实验2-矩阵连乘问题

问题分析

- 首先对于一整个区间，肯定要去枚举断点，即先处理断点左半部分，再处理断点右半部分，然后将左右两边合并。在枚举的过程中记录下最小值以及断点位置
- 该问题很明显具有最优子结构性质，因为合并两个区间用到的分别是两个区间的最优解
- 该问题同样具有重复计算性质，有一些区间会被重复利用到，因此我们可以采用记忆化搜索的形式
- 定义`dp[i][j]`为对第`i`个矩阵到第`j`个矩阵这一区间合并成一个矩阵所需的最小代价
- 那么最终答案即为`dp[1][n]`
- 接下来考虑状态转移，若区间`[l-r]`从第`k`个矩阵断开，则结果应为

  ```
  chain(l,k)+chain(k+1,r)+p[l-1]*p[k]*p[r];
  ```

  左边矩阵此时是`p[l-1]`行,`p[k]`列;右边矩阵此时是`p[k]`行,`p[r]`列

  合并的代价即为左半边的代价`chain(l,k)`加右半边的代价`chain(k+1,r)`

  加上此次合并的代价`p[l-1]*p[k]*p[r]`

  其中`chain(l,r)`函数返回的就是合并区间`[l-r]`的矩阵所需的最小代价
- 为了输出最终划分方案，我们定义`pos[i][j]`为合并`i~j`这一区间的矩阵选取的断点

  最后即可用分治的策略将划分方案打印出来

chain函数采用记忆化搜索的形式实现，首先判断该区间是否已经被计算过，如果已经被计算过，则直接返回结果；否则采用`for`循环枚举断点，在枚举的过程中记录下最小值以及断点位置，最后返回最小值即可。

```
int chain(int l,int r)
{
    int &v=dp[l][r];
    if(v>0)return v;
    if(l==r)return 0;
    int mn=inf;
    pos[l][r]=l;//pos[l][r]表示矩阵l到r在pos[l][r]后分开
    for(int k=l;k<r;k++)
```

```
    {
        int t=chain(l,k)+chain(k+1,r)+p[l-1]*p[k]*p[r];
        if(t<mn)
        {
            mn=t;
            pos[l][r]=k;
        }
    }
    return v=mn;
}
```

最后打印结果采用分治的策略

当所打印的区间只包含一个值时，即(l==r)直接打印当前矩阵下标，返回即可

然后先打印左括号，再根据断点的位置，去递归处理子区间的划分情况。最后打印右括号

这里需要注意如果当前区间是[1-n]则不需要打印括号

```
void print(int l, int r)
{
    if (l==r)
    {
        cout<<"A"<<l;
        return;
    }
    if (l!=1||r!=n) {
        cout << "(";
    }
    print(l, pos[l][r]);
    print(pos[l][r]+1, r);
    if (l!=1||r!=n) {
        cout << ")";
    }
}
```

运行结果截图

Matrix

```
3
10 100 5 50
Case 1
7500 (A1A2)A3
4
50 10 40 30 5
Case 2
10500 A1(A2(A3A4))
5
50 70 35 5 89 90
Case 3
92300 (A1(A2A3))(A4A5)
2
10 5 30
Case 4
1500 A1A2
6
10 20 30 25 15 50 5
Case 5
13375 A1(A2(A3(A4(A5A6))))
```

设计调试中的问题

- 因本题中有多组测试数据，每轮测试中忘记将dp数组清空导致答案错误，在循环开头加上清空操作即可

  ```
  memset(dp, 0, sizeof dp);
  ```

- 打印括号的时候，最外层的括号也被打印了出来，加上边界判断即可

```
if (l!=1||r!=n) {
  cout << "(";
}
//.....
if (l!=1||r!=n) {
  cout << ")";
}
```

实验体会

本次实验是动态规划中的典型例题，即采用大的区间划分成小的区间进行求解，这一类问题采用记忆化搜索比较简便。本次实验加深了我对动态规划的理解，让我了解到了在动态规划的过程中，怎么去巧妙地定义状态，并且简便地进行状态的转移。这个状态的定义应使得状态总数在一个理想的范围内。且状态转移的过程要做到不重不漏，只有这样才能的到最终的正确结果。我一直认为，动态规划的思想是算法的精髓，它充分体现了算法的美妙之处。在初学动态规划的时候，我觉得它非常困难，比较抽象且难于理解。在不断地学习与探索之后，当我看到一个又一个繁琐的题目被动态规划的解法巧妙解出之后，我被动态规划的伟大所折服，它当之无愧是人类智慧的结晶。