NUS National University of Singapore | Computing

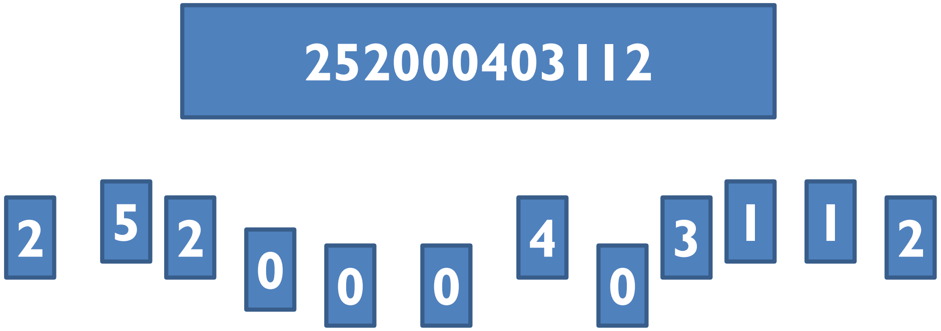
Programming Refresher Workshop

Session 6 Dr Henry Chia

Contents

- Breaking up a Number
  - Examples
    - NRIC validation
    - Checking 9-divisibility
    - Encrypting a number
  - Usage
    - Storing individual digits
    - Finding number of digits
    - Finding sum of digits
    - Finding largest pair of digits
- Constructing a Number
  - Forward/reverse construction

Breaking Up a Number

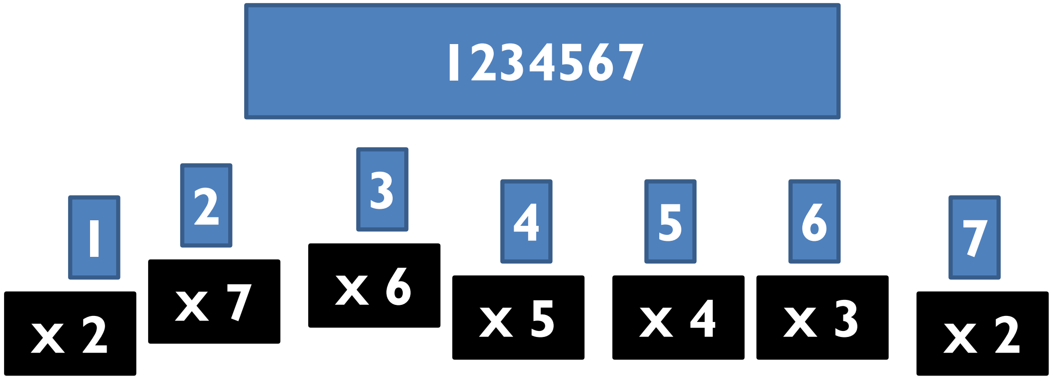


252000403112

2 5 2 0 0 0 4 0 3 1 1 2

- Common thing to do in number manipulation
- Know of any situations that you need to break up the number?

