实验舱
青少年编程
走近科学 走进名校

蛟龙四班

深度优先搜索

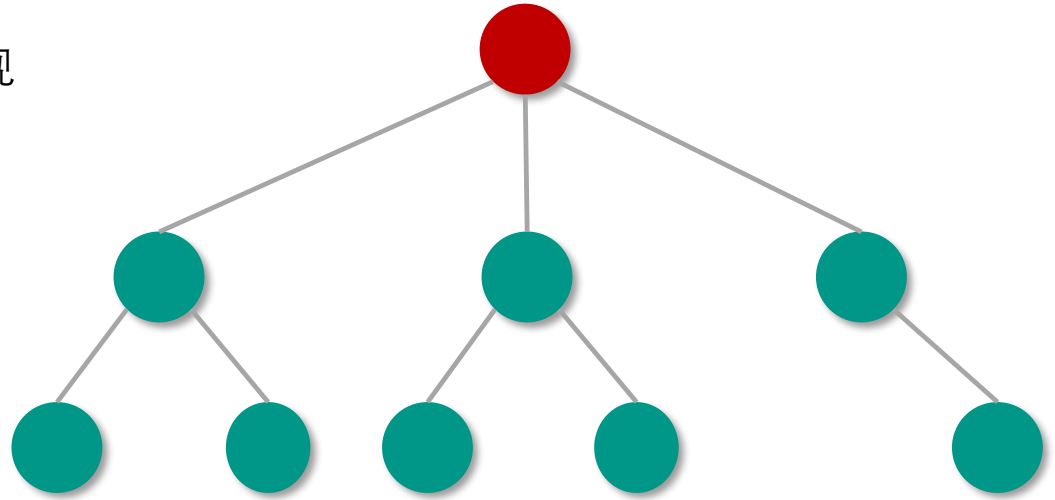
Mas

深度优先搜索

搜索是一种枚举的方式,把所有可行的解进行尝试

深度优先搜索($depth-first-search$)按照深度优先的方式进行搜索

深度优先搜索往往使用递归实现,也可以使用栈模拟递归实现



#1105 组合输出

题目描述

排列与组合是常用的数学方法，其中组合就是从 n 个元素中抽出 r 个元素(不分顺序且 $r \leq n$)，我们可以简单地将 n 个元素理解为自然数 $1，2，\ldots，n$，从中任取 r 个数。
现要求你用输出所有组合。

