Chapter 4 notes

Diego R.R.

October 16, 2023

Chapter 1

Algorithms Ch

1.1 Algorithms

1.1.1 Problem Section (book)

Exercise 1. 49

Exercise 2. 51

function BINARY_INSERTION_SORT($(a_1, a_2, \ldots, a_n)$: list of numbers)
    $left := 1$
    $right := n$
    while $left_i dx < right_i dx$ **do**
        $mid := \lfloor \frac{left+right}{2} \rfloor$
        if $a_{mid} < a_{mid+1}$ **then**
            $left_i dx := mid$
        else
            $right_i dx := mid_i dx$
    $insert\_idx := left_i dx$
    $insert\_val := a_{insert\_idx}$
    $idx := insert\_idx - 1$
    while $idx \geq 1$ and $a_{idx} > insert\_val$ **do**
        $a_{idx+1} := a_{idx}$
        $idx := idx - 1$
    $a_{idx+1} := insert\_val$
    return $(a_1, a_2, \ldots, a_n)$

Chapter 2

Divisibility and Modular Arithmetic

Definition 1. Let a and b be integers with $a \neq 0$. We say that a **divides** b if there exists an integer c such that $b = ac$ (or equivalently, if $\frac{b}{a}$ is an integer). IWhen a divides b we say that a is a **factor** or **divisor** of b, and that b is a **multiple** of a. The notation $a \mid b$ denotes that a divides b. We write $a \nmid b$ to denote that a does not divide b.

Theorem 1. Let a, b, and c be integers. Where $a \neq 0$. Then

(j) if $a \mid b$ and $a \mid c$, then $a \mid (b + c)$.

(j) if $a \mid b$, then $a \mid (bc)$ for all integers c.

(j) if $a \mid b$ and $b \mid c$, then $a \mid c$.

Corollary 1. If a, b, and c are integers, where $a \neq 0$, such that $a \mid b$ and $a \mid c$, then $a \mid (mb+nc)$ whenever m and n are integers.

Theorem 2 (Division Algorithm)**.** Let a be an integer and d a positive integer. Then there are unique integers q and r, with $0 \leq r < d$, such that $a = dq + r$.

Definition 2. From the division algorithm, d is the *divisor* and a is the *dividend*, q is called the *quotient* and r is called the *remainder*. This notation is used to express the quotient and remainder:

$$q = a \textbf{ div } d, \quad r = a \textbf{ mod } d$$

Definition 3. If a and b are integers and m is a positive integer, then a is *congruent to* b *modulem* if m divides $a - b$. We write $a \equiv b \pmod{m}$ to denote that a is congruent to b modulo m. We say that $a \equiv b \pmod{m}$ is a **congruence** and m is its **modulus** (plural **moduli**). If a and b are not congruent modulo m, we write $a \not\equiv b \pmod{m}$.

Theorem 3. Let a and b be integers, and let m be a positive integer. Then $a \equiv b \pmod{m}$ if and only if $a \pmod{m} = b \pmod{m}$.

Theorem 4. Let m be a positive integer. The Integers a and b are congruent modulo m if and only if there is an integer k such that $a = b + km$.

Theorem 5. Let m be a positive integer. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then

$$a + c \equiv b + d \pmod{m} \qquad ac \equiv bd \pmod{m}.$$

Corollary 2. Let m be a positive integer and let a, b be integers. Then

$$(a + b) \pmod{m} = ((a \pmod{m}) + (b \pmod{m})) \pmod{m}$$

and

$$(ab) \pmod{m} = ((a \pmod{m})(b \pmod{m})) \pmod{m}$$

2.1 Integer Representations and Algorithms

Theorem 1. Let b be an integer greater than 1. Then if n is a positive integer, it can be expressed uniquely in the form

$$n = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0$$

where k is a nonnegative integer, $a_0, a_1, \ldots, a_k$ are nonnegative integers less than b, and $a_k \neq 0$.

