哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

数据结构
Data Structures

Chapter 7 Strings

Prof. Yitian Shao
School of Computer Science and Technology

Strings

- A **string** is a (possibly empty) **sequence of characters**, is **mutable** (can be changed) in C++
- There are two types of strings in C++: C strings (**char arrays**) and C++ strings (**string objects**)
- A string literal such as “hello world" is a C string
- Converting between the two types:
  - `string("text")` **C** string to **C++** string
  - `string.c_str()` **C++** string to **C** string

Previously in High-level Language Programming

- **C++ strings defined by a class**
- **The string data type is not built into C++**

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string password = "secret";
    string user_input;
    cout << "Enter Password: ";
    cin >> user_input;
    if ( password == user_input )
        cout << "Correct password. Welcome to the system ..." << endl;
    else
        cout << "Invalid password" << endl;
    return 0;

}
```

- C++ strings are compared using **==, !=, <, <=, >, >=** instead of *strcmp*() in C-strings

Previously in High-level Language Programming

- **C++ string concatenation**

```
string str1 = “Hello”, str2 = “World”;
```

```
str3 = str1 + str2;   ⟶ str3  “HelloWorld”
```

```
str1.append(str3)     ⟶ str1  “HelloHelloWorld”
```

Previously in High-level Language Programming

- **C++ string swap**

```
string str1 = “Hello”, str2 = “World”;
```

```
str1.swap(str2);
```

str1 “World” **str2** “Hello”

Previously in High-level Language Programming

- **C++ Character classification**
- The following functions return a true (non-zero integer) value or a false (zero integer) value depending on whether or not the character belongs to a particular set of characters.
- Covert the case of a character: **tolower**() and **toupper**()

Function	Character set
`isalnum`	Alphanumeric character: A-Z, a-z, 0-9
`isalpha`	Alphabetic character: A-Z, a-z
`isascii`	ASCII character: ASCII codes 0-127
`iscntrl`	Control character: ASCII codes 0-31 or 127
`isdigit`	Decimal digit: 0-9
`isgraph`	Any printable character other than a space
`islower`	Lowercase letter: a-z
`isprint`	Any printable character, including a space
`ispunct`	Any punctuation character
`isspace`	Whitespace character: \t,\v,\f,\r,\n or space ASCII codes 9-13 or 32
`isupper`	Uppercase letter: A-Z
`isxdigit`	Hexadecimal digit: 0-9 and A-F

Function	Purpose
`tolower`	Converts an uppercase character to lowercase.
`toupper`	Converts a lowercase character to uppercase.

Previously in High-level Language Programming

- **C++ string pattern searching**

```
                  0 1 2 3 4 5 6 7 8
string s = "Welcome to Data Structure!";
string sub = "to";
cout << s.find(sub); // (Print out) 8
string sub2 = "hello";
if (s.find(sub2) != string::npos)
      cout << “Not found”;
```

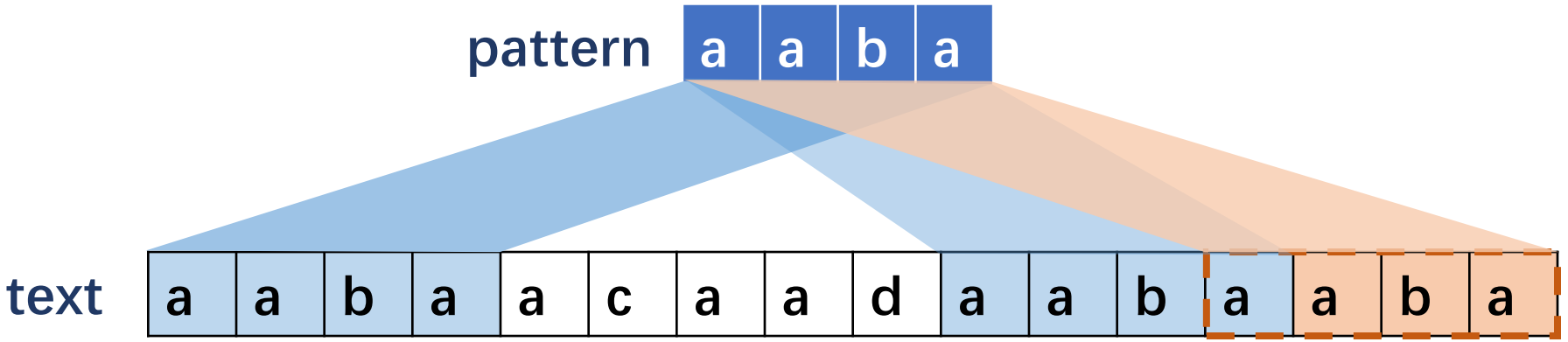
`string::npos` is returned if no substring can be found

Pattern Searching Algorithm for Strings

- How to implement the **`find()`** function?
- How to find all matched patterns in the text?

Pattern Searching Example

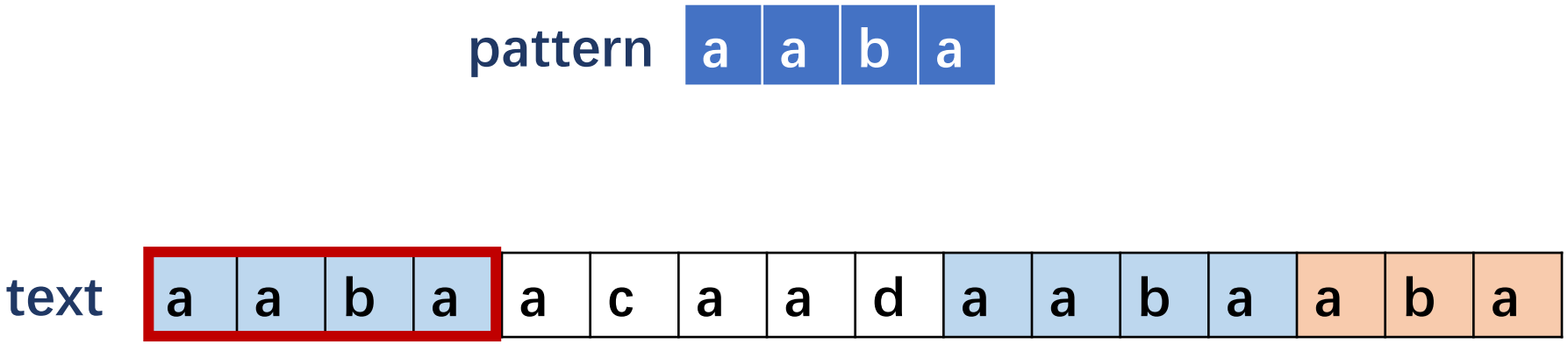
- string text = “aabaacaadaabaaba”, pattern = “aaba”;



- Answer: [0, 9, 12]

Brute Force Searching

- Slide the pattern over text **one by one** and check for a match. If a match is found, then slide by one character again to check for subsequent matches



In-Class Exercise

- Implement the brute force searching using C++

