KMP

Idea：`fail[]` 失配数组的应用充分利用了模式串本身的信息减少不必要的匹配，使复杂度降至线性。具体来说，`fail[i]` 指向模式串 `t[1...i-1]` 中最长公共前后缀的前缀的下一位，如此 `i` 匹配失败时跳至 `fail[i]` 就可以继续匹配。

注意，第 `t[1...i]` 的最长公共前后缀的长度为 `fail[i+1]-1`。

Complexity：$O(n+m)$，其中 n 和 m 分别时目标串和模式串的长度。

Code：

```
int fail[N];
void getFail(char t[], int lent){
    int i = 1, j = 0;
    fail[1] = 0;
    while(i <= lent){
        if(!j || t[i] == t[j])  fail[++i] = ++j;
        else    j = fail[j];
    }
}
void KMP(char s[], int lens, char t[], int lent){
    int i = 1, j = 1;
    while(i <= lens){
        if(!j || s[i] == t[j]){
            i++, j++;
            if(j == lent + 1){
                printf("%d\n", i - lent);
                j = fail[j];
            }
        }
        else    j = fail[j];
    }
}
```

Code（优化 `fail[]` 数组）：

```
int fail[N];
void getFail(char t[], int lent){
    int i = 1, j = 0;
    fail[1] = 0;
    while(i <= lent){
        if(!j || t[i] == t[j]){
            i++, j++;
            if(t[i] != t[j])    fail[i] = j;
            else    fail[i] = fail[j];
        }
        else    j = fail[j];
    }
}
void KMP(char s[], int lens, char t[], int lent){
    int i = 1, j = 1;
    while(i <= lens){
        if(!j || s[i] == t[j]){
            i++, j++;
            if(j == lent + 1){
                printf("%d\n", i - lent);
                j = fail[j];
            }
        }
        else    j = fail[j];
```

```
25      }
26  }
```