2021-11-7 作业

1. 构建最优二叉树：

代码如下：

```cpp
#include <iostream>

using namespace std;

#define LENGTHP (sizeof(p)/sizeof(p[0]))
#define LENGTHQ (sizeof(q)/sizeof(q[0]))
#define DYNAMICDEBUG(TY, NM, N)                                    \
            printf("Showing type:" #TY ", variable name:" #NM "\n");    \
            showbst<TY>((NM),(N))


template <typename T>
void freebst(T **p, int n) {
    for (int i = 0; i < n; i++)
        delete[] p[i];
    delete[] p;
}

template <typename T>
void showbst(T **p, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            if ((float)p[i][j] != -1.0f)
                printf("%.2f ", (float)p[i][j]);
            else
                printf("%-05s", " ");
        printf("\n");
    }
    printf("\n");
}
```

```cpp
int **optimalbst(float *p, float *q, int lengthp, int lengthq, float ***e, float ***w) {//返回root，记录着对应的索引
    if (lengthp <= 0 || lengthq <= 0) return NULL;
    int **root = new int *[lengthp], i, j;
    float t;
    *e = new float *[lengthq];
    *w = new float *[lengthq];
    for (i = 0; i < lengthp; i++)
        root[i] = new int[lengthp];
    for (i = 0; i < lengthq; i++) {
        (*e)[i] = new float[lengthq];
        (*w)[i] = new float[lengthq];
    }
    //初始化数据
    for (i = 0; i < lengthp; i++)
        for (j = 0; j < lengthp; j++)
            root[i][j] = -1;
    for (i = 0; i < lengthq; i++)
        for (j = 0; j < lengthq; j++)
            (*e)[i][j] = (*w)[i][j] = -1;
    //动态规划算法
    for (i = 0; i < lengthq; i++)
        (*e)[i][i] = (*w)[i][i] = q[i];
    for (int l = 1; l <= lengthp; l++)
        for (i = 1; i <= lengthp - l + 1; i++) {
            j = i + l - 1;
            (*w)[i - 1][j] = (*w)[i - 1][j - 1] + p[j - 1] + q[j];//i下标全部调整，j下标保持
            for (int r = i; r <= j; r++) {
                t = (*e)[i - 1][r - 1] + (*e)[r][j] + (*w)[i - 1][j];
                if ((*e)[i - 1][j] == -1.0f || t < (*e)[i - 1][j]) {
                    (*e)[i - 1][j] = t;
                    root[i - 1][j - 1] = r - 1;
                }
            }
        }
    return root;
}
```

（接下页）

```
void printfbst(int **root, int i, int j, int n) {
    if (i == 0 && j == n - 1)
        printf("k%d是根\n", root[i][j]+1);
    if (i < j)
    {
        int index = root[i][j];
        if (index != i)
            printf("k%d是k%d的左节点\n", root[i][index - 1] + 1, index + 1);
        printfbst(root, i, index - 1, n);
        if (index != j)
            printf("k%d是k%d的右节点\n", root[index + 1][j] + 1, index + 1);
        printfbst(root, index + 1, j, n);
    }
    else if (i == j)
    {
        printf("d%d是k%d的左节点\n", i, i + 1);
        printf("d%d是k%d的右节点\n", i + 1, i + 1);
    }
    else
        printf("d%d是k%d的右节点\n", j + 1, j + 1);
}


int main()
{
    system("chcp 65001"); // 解决window输出中文乱码问题

    float p[] = { 0.15,0.1,0.05,0.1,0.2 }, q[] = { 0.05,0.1,0.05,0.05,0.05,0.1 }, **e, **w;//k1 - k5, d0-d5
    int **root;//root保存的是数组下标
    root = optimalbst(p, q, LENGTHP, LENGTHQ, &e, &w);
    DYNAMICDEBUG(float, e,LENGTHQ);
    DYNAMICDEBUG(float, w, LENGTHQ);
    DYNAMICDEBUG(int, root, LENGTHP);
    printf("解释最优二叉树，数组下标转换为实际的k1-k5, d0-d5\n");
    printfbst(root, 0, LENGTHP-1, LENGTHP);
    freebst<int>(root, LENGTHP);
    freebst<float>(e, LENGTHQ);
    freebst<float>(w, LENGTHQ);



    return 0;
}
```

打印二叉树结果：

```
F:\我的\中大教学\算法设计与分析\21215122_hezhi_20211107>main1.exe
Active code page: 65001
Showing type:float, variable name:e
0.05 0.45 0.90 1.25 1.75 2.75
     0.10 0.40 0.70 1.20 2.00
          0.05 0.25 0.60 1.30
               0.05 0.30 0.90
                    0.05 0.50
                         0.10

Showing type:float, variable name:w
0.05 0.30 0.45 0.55 0.70 1.00
     0.10 0.25 0.35 0.50 0.80
          0.05 0.15 0.30 0.60
               0.05 0.20 0.50
                    0.05 0.35
                         0.10

Showing type:int, variable name:root
0.00 0.00 1.00 1.00 1.00
     1.00 1.00 1.00 3.00
          2.00 3.00 4.00
               3.00 4.00
                    4.00

解释最优二叉树，数组下标转换为实际的k1-k5, d0-d5
k2是根
k1是k2的左节点
d0是k1的左节点
d1是k1的右节点
k5是k2的右节点
k4是k5的左节点
k3是k4的左节点
d2是k3的左节点
d3是k3的右节点
d4是k4的右节点
d5是k5的右节点
```

2. 矩阵连乘问题中，使用动态规划法的状态转移方程为：

$$m[i][j]=\begin{cases}0 & i=j\\ \min\limits_{i\le k<j}\{m[i][j]+m[k+1][j]+w[i][j]\} & i<j\end{cases}$$

其中 $w[i][j]=P_{i-1}P_kP_j$ 表示 $A[i]*A[j]$ 的计算次数

则由 $w[i][j]$ 满足的四边形不等式条件，

只要使 $m[i][j]$ 满足下列条件即可。

$$m[i][j-1]+m[i+1][j]>m[i][j]+m[i+1][j-1]$$