```
void search(string& pat, string& txt) {

    for(        ){  // Slide pattern window by 1 step repetitively


        for(        ){  // Check for pattern match for each i



        if(         ) {  // If pattern matches at index i
            cout << "Pattern found at index " << i << endl;
        }
    }
}
```

Exercise 7.1

- Complete LeetCode 3248

3248. Snake in Matrix

Easy | Topics | Companies | Hint

There is a snake in an `n x n` matrix `grid` and can move in **four possible directions**. Each cell in the `grid` is identified by the position: `grid[i][j] = (i * n) + j`.

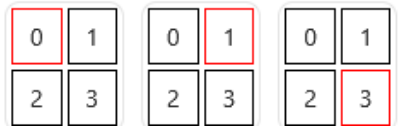
The snake starts at cell 0 and follows a sequence of commands.

You are given an integer `n` representing the size of the `grid` and an array of strings `commands` where each `command[i]` is either `"UP"`, `"RIGHT"`, `"DOWN"`, and `"LEFT"`. It's guaranteed that the snake will remain within the `grid` boundaries throughout its movement.

Return the position of the final cell where the snake ends up after executing `commands`.

Example 1:

> **Input:** `n = 2, commands = ["RIGHT","DOWN"]`
>
> **Output:** `3`
>
> **Explanation:**



Exercise 7.2

- Complete LeetCode 682

682. Baseball Game

Easy | Topics | Companies

You are keeping the scores for a baseball game with strange rules. At the beginning of the game, you start with an empty record.

You are given a list of strings `operations`, where `operations[i]` is the i^{th} operation you must apply to the record and is one of the following:

- An integer `x`.
  - Record a new score of `x`.
- `'+'`.
  - Record a new score that is the sum of the previous two scores.
- `'D'`.
  - Record a new score that is the double of the previous score.
- `'C'`.
  - Invalidate the previous score, removing it from the record.

Return *the sum of all the scores on the record after applying all the operations.*

The test cases are generated such that the answer and all intermediate calculations fit in a **32-bit** integer and that all operations are valid.

Exercise 7.3

- Complete LeetCode 1684

1684. Count the Number of Consistent Strings

Easy | Topics | Companies | Hint

You are given a string `allowed` consisting of **distinct** characters and an array of strings `words`. A string is **consistent** if all characters in the string appear in the string `allowed`.

Return *the number of **consistent** strings in the array* `words`.

Example 1:

```
Input: allowed = "ab", words = ["ad","bd","aaab","baa","badab"]
Output: 2
Explanation: Strings "aaab" and "baa" are consistent since they only contain characters 'a' and 'b'.
```

Example 2:

```
Input: allowed = "abc", words = ["a","b","c","ab","ac","bc","abc"]
Output: 7
Explanation: All strings are consistent.
```