输入

一行两个自然数 $n、r(1 < n < 21，1 \leq r \leq n)$。

输出

所有的组合，每一个组合占一行且其中的元素按由小到大的顺序排列，所有的组合也按字典顺序

> *保证方案数不超过* 10^4

输入样例

```
5 3
```

输出样例

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

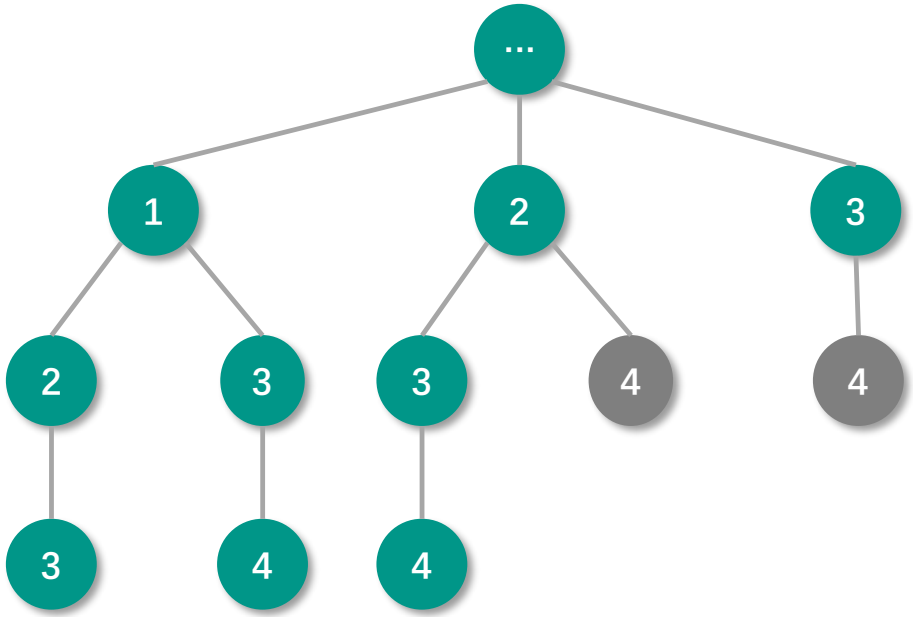
#1105 组合输出

若r确定为3,不难写出三层的循环枚举的代码

当r不是常量时,可以使用递归实现r层的枚举

每一层枚举只要保证序列递增,就不会出现重复

为什么这么写能保证字典序?




```cpp
#include <bits/stdc++.h>
using namespace std;
int n, r, ans[21];
void dfs(int pre, int deep)
{
  if (deep == r + 1) //第 r + 1 层(边界)
  {
    for (int i = 1; i <= r; i++)
      printf("%d%c", ans[i], " \n"[i == r]);
    return;
  }
  for (int i = pre; i <= n; i++) //每一层的枚举
  {
    ans[deep] = i;          //记录答案
    dfs(i + 1, deep + 1); //进行下一层的枚举
  }
}
int main()
{
  scanf("%d%d", &n, &r);
  dfs(0, 1);
  return 0;
}
```

#1065 自然数的拆分

【题目描述】

任何一个大于 1 的自然数 n，总可以拆分成若干个小于 n 的自然数之和。

当 $n = 7$ 共 14 种拆分方法：

```
7=1+1+1+1+1+1+1
7=1+1+1+1+1+2
7=1+1+1+1+3
7=1+1+1+2+2
7=1+1+1+4
7=1+1+2+3
7=1+1+5
7=1+2+2+2
7=1+2+4
7=1+3+3
7=1+6
7=2+2+3
7=2+5
7=3+4
total=14
```

【输入】

输入 n。

【输出】

按字典序输出具体的方案。

【输入样例】

```
7
```

【输出样例】

```
1+1+1+1+1+1+1
1+1+1+1+1+2
1+1+1+1+3
1+1+1+2+2
1+1+1+4
1+1+2+3
1+1+5
1+2+2+2
1+2+4
1+3+3
1+6
2+2+3
2+5
3+4
```

#1065 自然数的拆分

当数不可再拆分时,结束搜索

保证拆分后的方案是单调非降的,可以避免重复

对于每一层枚举满足的x, 只剩下 $n - x$

如何避免拆分成 n 自身的情况?

```
void dfs(int pre, int num, int deep)
{
  if (num == 0)
  {
    for (int i = 1; i < deep; i++)
      printf("%d%c", ans[i], "+\n"[i == deep - 1]);
    return;
  }
  for (int i = pre; i <= num; i++)
  {
    ans[deep] = i;
    dfs(i, num - i, deep + 1);
  }
}
```

#2198、素数分解

题目描述

素数，又称质数，是指除 1 和其自身之外，没有其他约数的正整数。
例如 2、3、5、13 都是素数，而 4、9、12、18 则不是。
虽然素数不能分解成除 1 和其自身之外整数的乘积，但却可以分解成更多素数的和。
你需要求出一个正整数最多能分解成多少个互不相同的素数的和。

例如，$21 = 2 + 19$ 是 21 的合法分解方法。

$21 = 2 + 3 + 5 + 11$ 则是分解为最多素数的方法。
再比如：128，最多可以分解为 9 个素数的和。

输入格式

第一行是输入一个正整数 n

输出格式

输出一个正整数,表示最多能分解成多少个不同的素数的和。

样例输入1

```
21
```

样例输出1

```
4
```

样例输入2

```
128
```

样例输出2

```
9
```

数据规模

对于全部的数据 $2 \leq n \leq 200$

#2327、超级素数

题目描述

在大于 1 的自然数中，除了 1 和它本身以外不再有其他因数的数，被称为素数，又叫质数。

超级素数是指一个素数，每去掉最后面的一个数字，总能保证剩下的数依然为素数。比如 373 就是一个超级素数，去掉个位的 3 后，37 依然是素数；继续去掉 37 个位的 7 后，3 还是素数。

