Lecture 5: Recurrence Relations

Appendix B

Recurrence Relations

Problem: Calculate $n!$

Inputs: a nonnegative integer n

Outputs: $n!$

```
int fact (int n)
{
      if (n == 0)
            return 1;
      else
            return n * fact(n - 1);
}
```

What is the basic operation of this function?

Recurrence Relations

Problem: Calculate n!

Inputs: a nonnegative integer n

Outputs: n!

```
int fact (int n)
{
     if (n == 0)
          return 1;
     else
          return n * fact(n - 1);
}
```

What is the basic operation of this function?

- How many times is it performed?

Recurrence Relations

Problem: Calculate *n*!

Inputs: a nonnegative integer *n*

Outputs: *n*!

```
int fact (int n)
{
      if (n == 0)
            return 1;
      else
            return n * fact(n - 1);
}
```

What is the basic operation of this function?

- How many times is it performed? We might intuitively know it's *n* times, but complex recursive algorithms are harder to analyze.

Recurrence Relations

Analysis of recursive algorithms is often not straightforward!

Steps to take:

1. Represent the algorithm's time complexity with a **recurrence relation**.
2. Solve the recurrence relation.

- t_n = total # of basic operations performed in a recursive algorithm.
- t_{n-1} = # of basic operations performed in the first recursive call of an algorithm that reduces its input size by 1.

Recurrence Relations

Basic Operation: `n * fact(n - 1);`

How many times is this multiplication performed?

Recurrence Relations

Basic Operation: `n * fact(n - 1);`

How many times is this multiplication performed?

➢ The total # of multiplications is the # performed when `fact(n - 1)` is recursively computed plus one more at the top level.

t_n	$=$	t_{n-1}	$+$	1
(Total multiplications)		(# Multiplications in 1st recursive call)		(1 Multiplication at top level)

- This is a simple example of a **recurrence relation**.

Recurrence Relations

$t_n = t_{n-1} + 1$

- This is a simple example of a **recurrence relation**.
  - It is called this because the output of the function at *n* is given in terms of the output of the function at a smaller value of *n*.
- A recurrence relation is <u>not</u> a **unique function**.
  - i.e. plugging 100 in for *n* doesn't immediately output the total # of operations.

We can **solve** a recurrence relation to generate a unique function.

- To do this, we first find an **initial condition** (i.e. for what value of *n* does recursion stop? How many operations are performed at that level?)

Recurrence Relations

Once we have a recurrence relation and have determined the **initial condition**, we follow three steps to solve it:

1. **Expand**: Write out several instances of the recurrence relation, starting with the base case, until we see a pattern emerge.
2. **Guess**: Guess what the solution is based on this pattern
3. **Verify**: Use mathematical induction to prove our guess is correct.

Recurrence Relations

What is the initial condition (i.e. **base case)** for this factorial algorithm?

How many multiplications occur at this base case?

```
int fact (int n)
{
      if (n == 0)
            return 1;
      else
            return n * fact(n - 1);
}
```

Recurrence Relations

What is the initial condition (i.e. **base case)** for this factorial algorithm?

- When $n = 0$

How many multiplications occur at this base case?

```
if (n == 0)
    return 1;
```

No multiplications occur, so the **initial condition** is $t_0 = 0$

$t_n = t_{n-1} + 1 \qquad \text{for } n > 0$

$t_0 = 0$

Recurrence Relations

$t_n = t_{n-1} + 1 \qquad \text{for } n > 0$

$t_0 = 0$

Next, we **expand**: Start at the initial condition and write out the next few cases.

$$
\begin{aligned}
t_0 &= 0 \\
t_1 &= t_{1-1} + 1 = 0 + 1 = 1 \\
t_2 &= t_{2-1} + 1 = 1 + 1 = 2 \\
t_3 &= t_{3-1} + 1 = 2 + 1 = 3
\end{aligned}
$$

What pattern emerges from this expansion? (i.e. how does the result of each equation relate to n in each t_n?)

Recurrence Relations

$t_n = t_{n-1} + 1$ for $n > 0$

$t_0 = 0$

Next, we **expand**: Start at the initial condition and write out the next few cases.

$$t_0 = 0$$
$$t_{\mathbf{1}} = t_{1-1} + 1 = 0 + 1 = \mathbf{1}$$
$$t_{\mathbf{2}} = t_{2-1} + 1 = 1 + 1 = \mathbf{2}$$
$$t_{\mathbf{3}} = t_{3-1} + 1 = 2 + 1 = \mathbf{3}$$

What pattern emerges from this expansion? (i.e. how does the result of each equation relate to n in each t_n?) $\boldsymbol{t_n = n}$

Recurrence Relations

$t_n = n$ is our **guess**. We next use induction to **verify** this is true.

Induction base: $t_0 = 0$
Induction hypothesis (guess): $t_n = n$
Induction step: show that *if* the hypothesis is true, $t_{n+1} = n + 1$ is also true.

What's next? (The recurrence relation is $t_n = t_{n-1} + 1$)

Recurrence Relations

$t_n = n$ is our **guess**. We next use induction to **verify** this is true.

Induction base: $t_0 = 0$

Induction hypothesis (guess): $t_n = n$

Induction step: show that *if* the hypothesis is true, $t_{n+1} = n + 1$ is also true.

