# 题目

整数可以被看作是其因子的乘积。

例如：

8 = 2 x 2 x 2;

= 2 x 4.

请实现一个函数，该函数接收一个整数 n 并返回该整数所有的因子组合。

注意：

你可以假定 n 为永远为正数。

因子必须大于 1 并且小于 n。

示例 1：

输入: 1

输出: []

示例 2：

输入: 37

输出: []

示例 3：

输入: 12

输出:

[

[2, 6],

[2, 2, 3],

[3, 4]

]

示例 4:

输入: 32

输出:

[

[2, 16],

[2, 2, 8],

[2, 2, 2, 4],

[2, 2, 2, 2, 2],

[2, 4, 4],

[4, 8]

]

通过次数4,822提交次数8,474

请问您在哪类招聘中遇到此题？

# 分析

## 方法一：递归

class Solution {

public:

// l 代表因子的起始数，保证因子的有序性可以做到天然的去重

vector<vector<int> > dfs(int n, int l) {

vector<vector<int> > res;

for (int i = l; i \* i <= n; ++i) {

if (n % i == 0) {

res.push\_back({n / i, i});

for (auto v : dfs(n / i, i)) {

v.push\_back(i);

res.push\_back(v);

}

}

}

return res;

}

vector<vector<int>> getFactors(int n) {

return dfs(n, 2);

}

};

另一种思路：

思路和心得：

1、递归，DFS

2、就是一层一层剥开的过程

3、注意边界

代码：

class Solution

{

public:

vector<vector<int>> dfs(int n, int L)

{

vector<vector<int>> res;

int i = L;

while (i \* i <= n)

{

if (n % i == 0)

{

res.push\_back({i, n/i}); //i=2, n=100, n/i=50

for (vector<int> v: dfs(n/i, i)) //dfs(50, 2)

{

v.push\_back(i); //n/i \* i = n

res.push\_back(v);

}

}

i ++;

}

return res;

}

vector<vector<int>> getFactors(int n)

{

return dfs(n, 2);

}

};

## 方法二：回溯

思路：比如说24，主函数把他拆成了两个数的乘积2x12、3x8、4x6（只需要判断到n的平方根即可），递归函数中的核心是将当前数组输出到结果中，并将最后一个数进行拆分，注意最后一个数拆分出的因子必须必前一个数的要小，比如4x16只能拆成4x4x4而不能在拆出比4小的数，这样输出的每一个结果都是非递减数组，并进行了剪枝。

class Solution {

vector<vector<int>> res;

void dfs(vector<int> m)

{

res.push\_back(m);

for(int i=m[m.size()-2];i<=sqrt(m[m.size()-1]);++i)

{

if(m[m.size()-1]%i==0)

{

vector<int> temp(m.begin(),m.end()-1);

temp.push\_back(i);

temp.push\_back(m[m.size()-1]/i);

dfs(temp);

}

}

}

public:

vector<vector<int>> getFactors(int n) {

for(int i=2;i<=sqrt(n);++i)

{

if(n%i==0)

{

vector<int> temp({i,n/i});

dfs(temp);

}

}

return res;

}

};