Week 6: Recursive Algorithms

CSC 236:Introduction to the Theory of Computation

Summer 2024

Instructor: Lily

Announcement

- Tutorial BA 1130 *only*
  - Tutorials now focus on more examples
- Midterm details
  - Multiple Choice and True/False: +1 if answer completely correctly, -1 if answer incorrectly. Can be left blank. *Don't guess*.
  - Short answer: no justification required.
  - Other types: 20% IDK for entire question or part of a question.

Recursion

Iterative algorithms:

Fibonacci sequence 0, 1, 1, 2, 3, 5, ...

$$f_{n+2} = f_{n+1} + f_n$$

```
 def fib(n):
    # Pre: natural number n
    # Post: returns f_n
1. if n == 0:
2.    return 0
3. if n == 1:
4.    return 1
5. return fib(n-1) + fib(n-2)
```

From last lecture

```
def mult(a, b):
    # Pre: a and b are natural
number with at most n bits
    # Post: returns a*b
1   m = 0
2   count = 0
3   while count < b:
4     m += a
5     count += 1
6   return m
```

Elementary Multiplication

```
def mult_elementary(a, b):
    # Pre: a and b are natural
number with at most n bits
    # Post: returns a*b
1   m = 0
3   while b > 0:
4     m += a * (b % 10)
5     b //= 10 # int division
6   return m
```

Fast Multiplication (Karatsuba's Algorithm)

```
def karatsuba(a, b):
   # Pre: a and b are natural
number with at most n digits
   # Post: returns a*b
if |a| == 1 and |b| == 1:
  return a*b
a1, a2 = a[0..n/2], a[n/2..n]
b1, b2 = b[0..n/2], b[n/2..n]
p1 = karatsuba(a1, b1)
p2 = karatsuba(a1+a2, b1+b2)
p3 = karatsuba(a2, b2)
return p1*10^{n} + (p2-p1-
p3)*10^{n/2} + p3
```

Now your turn!

1. When computing f_n using the algorithm shown to the right how many times does `fib(k)` get called for $0 \leq k \leq n$?
2. Use Karatsuba's algorithm to multiply together the number 1024 and 1729.

```
def fib(n):
    # Pre: natural number n
    # Post: returns f_n
1. if n == 0:
2.     return 0
3. if n == 1:
4.     return 1
5. return fib(n-1) + fib(n-2)
```

Q1. When computing f_n using the algorithm shown to the right how many times does `fib(k)` get called for $0 \leq k \leq n$?

```
def fib(n):
    # Pre: natural number n
    # Post: returns f_n
1.  if n == 0:
2.      return 0
3.  if n == 1:
4.      return 1
5.  return fib(n-1) + fib(n-2)
```

Q2. Use Karatsuba's algorithm to multiply together the number 1024 and 1729.

```
def karatsuba(a, b):
   # Pre: a and b are natural
number with at most n digits
   # Post: returns a*b
if |a| == 1 and |b| == 1:
  return a*b
a1, a2 = a[0..n/2], a[n/2..n]
b1, b2 = b[0..n/2], b[n/2..n]
p1 = karatsuba(a1, b1)
p2 = karatsuba(a1+a2, b1+b2)
p3 = karatsuba(a2, b2)
return p1*10^{n} + (p2-p1-
p3)*10^{n/2} + p3
```

Fast Multiplication (Karatsuba's Algorithm)

```
def karatsuba(a, b):
   # Pre: a and b are natural
number with at most n digits
   # Post: returns a*b
if |a| == 1 and |b| == 1:
  return a*b
a1, a2 = a[0..n/2], a[n/2..n]
b1, b2 = b[0..n/2], b[n/2..n]
p1 = karatsuba(a1, b1)
p2 = karatsuba(a1+a2, b1+b2)
p3 = karatsuba(a2, b2)
return p1*10^{n} + (p2-p1-
p3)*10^{n/2} + p3
```

Quick Sort

```
def partition(A, i, j):
  # Pre: i, j indices of A (i <= j)
  # Post: index p so that A[k] < A[p] for
  # k = i, …, p-1 and A[l] > A[p] for
  # l = p+1, …, j
  p = i
  pivot = A[j-1]
  for k in range(i, j-1):
    if pivot > A[k]:
      swap(A, p, k)
      p += 1
  swap(A, p, j-1)
  return p
```

```
def quicksort(A, i, j):
  # Pre: list A. i, j indices
  # Post: A is sorted
  if j-i <= 1:
    return
  p = partition(A, i, j)
  quicksort(A, i, p)
  quicksort(A, p+1, j)
```

```
def partition(A, i, j):
  # Pre: i, j indices of A (i <= j)
  # Post: index p so that A[k] < A[p] for
  # k = i, …, p-1 and A[l] > A[p] for
  # l = p+1, …, j
  p = i
  pivot = A[j-1]
  for k in range(i, j-1):
    if pivot > A[k]:
      swap(A, p, k)
      p += 1
  swap(A, p, j-1)
  return p

def quicksort(A, i, j):
  # Pre: i, j indices of A (i <= j)
  # Post: A is sorted
  if j-i <= 1:
    return
  p = partition(A, i, j)
  quicksort(A, i, p)
  quicksort(A, p+1, j)
```

Quick Sort (Worst Case)

```
def partition(A, i, j):
  # Pre: i, j indices of A (i <= j)
  # Post: index p so that A[k] < A[p] for
  # k = i, …, p-1 and A[l] > A[p] for
  # l = p+1, …, j
  p = i
  pivot = A[j-1]
  for k in range(i, j-1):
    if pivot > A[k]:
      swap(A, p, k)
      p += 1
  swap(A, p, j-1)
  return p
```

Recap

- Recursive vs iterative algorithms
- Classic Multiplication
- Karatsuba Multiplication
- Quick Sort

Next time… running time of recursive algorithms via the Master Method