CMPSC 40, Section 6

TA: Jiyu Zhang

Email: jiyuzhang at ucsb dot edu

November 11, 2020

To Jiyu: remember to record the section.

1 Schedule

If you feel you need clarifications for problems in HW5, please ask your questions in Piazza. This will help other students too.

Problem from last HW (HW4) - 10 min

Inductions - 5-10 min

Recursive Algorithms - 20-25 min

Q&A

2 Problem from HW4

Problem 6

Martin correctly multiplied two octal numbers, but forgot to write down the most significant digit d of the product. Can you help him?

$$(12345)_8 \times (54321)_8 = (d\,17743365)_8$$

Show your work.

- Most of you are using the classical multiplication algorithm you have learned in elementary school. What's the time complexity of the multiplication algorithm?

$$\begin{array}{ccccccccc}
 & & & & 1 & 2 & 3 & 4 & 5 \\
\times & & & & 5 & 4 & 3 & 2 & 1 \\
\hline
 & & & & 1 & 2 & 3 & 4 & 5 \\
 & & & 1 & 4 & 7 & 1 & 2 & \\
 & & 3 & 6 & 2 & 5 & 7 & & \\
 & 5 & 1 & 6 & 2 & 4 & & & \\
6 & 4 & 1 & 7 & 1 & & & & \\
\hline
7 & 1 & 7 & 7 & 4 & 3 & 3 & 6 & 5
\end{array}$$

- Essentially, to multiply any two n-digit numbers, you'll write down n-rows of $n/(n+1)$-digit numbers.
- If addition/multiplication of two single digits (in 0-8) uses 1 step, then the time complexity is $O(n^2)$.

A better solution

- Now using modular arithmetic I can give an alternative and more "efficient" solution.
- Let's represent $(12345)_8$ as $1 \times 8^4 + 2 \times 8^3 + 3 \times 8^2 + 4 \times 8 + 5$. What happens if I compute its modulo 7?
- Since $8 \equiv 1 \mod 7$, it becomes $(1 \times 1 + 2 \times 1 + 3 \times 1 + 4 \times 1 + 5 \times 1) \mod 7 = (1+2+3+4+5) \mod 7$
- We have similar results for $(54321)_8$ and $(d17743365)_8$
- $(1+2+3+4+5) \times (5+4+3+2+1) \equiv (d+1+7+7+4+3+3+6+5) \mod 7$
- Finally we have $1 \equiv d+1 \mod 7$.
- $d = 7$.
- Time complexity?
- $3n + 1 = O(n)$.
- What about decimal multiplication? $\mod 9$.
- In general?

3 Inductions

Find a formula for $\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\cdots+\frac{1}{(n-1)n}$. Prove it by induction.

Solution.

$1-\frac{1}{n}$

- Base case: $f(1)=1-1/2$.
- Inductive assumptions: assume it holds for $f(n)$. i.e. $f(n)=1-\frac{1}{n}$
- Inductive step: $f(n)+\frac{1}{n(n+1)}=1-\frac{1}{n}+\frac{1}{n}-\frac{1}{n+1}=1-\frac{1}{n+1}=f(n+1)$
- By mathematical induction, we have proved that our conjecture is true.

Alternative proof?

$\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\cdots+\frac{1}{(n-1)n}=1-1/2+1/2-1/3+\cdots+1/(n-1)-1/n=1-1/n.$

4 Recursive Algorithms

4.1 Example from the textbook

Fibonacci sequence is defined recursively:

$f(0) = 0, f(1) = 1$
$f(n+2) = f(n) + f(n+1)$

Question: find $f(6)$

- $f(2) = 1$
- $f(3) = 1 + 1 = 2$
- $f(4) = 1 + 2 = 3$
- $f(5) = 2 + 3 = 5$
- $f(6) = 3 + 5 = 8$

Question: Write a computer program (algorithm) to solve $f(n)$ (recursively).

- **PROCEDURE** $f(n)$:
- Input is n, output is $f(n)$, the n-th Fibonacci number.
- check if $n \geq 0$, if not, output "undefined"
- //Base Cases:
- if $n = 0$, output 0; if $n = 1$, output 1.
- //Recursive Steps:
- otherwise, output $f(n-1) + f(n-2)$

How it works?

- $f(6)$? Find $f(5)$ and $f(4)$ first!
- $f(5)$? Find $f(4)$ and $f(3)$ first.
- ...
- $f(0)$?, 0; $f(1)$?, 1.

Deficiency?

Note. One deficiency is that $f(4)$ is computed twice! For $f(6)$ we computed $f(4)$, for $f(5)$ we computed $f(4)$.

An iterative approach is already above!

4.2 Euclidean Algorithm

Goal: Find the gcd of a, b.

Fact. if c divides a and b, then it divides $a \mod b$. i.e. For $a = m \cdot b + n$, c divides n.

Question: suppose $f(a, b)$ outputs the gcd of a, b, what's the recursive relation here?

$f(a, b) = f(b, a \mod b)$

Question: can you write an algorithm (in pseudo code) for finding the gcd of a, b?

WLOG, assume $a \geq b$

- **PROCEDURE** $f(a, b)$
- Input is a, b, output is $f(a, b)$
- What's the base cases?
- What's the recursive steps?

- **PROCEDURE** $f(a, b)$
- Input is a, b, output is $f(a, b)$
- Base Cases:
- if $b = 0$ output a
- Recursive Steps:
- otherwise, compute and output $f(b, a \mod b)$

Exercise

Write a recursive procedure (algorithm) for finding Bezout's coefficients.

This algorithm is called **the extended Euclidean algorithm**