20190408-左士海-作业五

1. 计算下列串的 next 数组：

（1）"ABCDEFG"
（2）"AAAAAAAA"
（3）"BABBABAB"
（4）"AAAAAAB"
（5）"ABCABDAAABC"
（6）"ABCABDABEABCABDABF"
（7）"ABBACXY"

```
void callNext(string p, int next[]) {
    int len = p.length();
    next[0] = -1;
    int j = 0, k = -1;
    while (j < len - 1) {
        if (k == -1 || p[j] == p[k]) {
            k++;
            j++;
            next[j] = k;
        } else {
            k = next[k];
        }
    }
}
```

```
ABCDEFG: -1 0 0 0 0 0 0
AAAAAAAA: -1 0 1 2 3 4 5 6
BABBABAB: -1 0 0 1 1 2 3 2
AAAAAAB: -1 0 1 2 3 4 5
ABCABDAAABC: -1 0 0 0 1 2 0 1 1 1 2
ABCABDABEABCABDABF: -1 0 0 0 1 2 0 1 2 0 1 2 3 4 5 6 7 8
ABBACXY: -1 0 0 0 1 0 0
```

2. 要求输入两个字符串 s 和 t，统计 s 包含串 t 的个数。

利用 KMP 算法在字符串 s 中查找 t，在完成一次匹配时不退出，继续进行匹配。

```cpp
int subCnt(char * s, char * t) {
    int i = 0, j = 0, cnt = 0;
    int n = strlen(s), m = strlen(t);
    int next[100];
    callNext(t, next);
    while (i < n) {
        if (j == -1 || s[i] == t[j]) {
            ++i;
            ++j;
        } else {
            j = next[j];
        }
        if (j >= m) {
            cnt ++;
            j = 0;
        }
    }
    cout << cnt << endl;
    return cnt;
}
```

3. 编写从串 s 中删除所有与串 t 相同的子串的算法

```cpp
void del_sub(char *s, char *t) {
    int pos = 0;
    int n = strlen(s), m = strlen(t);
    while (pos != -1) {
        pos = KMPMatching(s + pos, t);
        int i = pos, j = pos + m;
        while (j < n) s[i++] = s[j++];
        s[i] = '\0';
    }
}
```

利用 KMP 算法在字符串 s 中查找 t 子串并返回其初始字符的索引，进行删除操作，接着从当前位置继续向后搜索，找到下一个匹配的位置。

4. 试给出求串 s 和串 p 的最大公共子串的算法

使用动态规划的思想，dp[i][j]代表串 s 的前 i 部分子串与串 p 前 j 部分子串的最大公共子串长度，对于:

dp[i][j] = dp[i - 1][j - 1] + 1, s[i] == p[j];

dp[i][j] = dp[i - 1][j - 1], s[i] != p[j];

```cpp
string longestComSub(char *s, char *p) {
    string lcs;
    int n = strlen(s), m = strlen(p);
    int dp[n+1][m+1];
    memset(dp, 0, sizeof(dp));
    int mmax = 0, pos;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            if (s[i] == p[j]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
                if (dp[i][j] > mmax) {
                    mmax = dp[i][j];
                    pos = i;
                }
            }
        }
    }
    for (int i = pos - mmax; i <= pos; ++i) {
        lcs += s[i];
    }
    return lcs;
}
```

5. 编写一个函数来颠倒单词在字符串里的出现顺序。例如，把字符串"Do or do not, there is no try. "转换为"try． no is there not do, or Do"。假设所有单词都以空格为分隔符，标点符号也当做字母来对待。请对你的设计思路做出解释，并对你的解决方案的执行效率进行评估。

```cpp
string reverseWords(char * s) {
    string str;
    int pos = 0;
    int len = strlen(s);
    int t = -1;
    while (pos != t) {
        t = pos;
        pos = (pos == 0) ? KMPMatching(s , " ") : KMPMatching(s + pos + 1, " ") + pos + 1;
        if (t == pos) break;
        cout << t << " " << pos << endl;
        for (int i = pos; i >= t; --i) {
            str = s[i] + str;
        }
    }
    for (int i = len-1; i > pos; --i) {
        str = s[i] + str;
    }
    return str;
}
```

因为本章学习了 KMP，所以此题使用 KMP 算法。KMP 算法时间复杂度为 O(m+n)。遍历的时间复杂的为 O(n)。

6. 设有三对角矩阵 $\boldsymbol{A}_{n\times n}$，将其按行优先顺序压缩存储于一维数组 $b[3*n-2]$中，使得 $a_{ij}=b[k]$，请用 k 表示 i，j 的下标变换公式。

```
int n = sizeof(a) / sizeof(a[0]);
    int sa[3 * n - 2] = { 0 };
    int k;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if(a[i][j] != 0) {
                k = 2 * i + j;
                if(!sa[k]) sa[k] = a[i][j];
            }
        }
    }
```

7. 若在矩阵 $\boldsymbol{A}_{mxn}$ 中存在一个元素 a_{ij} $(0 \leqslant i \leqslant m\text{-}1, 0 \leqslant j \leqslant n\text{-}1)$ 满足：a_{ij} 是第 i 行元素中最小值，且又是第 j 列元素中最大值，则称此元素值为该矩阵的一个马鞍点。假设以二维数组存储矩阵 $\boldsymbol{A}_{mxn}$，试编写求出矩阵中所有马鞍点的算法。

```
int a[5][5] = {
            {1, 5, 6, 3, 6},
            {5, 0, 6, 7, 7},
            {5, 6, 4, 5, 0},
            {7, 6, 5, 0 ,7},
            {4, 1, 2, 7, 3},
    };
    int n = sizeof(a) / sizeof(a[0]);
    int minRow[n] = {0};
    for (int i = 0; i < n; ++i) {
        minRow[i] = 0;
        for (int j = 0; j < n; ++j) {
            if (a[i][j] < a[i][minRow[i]]) {
                minRow[i] = j;
            }
        }
    }
    for (int i = 0; i < n; ++i) {
        int mmin = 0;
        for (int j = 0; j < n; ++j) {
            if (a[j][i] < a[mmin][i]) {
                mmin = j;
            }
        }
        if (mmin == minRow[mmin]) {
            cout << a[mmin][i] << endl;
        }
    }
```

先行优先遍历，记录每一行最小值的索引，再列优先遍历，找出最小值索引，进行对比，若同时是行最小值和列最小值，即为所求。

8. 编写算法计算一个稀疏矩阵的对角线元素之和，要求稀疏矩阵用三元组顺序表表示。

对三组组进行遍历，若行和列索引相同，这进行加操作，最后返回总和。

```cpp
template<typename T>
struct Triple {
    int r, c;
    T elem;
};

template<typename T>
class SpareMatrix {
    vector<Triple<T>> triList;
    int rows, clos, num;
public:
    SpareMatrix();
    SpareMatrix(Triple<T> * tlist, int rs, int cs, int n) : triList(*tlist),rows(rs), clos(cs), num(n){}
    T diagonalSum() {
        T sum;
        for (auto i = triList.begin(); it < triList.end(); ++i) {
            if (triList[i].r == triList[i].c) {
                sum += triList[i].elem;
            }
        }
        return sum;
    }
};
```