What's next? (The recurrence relation is $t_n = t_{n-1} + 1$)

- Plug $n + 1$ into the recurrence relation:

Recurrence Relations

$t_n = n$ is our **guess**. We next use induction to **verify** this is true.

Induction base: $t_0 = 0$
Induction hypothesis (guess): $t_{\mathrm{n}} = n$
Induction step: show that *if* the hypothesis is true, $t_{n+1} = n + 1$ is also true.

What's next? (The recurrence relation is $t_n = t_{n-1} + 1$)

- Plug $n + 1$ into the recurrence relation:
  - $$\begin{aligned} t_{n+1} &= t_{(n+1)-1} + 1 \\ &= t_n + 1 \\ &= n + 1 \quad \text{(Substitute } n \text{ for } t_n \text{ by the induction hypothesis)} \end{aligned}$$

Inductive proof complete. We solved the recurrence relation!

Recurrence Relation #2

Consider the following recurrence relation:

$$t_n = t_{n/2} + 1 \qquad \text{for } n > 1,\ n \text{ is a power of } 2$$
$$t_1 = 1$$

- We are given the initial condition, $t_1 = 1$
- The first step is to **expand**:

$$t_2 = t_{2/2} + 1 = t_1 + 1 = 1 + 1 = 2$$

Do we write out t_3 next?

Recurrence Relation #2

Consider the following recurrence relation:

$t_n = t_{n/2} + 1$ for $n > 1$, n is a power of 2
$t_1 = 1$

- We are given the initial condition, $t_1 = 1$
- The first step is to **expand**:

$t_2 = t_{2/2} + 1 = t_1 + 1 = 1 + 1 = 2$

Do we write out t_3 next? **No!** Since n is a power of 2:

- $t_4 = t_{4/2} + 1 = t_2 + 1 = 3$
- $t_8 = t_{8/2} + 1 = t_4 + 1 = 4$
- $t_{16} = t_{16/2} + 1 = t_8 + 1 = 5$

Recurrence Relation #2

- $t_2 = t_{2/2} + 1 = t_1 + 1 = 1 + 1 = 2$
- $t_4 = t_{4/2} + 1 = t_2 + 1 = 2 + 1 = 3$
- $t_8 = t_{8/2} + 1 = t_4 + 1 = 3 + 1 = 4$
- $t_{16} = t_{16/2} + 1 = t_8 + 1 = 4 + 1 = 5$

The second step is to **guess**. What pattern emerges?

Recurrence Relation #2

- $t_2 = t_{2/2} + 1 = t_1 + 1 = 1 + 1 = 2$
- $t_4 = t_{4/2} + 1 = t_2 + 1 = 2 + 1 = 3$
- $t_8 = t_{8/2} + 1 = t_4 + 1 = 3 + 1 = 4$
- $t_{16} = t_{16/2} + 1 = t_8 + 1 = 4 + 1 = 5$

The second step is to **guess**. What pattern emerges? $t_n = \lg n + 1$ since:

$\lg 2 = 1$ and $t_2 = 1 + 1 = 2$

...

$\lg 16 = 4$ and $t_{16} = 4 + 1 = 5$

Our **guess** is:

$t_n = \lg n + 1$ for $n > 0$ and n is a power of 2

Recurrence Relation #2

The third step is to **verify** our guess with induction:

Induction base: $t_1 = 1 = \lg 1 + 1$ (since $\lg 1 = 0$)
Induction hypothesis (guess): $t_n = \lg n + 1$ for $n > 0$ and n is a power of 2
Induction step: show that if the hypothesis is true, $t_{2n} = \lg(2n) + 1$ is also true

In the induction step the next case is $2n$, not $n + 1$. This is because the recurrence states n must be a power of 2 so $2n$ comes after n.

What's next?

Recurrence Relation #2

The third step is to **verify** our guess with induction:

Induction base: $t_1 = 1 = \lg 1 + 1$ (since $\lg 1 = 0$)
Induction hypothesis (guess): $t_n = \lg n + 1$ for $n > 0$ and n is a power of 2
Induction step: show that if the hypothesis is true, $t_{2n} = \lg(2n) + 1$ is also true

In the induction step the next case is $2n$, not $n + 1$. This is because the recurrence states n must be a power of 2 so $2n$ comes after n.

What's next? Plug $2n$ into the original recurrence relation $t_n = t_{n/2} + 1$:

$$t_{2n} = t_{(2n)/2} + 1 = t_n + 1 = (\lg n + 1) + 1$$

How do we transform $(\lg n + 1) + 1$ to $\lg(2n) + 1$?

Recurrence Relation #2

How do we transform $(\lg n + 1) + 1$ to $\lg(2n) + 1$?

- $\lg 2 = 1$, so we can substitute $\lg 2$ in for one of the 1s:
  - $(\lg n + \lg 2) + 1$
- Remember this important logarithm fact: $\lg x_1 + \lg x_2 = \lg(x_1 * x_2)$
  - Therefore, $(\lg n + \lg 2) + 1 = \lg(2n) + 1$

Proof complete!

In-Class Exercise

Solve the following recurrence relation:

1. $t_1 = 2$

   $t_n = 2t_{n-1} \quad n \geq 2$