输入格式

输入一个整数 n 。

输出格式

第一行输出一个数，表示所有小于等于 n 的超级质因数个数。

接下来若干行,每行一个满足条件的超级素数,升序输出。

数据规模

对于 30% 的数据 $2 \leq n \leq 10^5$

对于 100% 的数据 $2 \leq n \leq 10^8$

样例输出1

```
6
2
3
5
7
23
29
```

样例解释1

有 $2\ 3\ 5\ 7\ 23\ 29$ 共 6 个数满足条件。

#2327、超级素数

对于当前数x,可以考虑将$i \in [1,9]$ 拼接到其后

对于新数$x \times 10 + i$如果为质数,那么继续往下搜索

如果x已经超过n,那么不再继续向下搜索

将搜索结果排序去重后输出

```
void dfs(int num)
{
  for (long long i = 1; i <= 9; i++)
    if (num * 10 + i <= n && isPrime(num * 10 + i))
    {
      dfs(num * 10 + i);
      ans[++cnt] = num * 10 + i;
    }
}
```

#1509、选球游戏

题目描述

有 K 种颜色的小球 $(K \leq 10)$，每种小球有若干个，每种小球的数量小于 10 个。
现在要从中挑出 $N(1 \leq n \leq 20)$ 个小球,请你计算一下有多少种挑选方式。

> *注：每种颜色的小球之间没有差别。*

请按数字递增顺序输出挑选小球的所有方式。

如有 3 种颜色，每种颜色小球的个数分别为 $a:1, b:2, c:3$，挑出 3 个小球的挑法有：
$003, 012, 021, 102, 111, 120$

输入描述

第一行两个数字 K N，分别表示小球种类数目和挑选的小球个数
第二行开始为每种小球的数目，共 K 行数据

输出描述

输出所有可行的挑选方案，按升序排列

输入样例

```
3 3
1
2
3
```

输出样例

```
003
012
021
102
111
120
```

做k层的枚举

每层枚举同一小球的数量

到达边界后检查小球数量之和是否为N

```cpp
int a[101], ans[101], k, n;
void dfs(int cur, int sum)
{
  if (cur == k + 1)
  {
    if (sum == n)
      for (int i = 1; i <= k; i++)
        cout << ans[i] << " \n"[i == k];
    return;
  }
  for (int i = 0; i <= a[cur]; i++)
  {
    ans[cur] = i;
    if (sum + i <= n)
      dfs(cur + 1, sum + i);
  }
}
```

#1156、和为T

问题描述

从一个大小为 n 的整数集中选取一些元素，使得它们的和等于给定的值 T 。
每个元素限选一次，不能一个都不选。

输入格式

第一行一个正整数 n ，表示整数集内元素的个数。
第二行 n 个整数，用空格隔开。
第三行一个整数 T ，表示要达到的和。

输出格式

输出有若干行，每行输出一组解，即所选取的数字，按照输入中的顺序排列。

若有多组解，优先输出不包含第 n 个整数的；

若都包含或都不包含，优先输出不包含第 $n-1$ 个整数的，依次类推。

最后一行输出总方案数。

> *观察输出样例,应该优先选择靠后的元素*

样例输入

```
5
-7 -3 -2 5 9
0
```

样例输出

```
-3 -2 5
-7 -2 9
2
```

数据规模和约定

对于全部数据 $1 \leq n \leq 22$， $-2^{63} \leq T < 2^{63}$

#1156、和为T

对于一个集合,尝试枚举其子集

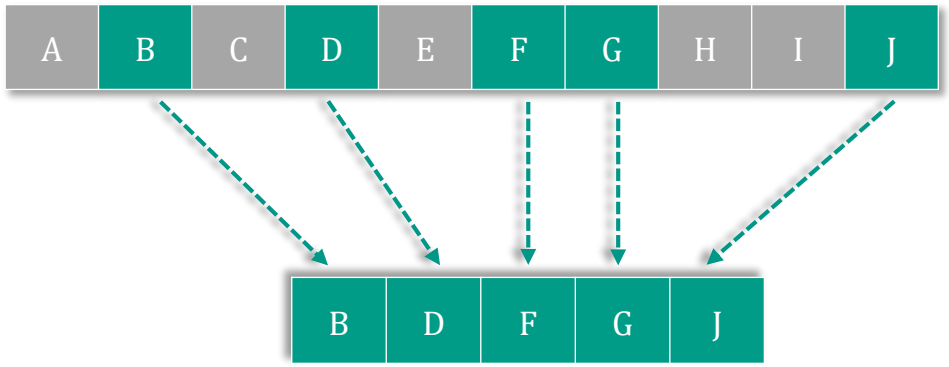
对于本题从后往前考虑,

对于第i个元素有取或不取两种选择(使用全局的$bool$数组标记)