Example: Finding checksum of NRIC



1234567

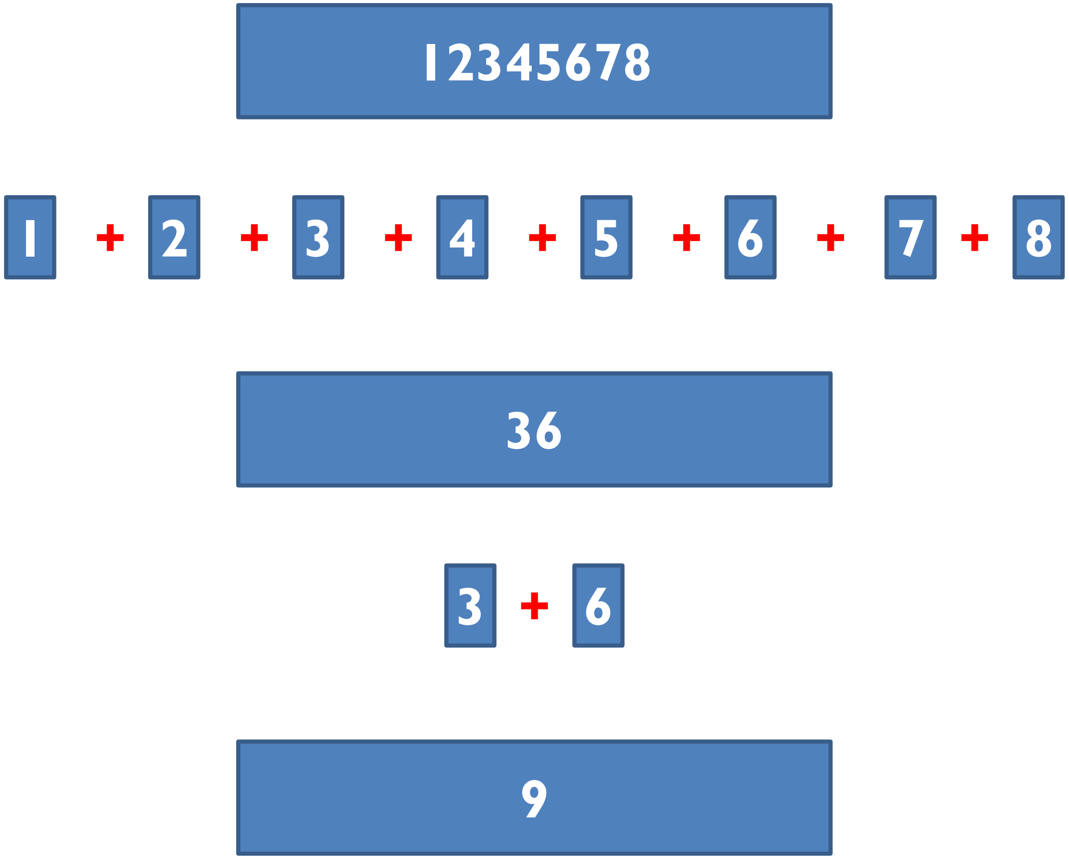
1 2 3 4 5 6 7

x 2 x 7 x 6 x 5 x 4 x 3 x 2

- Multiply each digit by a specific number
- Sum them up
- Divide by 11 to get the remainder
- Subtract the remainder from 11
- Match the table

1	2	3	4	5	6	7	8	9	10	11
A	B	C	D	E	F	G	H	I	Z	J

Example: Checking for 9-divisibility



12345678

1 + 2 + 3 + 4 + 5 + 6 + 7 + 8

36

3 + 6

9

Usage: Storing digits of a number

- Algorithm to store individual digits of a number

```
breakUp($n$):
  1. Initialize $i$ to 0, $arr$ to array of size 10
  2. while ($n > 0$)
        $r \leftarrow$ remainder of $n$ / 10
        $arr[i] \leftarrow r$
        increment $i$
        $n \leftarrow$ quotient of $n$ / 10
  3. Return $arr$.
```

#	*r*	*n*
0	?	45323
1	3	4532
2	2	453
3	3	45
4	5	4
5	4	0

What is the pre-condition of this algorithm?

Usage: Finding number of digits

- Algorithm to determine the number of digits

```
numDigits(n):
  pre: n > 0
  1. Initialize count to 0
  2. while (n > 0)
        r ← remainder of n / 10
        count ← count + 1
        n ← quotient of n / 10
  3. Return count.
```

#	*r*	*count*	*n*
0	?	0	45323
1	3	1	4532
2	2	2	453
3	3	3	45
4	5	4	4
5	4	5	0

Usage: Finding sum of digits (1/2)

- Iterative algorithm to sum up all digits

```
sumUp($n$):
  pre: $n$ >= 0
  1. Initialize $sum$ to 0
  2. while ($n > 0$)
        $r \leftarrow$ remainder of $n$ / 10
        $sum \leftarrow sum + r$
        $n \leftarrow$ quotient of $n$ / 10
  3. Return $sum$.
```

#	*r*	*sum*	*n*
0	?	0	45323
1	3	3	4532
2	2	5	453
3	3	8	45
4	5	13	4
5	4	17	0

Usage: Finding sum of digits (2/2)

- Recursive algorithm to sum up all digits

```
sumUpRec($n$):
  pre: $n >= 0$
  1. Initialize $sum$ to 0
  2. If ($n > 0$) then
         $r \leftarrow$ remainder of $n / 10$
         $n \leftarrow$ quotient of $n / 10$
         $sum \leftarrow$ sumUpRec($n$) + $r$
  3. Return $sum$.
```

