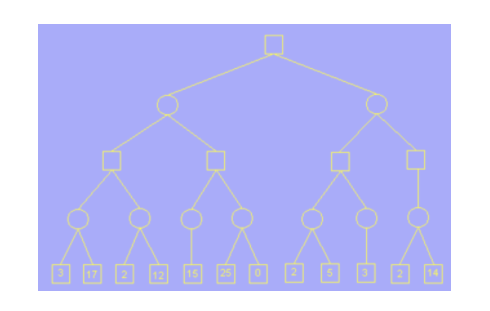
**剪枝算法(算法优化)**

剪枝算法：在搜索算法中优化中，剪枝，就是通过某种判断，避免一些不必要的遍历过程，形象的说，就是剪去了搜索树中的某些“枝条”，故称剪枝。应用剪枝优化的核心问题是设计剪枝判断方法，即确定哪些枝条应当舍弃，哪些枝条应当保留的方法。

剪枝优化三原则: 正确、准确、高效

所谓剪枝算法：

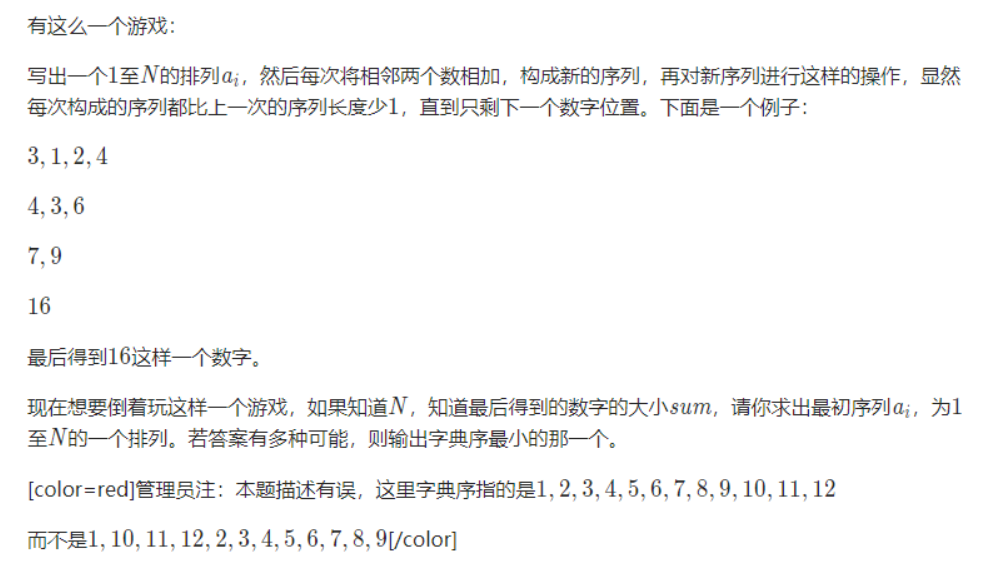


上图为一搜索树，正方形为自己（A），圆为对手（B），初始设置α为负无穷大，β为正无穷大

假设α为下界，β为上界，对于α ≤ N ≤ β

若 α ≤ β 则N有解

若 α > β 则N无解



程序代码：

#include<iostream>

#include<algorithm>

#define maxsize 15

using namespace std;

int n, target;

int used[maxsize];

int ans[maxsize];

int res[maxsize];

int table[maxsize][maxsize];

bool flag = false;

//建立规律公式表，将计算复杂度从n^2降到n

void buildtable(int n) {

table[1][1] = 1;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= i; j++) {

if (j < i) {

table[i][j] = table[i - 1][j] + table[i - 1][j - 1];

}

else if (i == j) {

table[i][j] = 1;

}

}

}

}

bool ansIsOk(int n,int targetval) {

//计算ans中的n个数的当前组合是否满足targetval

//ans[1,n]的计算，一共计算n次

int tmp[maxsize];

for (int i = 1; i <= n; i++) {

tmp[i] = ans[i];;//拷贝数组

}

for (int i = n-1; i >=1 ; i--) {

for (int j = 1; j <= i; j++) {

tmp[j] = tmp[j] + tmp[j + 1];

}

}

return tmp[1] == targetval;//判断是否等于具体值

}

//本题提升的关键是如何使得ansIsOk()这个函数得到简化

bool ansIsOk2(int currow, int targetval) {

int res = 0;

for (int i = 1; i <= currow; i++) {

res += table[currow][i] \* ans[i];

}

return res == targetval;

}

bool PreJudgeNotOk(int currow) {

if (currow < n + 1) {

int res = 0;

for (int i = 1; i < currow; i++) {

res += table[n][i] \* ans[i];

}

if (res >= target) {

return true;

}

}

return false;

}

void dfs(int row) {

//这里的框架仍然是先枚举，再判断。这里的枚举是先把数列排列出来，再另外去计算这个数列是否满足要求(作为返回要求）

//row指准备填充row个数，已经填充row-1个数

if (row < n + 1) {

if (PreJudgeNotOk(row)) {

return;

}

}

if (row == n + 1) {

if (ansIsOk2(n,target)) {

flag = true;

for (int i = 1; i <= n; i++) {

res[i] = ans[i];

}

return;

}

}

for (int i = 1; i <= n; i++) {//此处为枚举可能性

if (flag == true) {

break;//如果已经找到最小的排列，直接跳出

}

if (used[i] != 1) {

used[i] = 1;

ans[row] = i;//此处为将可能性带入当前的结点，即row结点

dfs(row + 1);

used[i] = 0;

}

}

}

int main() {

cin >> n >> target;

//从下往上搜索

buildtable(n);

dfs(1);//从第1个数排列到最后一个数

if (flag == true) {

for (int i = 1; i <= n; i++) {

cout << res[i] << " ";

}

}

else if (flag == false) {

}

return 0;

}

运行结果：

