DFS递归剪枝

例题：1103 Integer Factorization (30分)

The *K*−*P* factorization of a positive integer *N* is to write *N* as the sum of the *P*-th power of *K* positive integers. You are supposed to write a program to find the *K*−*P* factorization of *N* for any positive integers *N*, *K* and *P*.

Input Specification:

Each input file contains one test case which gives in a line the three positive integers *N* (≤400), *K* (≤*N*) and *P* (1<*P*≤7). The numbers in a line are separated by a space.

Output Specification:

For each case, if the solution exists, output in the format:

```
N = n[1]^P + ... n[K]^P
```

where `n[i]` (`i` = 1, ..., `K`) is the `i`-th factor. All the factors must be printed in non-increasing order.

Note: the solution may not be unique. For example, the 5-2 factorization of 169 has 9 solutions, such as 122+42+22+22+12, or 112+62+22+22+22, or more. You must output the one with the maximum sum of the factors. If there is a tie, the largest factor sequence must be chosen -- sequence { $a1,a2,\cdots,a**K$ } is said to be **larger** than { $b1,b2,\cdots,b**K$ } if there exists $1\le L\le K$ such that $a**i=b**i$ for $i<L$ and $a**L>b**L$.

If there is no solution, simple output `Impossible`.

Sample Input 1:

```
169 5 2
```

Sample Output 1:

```
169 = 6^2 + 6^2 + 6^2 + 6^2 + 5^2
```

Sample Input 2:

```
169 167 3
```

Sample Output 2:

```
Impossible
```

思路

1. 剪枝策略是：当

$$time == k \quad but \quad now! = n$$

   需要剪枝

2. 当需要non-increasing序列时，并且题目要求当条件满足之后要求输出字典序。直接从最后一个元素开始运算，可以免去排序，直接得到结果。
3. 因为从最高位开始，循环起始值没必要从0开始，可以从上一层的index作为起始循环点
4. 说实话好像没有啥固定的剪枝套路，自己看着办吧

代码实现

```
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <vector>
#include <cmath>
using namespace std;
int n, k, p, macSumFac = -1, length;
vector<int> result, path, value;
void init()
{
    cin >> n >> k >> p;
    for(int i=0;i<=n;i++)
    {
        int key = pow(i, p);
        if (key > n)
            break;
        else
            value.push_back(key);
    }
    length = value.size();
}
bool cmp(int a,int b)
{
    return  a > b;
}
void dfs(int index,int time,int now,int sumFac)
{
    if(time==k)
    {
        if(sumFac>macSumFac&&now==n)
        {
            macSumFac = sumFac;
            result = path;
        }
        return;
    }
    while(index>=1)
    {
        if (now + value[index] <= n)
        {
            path.push_back(index);
            dfs(index, time + 1, now + value[index], sumFac + index);
            path.pop_back();
        }
        if(index==1)
```

```
            return;
        index--;
    }
}
int main()
{
    init();
    dfs(value.size() - 1, 0, 0, 0);
    if (macSumFac == -1)
        cout << "Impossible";
    else
    {
        cout << n << " = " << result[0] << "^" << p;
        for(int i=1;i<result.size();i++)
        {
            cout << " + " << result[i] << "^" << p;
        }
    }
    return 0;
}
```