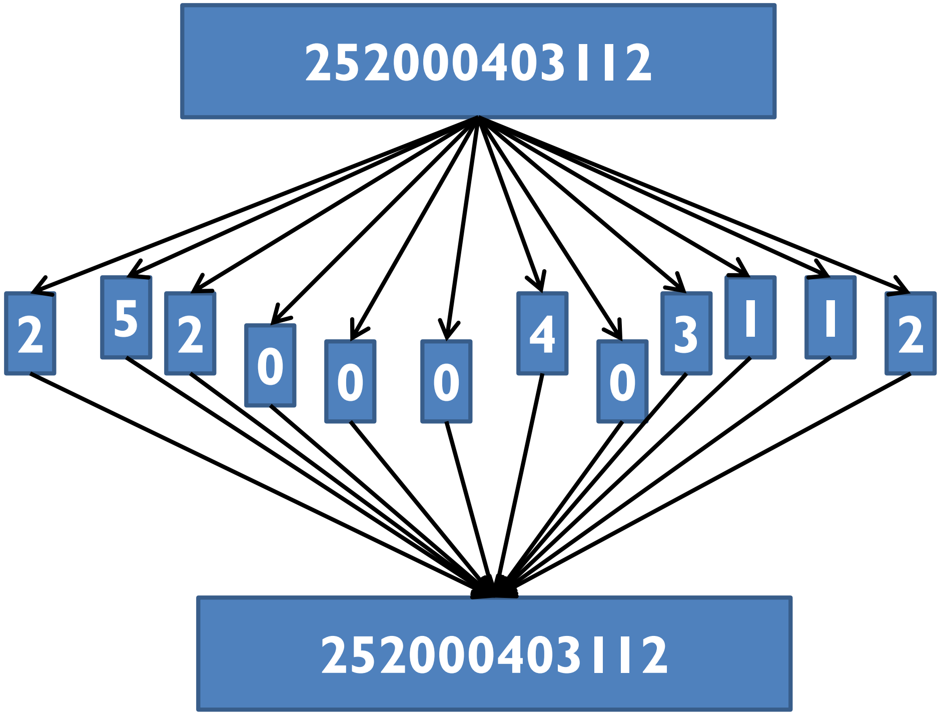
Usage: Finding largest pair of digits

- Algorithm to determine the largest pair of digits
  - 45323 = 045323 → maximum(04, 53, 23) = 53

```
maxPairs($n$):
  pre : $n >= 0$
  1. Initialize $max$ to 0
  2. while ($n > 0$)
        $r$ ← remainder of $n$ / 100
        if ($r > max$) then
             $max$ ← $r$
        $n$ ← quotient of $n$ / 100
  3. Return $max$.
```

#	*r*	*max*	*n*
0	?	0	45323
1	23	23	453
2	53	53	4
3	4	53	0

Break up, then put back (1/3)



Break up, then put back (2/3)

4	0	3	1	1	2
…	…	thousands	hundreds	tens	ones
10^5	10^4	10^3	10^2	10^1	10^0

$$2*10^0 + 1 * 10^1 + 1 * 10^2 + 3 * 10^3 + 0 * 10^4 + 4 * 10^5$$

$$= 400\ 000 + 0 + 3\ 000 + 100 + 10 + 2$$
$$= 403\ 112$$

Break up, then put back (3/3)

- Algorithm to break up and get back the same number

```
sameNum($n$):
  pre: $n$ >= 0
  1. Initialize $acc$ to 0, $p$ to 0
  2. while ($n > 0$)
        $r \leftarrow$ remainder of $n / 10$
        $acc \leftarrow acc + r * 10^p$
        $p \leftarrow p + 1$
        $n \leftarrow$ quotient of $n / 10$
  3. Return $acc$.
```

#	*r*	*acc*	*p*	*n*
0	?	0	0	45323
1	3	3	1	4532
2	2	23	2	453
3	3	323	3	45
4	5	5323	4	4
5	4	45323	5	0

Break up, then put back in reverse (1/2)

4	0	3	1	1	2
…	…	thousands	hundreds	tens	ones
10^0	10^1	10^2	10^3	10^4	10^5

Horner's Method

$$2*10^5 + 1 * 10^4 + 1 * 10^3 + 3 * 10^2 + 0 * 10^1 + 4 * 10^0$$

$$\boxed{= ((((2 *10 + 1)*10 +1) * 10 +3) * 10 + 0) * 10 + 4}$$

$$= (((((0 * 10) + 2) *10 + 1)*10 +1) * 10 +3) * 10 + 0) * 10 + 4$$

$$= 200\,000 + 10\,000 + 1\,000 + 300 + 0 + 4$$

$$= 211304$$

Break up, then put back in reverse (2/2)

- Algorithm to break up and get back the reverse

```
revNum(n):
  pre: n >= 0
  1. Initialize acc to 0
  2. while (n > 0) then
         r ← remainder of n / 10
         acc ← acc * 10 + r
         n ← quotient of n / 10
  3. Return acc.
```

#	*r*	*acc*	*n*
0	?	0	45323
1	3	3	4532
2	2	32	453
3	3	323	45
4	5	3235	4
5	4	32354	0

Exercise

- How to return the three most significant digits in a number?
  - 45323 → 453
  - 827 → 827
  - 92 → 92

Final Notes

- Start with an iterative solution in this afternoon’s exercises, and whenever possible, write an equivalent solution using recursion.

Thanks!

- Thanks for attending this workshop. We hope it has been useful to you.
- See you when the semester starts!