Algorithm 1. **function** BASE B EXPANSION(n, b: positive integers with $b > 1$)

$q := n$
$k := 0$
while $q \neq 0$ **do**
  $a_k := q \pmod{b}$
  $q := q \ (\text{div b})$
  $k := k + 1$

return $(a_{k-1}a_{k-2}\cdots a_1a_0)$

Algorithm 2. **function** ADD(a, b: nonnegative integers, where $a = (a_{n-1}a_{n-2}\cdots a_1a_0)_2$ and $b = (b_{n-1}b_{n-2}\cdots b_1b_0)_2$)

```
c := 0
for j := 0 to n − 1 do
    d := ⌊(a_i + b_i + c)/2⌋
    s_i := (a_i + b_i + c) − 2d
    c := d
s_n := c
return (s_n s_{n−1} ··· s_1 s_0)_2
```

Algorithm 3. **function** MULTIPLY(a, b: nonnegative integers, where $a = (a_{n-1}a_{n-2}\cdots a_1a_0)_2$ and $b = (b_{n-1}b_{n-2}\cdots b_1b_0)_2$)

```
for j := 0 to n − 1 do
    if b_i = 1 then
        c := a ≪ j
    else
        c := 0
                                        ▷ {c_0, c_1, ..., c_{n−1}} are the partial products]
    p := 0
    for j := 0 to n − 1 do
        p := add(p, c_j)
    return p
```

Algorithm 4. **function** DIVIDE(a: integer, d positive integer)

```
q := 0
r := |a|
while r ≥ d do
    r := r − d
    q := q + 1
if a < 0 and r > 0 then
    r := d − r
    q := −(q + 1)
                                        ▷ {q, r} q = a  (div d), r = a  (mod d)
return (q, r)
```

Algorithm 5. **function** MODULAR EXPONENTIATION(b: integer, $n = (a_{k-1}, a_{k-2}, \ldots, a_1, a_0)_2$, m: positive integer)

```
x := 1
power := b (mod m)
for i := 0 to k − 1 do
```

```
        if a_i = 1 then
            x := (x · power) (mod m)
        power := (power · power) (mod m)
    return x
```

Chapter 3

Counting

3.1 The Basics of Counting

Definition 1. If a procedure can be broken into 2 different tasks, and for each there are n_1 and n_2 ways to do it respectively, then there are $n_1 n_2$ ways to do the procedure.

3.1.1 Problem Section (book)

Exercise 1. Just use of the two rules.

1. use product rule
2. use sum rule

Exercise 2. For each 27 floor there are 37 offices. If you were to assign a person in a office there would be 27×37 ways to do it.

Exercise 3. There are 10 problems and 4 choices for each problem.

1. the student has 4 choices for each 10 question, then there are 4^{10} ways to do it.
2. the student has 5 choices for each 10 question, then there are $5^1 0$ ways to do it.

Exercise 4. Use extended product rule, there would be 12 colors, 2 female or male and 3 sizes. So there would be $12 \times 2 \times 3$ types of shirts.

Exercise 5. There would be 6×7 pairs of airlines.

Exercise 6. Same as 5. 4×5 auto routes.

Exercise 7. By extended product rule there would be $26 \times 26 \times 26$ ways to do it or 26^3.

Exercise 8. Each time one letter is chosen it substract 1 to the following choices to account for no repetition, then we would have $26 \times (26 - 1) \times (24 - 1 - 1)$ ways three consecutive non-repeated letters can be constructed.

Exercise 9. The same as 8 but with two letters, there would be $25 \times (25 - 1)$ ways to do it. **ASK PROF** it should be 676 and I get 600.

Exercise 10. There are 2^8 bit strings of length 8.

Exercise 11. There are 2^8 bit strings of length 10 that ends with both begin and end with a 1.

Exercise 12. There are $2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2 = 2^7 - 1$ bit strings of length 6 or less.

Exercise 13. There are n bit strings of 1s not exeding n. Not counting the empty string.

Exercise 14. There are 2^{n-2} bit strings of length n that begin and end with 1. Not counting the empty string.

Exercise 15. There are $26^4 + 26^3 + 26^2 + 26^1$ strings with lowercase letters of length 4 or less (not counting the empty string).

Exercise 16. There are $4 * 26^3 + 3 * 26^2 + 2 * 26^1 + 1$ strings. **ASK PROF**.