```
/*
给定一个非负整数 numRows，生成杨辉三角的前 numRows 行。
在杨辉三角中，每个数是它左上方和右上方的数的和。

示例:

输入: 5
输出:
[
     [1],
    [1,1],
   [1,2,1],
  [1,3,3,1],
 [1,4,6,4,1]
]

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/pascals-triangle
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 第一类方法:根据杨辉三角的第N行系数与第$N-1$行系数的之间关系进行递推(方法一、二和三);
- 第二类方法:根据二项式展开的公式,进行系数计算.

方法一:C++_Solution(递推方法)

```
class Solution
{
    public:
        vector<vector<int>> generate(int numRows)
        {
            vector<vector<int>>     ret_val         ;
            vector<int>             vi              ;
            int                     i       = 0     ;
            int                     j       = 0     ;
            if(numRows<=0)
            {
                return ret_val;
            }
            else
            {
                for(i = 0; i < numRows ;i++)
                {
                    vi.clear();
                    do
                    {
```

```
                        if(i==0)
                        {
                            vi.push_back(1);
                            break;
                        }
                        if(i==1)
                        {
                            vi.push_back(1);
                            vi.push_back(1);
                            break;
                        }
                        vi.push_back(1);
                        for(j = 0; j <ret_val[i-1].size()-1;j++)
                        {
                            vi.push_back(ret_val[i-1][j]+ret_val[i-1][j+1]);
                        }
                        vi.push_back(1);
                    }while(0);
                    ret_val.push_back(vi);
                }
                return ret_val;
            }
        }
};

/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 C++ 提交中击败了88.48%的用户
内存消耗 :8.6 MB, 在所有 C++ 提交中击败了76.88%的用户
*/
```

方法二:C_Solution(递推方法)

```
int** generate( int     numRows             ,
                int*    returnSize          ,
                int**   returnColumnSizes
              )
{
    int** ret_val   = NULL;
    int*  ret_size  = NULL;
    int   i         = 0;
    int   j         = 0;

    do
    {
        if(numRows<=0)
        {
            *returnSize         =   0       ;
            *returnColumnSizes  =   NULL    ;
            break;
```

```
        }

        ret_val  = (int**) malloc(sizeof(int*)*numRows);
        ret_size = (int*)  malloc(sizeof(int) *numRows);

        for(i = 0;i < numRows ;i++)
        {
            ret_size[i]     = i+1;
            ret_val[i]      = (int*)malloc((i+1)*sizeof(int));
            ret_val[i][0]   = 1;    /*头尾补1*/
            ret_val[i][i]   = 1;
            if(i>1)
            {
                for(j=0;j<i-1;j++)
                {
                    ret_val[i][j+1] = ret_val[i-1][j] + ret_val[i-1][j+1];
                }
            }
        }
        *returnSize         = numRows   ;
        *returnColumnSizes  = ret_size  ;
    }while(0);
    return ret_val;
}

/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 C 提交中击败了88.46%的用户
内存消耗 :7.3 MB, 在所有 C 提交中击败了5.69%的用户
*/
```

方法一:C++_Solution(递推方法,优化细节)

```
class Solution
{
    public:
        vector<vector<int>> generate(int numRows)
        {
            vector<vector<int>>     ret_val         ;
            vector<int>             vi              ;
            int                     i       = 0     ;
            int                     j       = 0     ;
            if(numRows<=0)
            {
                return ret_val;
            }
            else
            {
                for(i = 0; i < numRows ; i++ )
                {
```

```
                    vi.clear();
                    do
                    {
                        vi.push_back(1);    /*头*/
                        if(i==0)
                        {
                            break;
                        }
                        for(j = 0; j <i-1;j++)
                        {
                            vi.push_back(ret_val[i-1][j]+ret_val[i-1][j+1]);
                        }
                        vi.push_back(1);    /*尾*/
                    }while(0);
                    ret_val.push_back(vi);
                }
                return ret_val;
            }
        }
};
/*
执行结果：
通过
显示详情
执行用时 :0 ms, 在所有 C++ 提交中击败了100.00%的用户
内存消耗 :8.7 MB, 在所有 C++ 提交中击败了40.12%的用户
*/
```

AlimyBreak
2019.08.01