【第二十九周】经典匹配算法：KMP、Sunday 与 Shift-[AndOr] 算法

1.string_match

```
/*************************************************************************
  > File Name: 1.string_match.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
using namespace std;

int brute_force(const char *text, const char *pattern) {
    for (int i = 0; text[i]; ++i) {
        int flag = 1;
        for (int j = 0; pattern[j]; ++j) {
            if (text[i + j] == pattern[j]) continue;
            flag = 0;
            break;
        }
        if (flag) return i;
    }
    return -1;
}

#define TEST(func, s1, s2) { \
    printf("%s(\"%s\", \"%s\") = %d\n", #func, s1, s2, func(s1, s2)); \
}

int main() {
    char s1[100], s2[100];
    while (cin >> s1 >> s2) {
        TEST(brute_force, s1, s2);
    }
    return 0;
}
```

2.kmp

```
/*************************************************************************
  > File Name: 1.string_match.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
using namespace std;

int brute_force(const char *text, const char *pattern) {
    for (int i = 0; text[i]; ++i) {
        int flag = 1;
        for (int j = 0; pattern[j]; ++j) {
            if (text[i + j] == pattern[j]) continue;
            flag = 0;
            break;
        }
        if (flag) return i;
    }
    return -1;
}

void GetNext(const char *pattern, int *next) {
    next[0] = -1;
    for (int i = 1, j = -1; pattern[i]; ++i) {
        while (j != -1 && pattern[j + 1] - pattern[i]) j = next[j];
        if (pattern[j + 1] == pattern[i]) j += 1;
        next[i] = j;
    }
    return ;
}


int kmp(const char *text, const char *pattern) {
    int n = strlen(pattern);
    int *next = (int *)malloc(sizeof(int) * n);
    GetNext(pattern, next);
    for (int i = 0, j = -1; text[i]; i++) {
        while (j != -1 && text[i] - pattern[j + 1]) j = next[j];
        if (text[i] == pattern[j + 1]) j += 1;
        if (pattern[j + 1] == 0) return i - j;
    }
```

```
    return -1;
}

#define TEST(func, s1, s2) { \
    printf("%s(\"%s\", \"%s\") = %d\n", #func, s1, s2, func(s1, s2)); \
}

int main() {
    char s1[100], s2[100];
    while (cin >> s1 >> s2) {
        TEST(brute_force, s1, s2);
        TEST(kmp, s1, s2);
    }
    return 0;
}
```

3.sunday

```
/*************************************************************************
  > File Name: 1.string_match.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
using namespace std;

int brute_force(const char *text, const char *pattern) {
    for (int i = 0; text[i]; ++i) {
        int flag = 1;
        for (int j = 0; pattern[j]; ++j) {
            if (text[i + j] == pattern[j]) continue;
            flag = 0;
            break;
        }
        if (flag) return i;
    }
    return -1;
}

void GetNext(const char *pattern, int *next) {
    next[0] = -1;
    for (int i = 1, j = -1; pattern[i]; ++i) {
        while (j != -1 && pattern[j + 1] - pattern[i]) j = next[j];
```

```
        if (pattern[j + 1] == pattern[i]) j += 1;
        next[i] = j;
    }
    return ;
}

int GetNextJ(char ch, const char *pattern, int *next, int j) {
    while (j != -1 && ch - pattern[j + 1]) j = next[j];
    if (ch == pattern[j + 1]) j += 1;
    return j;
}

int kmp(const char *text, const char *pattern) {
    int n = strlen(pattern);
    int *next = (int *)malloc(sizeof(int) * n);
    GetNext(pattern, next);
    for (int i = 0, j = -1; text[i]; i++) {
        j = GetNextJ(text[i], pattern, next, j);
        if (pattern[j + 1] == 0) return i - j;
    }
    return -1;
}

int sunday(const char *text, const char *pattern) {
    #define BASE 256
    int n = strlen(text), m, last_pos[BASE];
    for (int i = 0; i < BASE; i++) last_pos[i] = -1;
    for (m = 0; pattern[m]; ++m) last_pos[pattern[m]] = m;
    for (int i = 0; i + m <= n; i += (m - last_pos[text[i + m]])) {
        int flag = 1;
        for (int j = 0; pattern[j]; ++j) {
            if (text[i + j] == pattern[j]) continue;
            flag = 0;
            break;
        }
        if (flag) return i;
    }
    return -1;
}

#define TEST(func, s1, s2) { \
    printf("%s(\"%s\", \"%s\") = %d\n", #func, s1, s2, func(s1, s2)); \
}

int main() {
    char s1[100], s2[100];
    while (cin >> s1 >> s2) {
        TEST(brute_force, s1, s2);
        TEST(kmp, s1, s2);
        TEST(sunday, s1, s2);
    }
    return 0;
}
```

4.shift_and

```
/*************************************************************************
  > File Name: 1.string_match.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
using namespace std;

int brute_force(const char *text, const char *pattern) {
    for (int i = 0; text[i]; ++i) {
        int flag = 1;
        for (int j = 0; pattern[j]; ++j) {
            if (text[i + j] == pattern[j]) continue;
            flag = 0;
            break;
        }
        if (flag) return i;
    }
    return -1;
}

void GetNext(const char *pattern, int *next) {
    next[0] = -1;
    for (int i = 1, j = -1; pattern[i]; ++i) {
        while (j != -1 && pattern[j + 1] - pattern[i]) j = next[j];
        if (pattern[j + 1] == pattern[i]) j += 1;
        next[i] = j;
    }
    return ;
}

int GetNextJ(char ch, const char *pattern, int *next, int j) {
    while (j != -1 && ch - pattern[j + 1]) j = next[j];
    if (ch == pattern[j + 1]) j += 1;
    return j;
}

int kmp(const char *text, const char *pattern) {
    int n = strlen(pattern);
    int *next = (int *)malloc(sizeof(int) * n);
    GetNext(pattern, next);
    for (int i = 0, j = -1; text[i]; i++) {
        j = GetNextJ(text[i], pattern, next, j);
        if (pattern[j + 1] == 0) return i - j;
    }
```

```
    return -1;
}

int sunday(const char *text, const char *pattern) {
    #define BASE 256
    int n = strlen(text), m, last_pos[BASE];
    for (int i = 0; i < BASE; i++) last_pos[i] = -1;
    for (m = 0; pattern[m]; ++m) last_pos[pattern[m]] = m;
    for (int i = 0; i + m <= n; i += (m - last_pos[text[i + m]])) {
        int flag = 1;
        for (int j = 0; pattern[j]; ++j) {
            if (text[i + j] == pattern[j]) continue;
            flag = 0;
            break;
        }
        if (flag) return i;
    }
    return -1;
}

int GetNextP(char ch, int *code, int p) {
    return (p << 1 | 1) & code[ch];
}

int shift_and(const char *text, const char *pattern) {
    int code[256] = {0};
    int n = 0;
    for (n = 0; pattern[n]; ++n) code[pattern[n]] |= (1 << n);
    int p = 0;
    for (int i = 0; text[i]; i++) {
        p = GetNextP(text[i], code, p);
        if (p & (1 << (n - 1))) return i - n + 1;
    }
    return -1;
}

#define TEST(func, s1, s2) { \
    printf("%s(\"%s\", \"%s\") = %d\n", #func, s1, s2, func(s1, s2)); \
}

int main() {
    char s1[100], s2[100];
    while (cin >> s1 >> s2) {
        TEST(brute_force, s1, s2);
        TEST(kmp, s1, s2);
        TEST(sunday, s1, s2);
        TEST(shift_and, s1, s2);
    }
    return 0;
}
```