若考虑完第i个仅需要考虑$1 \sim i-1$范围内的元素

当所有元素都考虑完毕后,根据$bool$数组中的取值还原出子集

```
bool vis[25];
void dfs(int idx)
{
  if (idx == 0)
  {
    //还原子集并判断输出
    return;
  }
  vis[idx] = false;
  dfs(idx - 1);
  vis[idx] = true;
  dfs(idx - 1);
}
```



#1156、和为T

对于整形变量可使用按位与、按位或、按位异或 等位运算符,将整数作为二进制,逐一对每一个二进制位按如下规则计算

与：两者都为 1 时,结果为 1,否则为 0

或：两者都为 0 时,结果为 0,否则为 1

异或：两者同为 0 或 1 时,结果为 0,否则为 1

A	B	A&B	A|B	A^B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

左移<<、右移>>

A << B,表示把A转化为二进制后向左移动B位(在末尾添加B个0)

A >> B,表示把A转化为二进制后向右移动B位(删除末尾的B位)

#1156、和为T

每一个元素仅有取(1)和不取(0)两种情况,对于子集的选取状态是一个01序列

可用整数的二进制数位表示集合对应某一元素的选取状态

例如对于集合 $S = \{0,1,2,3,4,5,6\}$

那么二进制$(45)_{10} = (101101)_2$ 就代表子集合$\{0,2,3,5\}$

考虑在十进制下枚举n的元素的所有选取状态$(0 \sim 2^n - 1)$

将该状态的每一个进制位进行拆分即可还原出子集合

```
for (int i = 1; i < (1 << n); i++)
{
  long long sum = 0;
  for (int j = 0; j < n; j++)
    if ((i >> j) & 1)
      sum += a[j];
  // 判断并还原子集合输出
}
```

#1791、能力宝石

题目描述

Mas 有 n 块能力宝石石，每块宝石会给携带者带来一定的能力值,同时携带者也必须承受宝石带来的反噬效果。

Mas 佩戴能力宝石获得的总能力值为所有佩戴宝石的能力值之和，受到的总反噬值为所有宝石反噬值之积。

现在 Mas 想知道，如何选取能力宝石（不能一块都不选），可以使得宝石对自身的伤害最小(总能力值和总反噬值相差的值)。现在请你帮帮他，计算出这个相差最小的值。

输入格式

第一行一个整数 $n(1 \leq n \leq 24)$ 。

接下来 n 行，每行两个空格分隔的整数，分表表示一块能力宝石的反噬值和能力值。

数据保证，所有能力值之和和所有反噬值之积都在 int 范围内。

输出格式

一个整数，表示最小差值。

样例输入

```
4
1 7
2 6
3 8
4 9
```

样例输出

```
1
```

样例解释

选取如下宝石

```
2 6
3 8
4 9
```

到反噬值为 $2 \times 3 \times 4 = 24$,获得能力值为 $6 + 8 + 9 = 23$

#1791、能力宝石

枚举出所有的子集

反噬值累乘,能力值累加

```
for (int i = 1; i < (1 << n); i++)
{
  int a = 1, b = 0;
  for (int j = 0; j < n; j++)
    if ((i >> j) & 1)
      a *= rep[j].a, b += rep[j].b;
  ans = min(ans, abs(a - b));
}
```

```
void dfs(int idx)
{
  if (idx == n + 1)
  {
    int a = 1, b = 0;
    for (int i = 1; i <= n; i++)
      if (vis[i])
        a *= rep[i].a, b += rep[i].b;
    return;
  }
  vis[idx] = true;
  dfs(idx + 1);
  vis[idx] = false;
  dfs(idx + 1);
}
```

谢谢观看