Recurrence Relations

CCDSALG

Pau Rivera

Recursion

Recursion is just another form of iteration, except the iterations are defined as smaller versions of the original problem.

Recursion

This is what a recursive function looks like:

```
recurse (input) {
    if (base case condition)
        return x;

    return recurse(smaller input) + y;
}
```

Recursion

Let's try to find the frequency count per-line, just like for loops:

```
recurse (input) {
    if (base case condition)
        return x;

    return recurse(smaller input) +
    y;
}
```

Base Case Condition cost c
Base Case Return cost b

Recursive Case cost r

But what is r?

Recursion

If we define the total operation count as a function:

```
recurse (input) {
    if (base case condition)
        return x;

    return recurse(smaller input) + y;
}
```

Total T(input)

Base Case Condition cost c

Base Case Return cost b

Recursive Case cost r

Our r will also have the function inside it:

$r = T(\text{smaller input}) + c + r_{exp}$

Recursion

And our total frequency count will also be recursive:

Total $T(input) = c + b + ir$

Total recursions/iterations i

Base Case Condition cost c

Base Case Return cost b

Recursive Case cost $r = T(smaller\ input) + c + r_{exp}$

Recursion

How do we turn our recursive frequency count into something that isn't recursive or something that is closed-form?

Recursion

How do we turn our recursive frequency count into something that isn't recursive or something that is closed-form?

We have to start with redefining our operation count function into a **recurrence relation**.

Recurrence Relations

Basically the math term for recursive functions. They look like:

$$T(n) = \begin{cases} 1 & , n = 0 \\ T(n-1) + n & , n > 0 \end{cases}$$

Recurrence Relations

Pretty similar to our understanding of recursion:

$$T(n) = \begin{cases} 1 & , n = 0 \quad \leftarrow \text{base case} \\ T(n-1) + n & , n > 0 \quad \leftarrow \text{recursive case} \end{cases}$$

Recurrence Relations

Solve for the **nth triangle number** using recursion.

Ex:

Input	Process	Output
5	5 + 4 + 3 + 2 + 1	15

Recurrence Relations

Let's try it on a recursive function:

```
sum(n) {
    if (n == 0)
        return 0;           ← base case

    return sum(n-1) + n;    ← recursive case
}
```

Recurrence Relations

Let's try it on a recursive function:

```
sum(n) {
    if (n == 0)
        return 0;        ← base case      →
    
    return sum(n-1) + n; ← recursive case →
}
```

$$T(n) = \begin{cases} 1 & , n = 0 \\ & \end{cases}$$

Recurrence Relations

Let's try it on a recursive function:

```
sum(n) {
    if (n == 0)
        return 0;

    return sum(n-1) + n;
}
```

← base case →

← recursive case →

$$T(n) = \begin{cases} 2 & , n = 0 \\ T(n-1) + 1 & , n > 0 \end{cases}$$

Recursive Case cost $r = T(\text{smaller input}) + c + r_{exp}$

Recurrence Relations

So we have our recurrence relation for the frequency count, how do we find the closed-form equation?

$$T(n) = \begin{cases} 1 & , n = 0 \\ T(n-1) + 3 & , n > 0 \end{cases}$$

Iteration Method (Supplement)

https://www.youtube.com/watch?v=TEzbkIggJfo

Iteration Method

Given our recurrence relation, we can iterate via recursion to expand the recursive case:

$$T(n) = T(n-1) + 3$$

$$T(n) = \begin{cases} 1 & , n = 0 \\ T(n-1) + 3 & , n > 0 \end{cases}$$

Iteration Method

$$T(n) = \begin{cases} 1 & , n = 0 \\ T(n-1) + 3 & , n > 0 \end{cases}$$

Given our recurrence relation, we can iterate via recursion to expand the recursive case:

$$\begin{aligned} T(n) &= T(n-1) + 3 && = T(n-1) + 3 \\ &= T(n-1-1) + 3 + 3 && = T(n-2) + 6 \\ &= T(n-1-1-1) + 3 + 3 + 3 && = T(n-3) + 9 \\ &= T(n-1-1-1-1) + 3 + 3 + 3 + 3 && = T(n-4) + 12 \end{aligned}$$

... Let k = number of iterations/recursions

$$= T(n-k) + 3k$$

Iteration Method

$$T(n) = \begin{cases} 1 & , n = 0 \\ T(n-1) + 3 & , n > 0 \end{cases}$$

Now, assuming the recursions will eventually terminate and hit the base case, we can calculate how many iterations it takes to hit the base case from n by setting the value inside the recursive function to our base case:

$T(n) \quad = \quad T(n-k) + 3k$

where k = number of iterations/recursions

Iteration Method

$$T(n) = \begin{cases} 1 & , n = 0 \\ T(n-1) + 3 & , n > 0 \end{cases}$$

Now, assuming the recursions will eventually terminate and hit the base case, we can calculate how many iterations it takes to hit the base case from n by setting the value inside the recursive function to our base case:

$T(n) = T(n-k) + 3k$

where k = number of iterations/recursions

$T(n-k) = T(0)$

For function outputs to be similar, function inputs must also be similar:

$n - k = 0$

$n = k$

Iteration Method

$$T(n) = \begin{cases} 1 & , n = 0 \\ T(n-1) + 3 & , n > 0 \end{cases}$$

Now, we substitute our iteration count back into the original equation:

$T(n) \quad = \quad T(n-k) + 3k$

where k = number of iterations/recursions

$k = n$

$= \quad T(n - n) + 3n$

$= \quad T(0) + 3n$

we know the value of T(0)

And voila, closed-form expression!

$= \quad 1 + 3n$

Recursions: Guidelines

1) Define the ***recurrence relation***
2) **Iterate** the recursive case of the recurrence relation
3) Find the **relation/pattern** between the number of iterations k and the operation count $T(n)$ (e.g. $T(n) = T(n - k) + 3k$)
4) **Equate** the recursion to the base case
5) **Find** k in terms of n
6) **Substitute** k back into the relation, and substitute the base case for the function value

Questions? ☺