Induction and Recursion
Recursive Algorithms

March 18, 2016
Outline
Recursive Algorithms

Definition

An algorithm is recursive if it solves a problem by reducing it into one or more instances of the same problem with smaller input.
Recursive Algorithms

Definition

An algorithm is recursive if it solves a problem by reducing it into one or more instances of the same problem with smaller input.

Example (Recursive algorithm for $n$!)

```c
int factorial(int n) {
}
```
Recursive Algorithms

Definition

An algorithm is recursive if it solves a problem by reducing it into one or more instances of the same problem with smaller input.

Example (Recursive algorithm for $n!$)

```c
int factorial(int n) {
    if (n == 0) return 1;
    return n * factorial(n - 1);
}
```
Recursive Algorithms

Fibonacci Number

Example (Recursive algorithm for Fibonacci numbers)

```c
int fib(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n - 2) + fib(n - 1);
}
```
Recursive Algorithms

Palindromes

What is a palindrome?
What is a palindrome?

**Definition**

A *palindrome* is a string which reads the same from left-to-right as it does from right-to-left.

(There is often a relaxation in spacing/punctuation for multi-word phrases. Thus “Madam, I’m Adam” is considered a palindrome. We will ignore this *simplification.*)
Recursive Algorithms

Palindromes

What is a palindrome?

Definition

A palindrome is a string which reads the same from left-to-right as it does from right-to-left.

(There is often a relaxation in spacing/punctuation for multi-word phrases. Thus “Madam, I’m Adam” is considered a palindrome. We will ignore this simplification.)

Example

“A man, a plan, a canal, Panama!”
“aabaa”
“beaddaeb”
Recursive Algorithms

Palindromes

Is that definition helpful when considering *recursive* algorithms?
Recursive Algorithms

Palindromes

Is that definition helpful when considering recursive algorithms? Is there an alternative, recursive definition of a palindrome?
Recursive Algorithms
Palindromes

Is that definition helpful when considering *recursive* algorithms? Is there an alternative, recursive definition of a palindrome?

**Definition**

A palindrome across the alphabet $\Sigma$ has one of three forms:

$$
\begin{cases}
\lambda & \text{the empty string} \\
\sigma & \sigma \in \Sigma \\
\sigma \text{ palindrome } \sigma & \text{same starting and ending character around a palindrome}
\end{cases}
$$
Recursive Algorithms

Palindrome Algorithm

Example (Recursive algorithm for palindrome)

```java
boolean palindrome(String str) {
    if (str.length() <= 1)
        return true;
    if (str.firstChar() != str.lastChar())
        return false;
    return palindrome(str.middleSubstr());
}
```
Definition (Pig Latin Rules)
A word is converted to Pig Latin according to the rules:

\[
\begin{align*}
\text{tail head ay} & \quad \text{head is a consonant (or blend).} \\
\text{word ay} & \quad \text{otherwise.}
\end{align*}
\]

Example

<table>
<thead>
<tr>
<th>English</th>
<th>Pig Latin</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>dogay</td>
</tr>
<tr>
<td>otter</td>
<td>otteray</td>
</tr>
<tr>
<td>spaghetti</td>
<td>spaghettiay</td>
</tr>
</tbody>
</table>
Recursive Algorithms

Ig-pay Atin-lay

Definition (Pig Latin Rules)

A word is converted to Pig Latin according to the rules:

\[
\begin{align*}
\text{tail head } ay & \quad \text{head is a consonant (or blend).} \\
\text{word } ay & \quad \text{otherwise.}
\end{align*}
\]

Example

<table>
<thead>
<tr>
<th>English</th>
<th>Pig Latin</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>ogday</td>
</tr>
<tr>
<td>otter</td>
<td></td>
</tr>
<tr>
<td>spaghetti</td>
<td></td>
</tr>
</tbody>
</table>
Definition (Pig Latin Rules)

A word is converted to Pig Latin according to the rules:

\[
\begin{cases}
\text{tail head ay} & \text{head is a consonant (or blend).} \\
\text{word ay} & \text{otherwise.}
\end{cases}
\]

Example

<table>
<thead>
<tr>
<th>English</th>
<th>Pig Latin</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>ogday</td>
</tr>
<tr>
<td>otter</td>
<td>otteray</td>
</tr>
<tr>
<td>spaghetti</td>
<td></td>
</tr>
</tbody>
</table>
Recursive Algorithms
Ig-pay Atin-lay

Definition (Pig Latin Rules)
A word is converted to Pig Latin according to the rules:

\[
\begin{align*}
tail \text{ head } ay & \quad \text{head is a consonant (or blend).} \\
\text{word } ay & \quad \text{otherwise.}
\end{align*}
\]

Example

<table>
<thead>
<tr>
<th>English</th>
<th>Pig Latin</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>ogday</td>
</tr>
<tr>
<td>otter</td>
<td>otteray</td>
</tr>
<tr>
<td>spaghetti</td>
<td>aghettispay</td>
</tr>
</tbody>
</table>
Example (Recursive algorithm for Pig Latin)

```java
String pigLatin(String w) {
    // base case
    // recursive case
}
```
Recursive Algorithms

Pig Latin

Example (Recursive algorithm for Pig Latin)

```java
String pigLatin(String w) {
    // base case
    if (w.firstChar().isVowel())
        return w + "ay";
    // recursive case
}
```
Example (Recursive algorithm for Pig Latin)

```java
String pigLatin(String w) {
    // base case
    if (w.firstChar().isVowel())
        return w + "ay";
    // recursive case
    return pigLatin(w.restOfChar() + w.firstChar());
}
```
Recursive Algorithms

Euclid’s Algorithm

Example

```c
int gcd(non-negative int a, b with a > b) {
    // base case
    if (b == 0) return a;
    // recursive case
    return gcd(b, a % b);
}
```
Recursive Algorithms

Linear Search

What would a recursive linear search method look like?
Recursive Algorithms

Linear Search

What would a recursive linear search method look like?

What are the specifications of the problem?
Recursive Algorithms

Linear Search

What would a *recursive* linear search method look like?

What are the *specifications* of the problem?

Given an array, $A$, filled with $n$ values, a value to search for, $v$, and a starting index, $i$, determine if $A$ contains $v$ at position $i$ or higher. Return first matching index or -1 to indicate failure.
Example (Recursive linear search)

```java
int match(Object[] A, int n, Object v, int i) {
    // base case(s)
    if (i == n) return -1;
    if (A[i] == v) return i;
    // recursive case
    return match(A, n, v, i + 1);
}
```
Recursive Algorithms

Search a Linked List

Definition

```java
class ListNode {
    public Object data;
    public ListNode next;
}
```
Recursive Algorithms

Search a Linked List

Example

```java
ListNode match(ListNode head, Object v) {
    // base case(s)

    // recursive case
}
```
Recursive Algorithms

Search a Linked List

Example

```java
ListNode match(ListNode head, Object v) {
    // base case(s)
    if (head == null) return null;

    // recursive case
}
```
Recursive Algorithms

Search a Linked List

Example

```java
ListNodes match(ListNode head, Object v) {
    // base case(s)
    if (head == null) return null;
    if (head.data == v) return head;
    // recursive case
}
```
Recursive Algorithms

Search a Linked List

Example

```java
ListNode match(ListNode head, Object v) {
    // base case(s)
    if (head == null) return null;
    if (head.data == v) return head;
    // recursive case
    return match(head.next, v);
}
```