正整数n的所有可能和式的组合

很多人在面试中会被问到这样的题目，题目的含义是有如下的组合4=1+1+1+1、1+1+2、1+3、2+1+1、2+2。光从题目来看有两种理解：

将3 = 1 +2 和3 = 2 +1当作不同的组合。这种情况是比较简单的，直接将给定的n递归地分解成(n - 1) + 1当递归求得的结果和我们需要分解的整数n相等，则这次分解就完成了，我们可以把分解的组合输出来，然后返回。一直递归到n不能再分解（也就是分解成了n个1）。

```
#include <iostream>
#include <vector>

// 过往记忆
// www.iteblog.com
// 转载请注明
using namespace std;

void add(int sum, int start, int tempSum, vector<int> &v){
  if(sum == tempSum){
    vector<int>::iterator it = v.begin();
    for(; it != v.end(); it++){
      cout << *it;
      if(it + 1 != v.end()){
        cout << " + ";
      }
    }

    cout << endl;
    return;
  }else if(sum < tempSum){
    return;
  }

  for(int i = sum; i > 0; i--){
    v.push_back(i);
    add(sum, i, tempSum + i, v);
    v.pop_back();
  }
}

void add(int sum){
  int tempSum = 0;
  vector<int> v;
```

```
  add(sum, sum, tempSum, v);
}

int main(){
  add(5);
  return 0;
}
```

运行的结果：

```
5
4 + 1
3 + 2
3 + 1 + 1
2 + 3
2 + 2 + 1
2 + 1 + 2
2 + 1 + 1 + 1
1 + 4
1 + 3 + 1
1 + 2 + 2
1 + 2 + 1 + 1
1 + 1 + 3
1 + 1 + 2 + 1
1 + 1 + 1 + 2
1 + 1 + 1 + 1 + 1
```

如果把将3 = 1 +2 和3 = 2 +1当作相同的组合，这下相对来说比较难点，但是仔细分析，会发现，算法和上面的几乎一样，每次分解的数不再是从n开始，这样可以使得每次分解的数不会超过上一次分解出来的数，代码只修改了一个地方就可以实现：

```
#include <iostream>
#include <vector>

// 过往记忆
// www.iteblog.com
// 转载请注明
using namespace std;
```

```
void add(int sum, int start, int tempSum, vector<int> &v){
  if(sum == tempSum){
    vector<int>::iterator it = v.begin();
    for(; it != v.end(); it++){
      cout << *it;
      if(it + 1 != v.end()){
        cout << " + ";
      }
    }

    cout << endl;
    return;
  }else if(sum < tempSum){
    return;
  }

  for(int i = start; i > 0; i--){//和上面的程序相比，只修改了这个地方
    v.push_back(i);
    add(sum, i, tempSum + i, v);
    v.pop_back();
  }
}

void add(int sum){
  int tempSum = 0;
  vector<int> v;
  add(sum, sum, tempSum, v);
}

int main(){
  add(5);
  return 0;
}
```

程序运行结果：

```
5
4 + 1
3 + 2
3 + 1 + 1
2 + 2 + 1
2 + 1 + 1 + 1
1 + 1 + 1 + 1 + 1
```