X = a*10^{n-1}+B and Y = c*10^{n-1}+D
  R1 = Add2(B, D), R2 = a+c,
  Return R2*10^{n-1}+R1
    
```

Add2(19, 21)

$$\begin{array}{r} 19 + \\ 21 \\ \hline \end{array}$$

Add2 is Grade School Addition!

Add2(X, Y):
 If X and Y have 1 digit:
 Return X+Y
 Else
 $X = a \cdot 10^{n-1} + B$ and $Y = c \cdot 10^{n-1} + D$
 $R1 = \text{Add2}(B, D)$, $R2 = a+c$,
 Return $R2 \cdot 10^{n-1} + R1$

Add2(19, 21)

$a=1, B=9, c=2, D=1$

1. $R1 = \text{Add2}(9, 1)$

→ $R1 = 10$

$$\begin{array}{r} 1 \\ 19 + \\ 21 \\ \hline 0 \end{array}$$

Add2 is Grade School Addition!

Add2(X, Y):
 If X and Y have 1 digit:
 Return X+Y
 Else
 $X = a \cdot 10^{n-1} + B$ and $Y = c \cdot 10^{n-1} + D$
 $R1 = \text{Add2}(B, D)$, $R2 = a+c$,
 Return ($R2 + \text{carry from } R1$) followed by (last $n-1$ digits of $R1$)

Add2(19, 21)

$a=1, B=9, c=2, D=1$

1. $R1 = \text{Add2}(9, 1)$

→ $R1 = 10$

2. $R2 = 1 + 2 = 3$

3. Return $(3+1)$ followed by $0 = 40$.

$$\begin{array}{r} 1 \\ 19 + \\ 21 \\ \hline 40 \end{array}$$