## 框架：

回溯算法对解空间作深度优先搜索，根据图的深度优先算法我们把框架也构造成两层的：

层次一：驱动调用层次二

Function(参数列表){

…….. //相关操作

Backtrack(0);

……..

}

层次二：进行实际的搜索操作

void BackTrack(int t)

{

if (t > n)

Output(x);

else{

for (int i = f(n, t); i < g(n, t); i++){

x[t] = h(i);

if (Constraint(t) && Bound(t))

BackTrack(t + 1);

}

}

}

## 2.对框架的解释

1.t表示递归深度

2.x[]用来记录可行解

3.f(n, t)和 g(n, t)分别表示当前开展结点处未搜索过的子树（共有i个）的起始编号和终止编号

4.h（i）表示在当前扩展结点处，x[t]的第i可选值

## 3.对回溯算法的总结

1.回溯算法把问题的解空间组织成树后，问题的解就变成了一条从树根到树叶的路径。

2.回溯算法是如何把问题的解空间组织成树的？

首先必须要明白一点：回溯算法并没有在一开始就把问题的解空间的树就构造好了，而是在搜索的过程中动态地构造，具体的实现细节是：按照扩展结点的原则f(n, t)和g(n, t)扩展结点，从当前结点到扩展结点的边的取值可以从h(i)获得。f(n, t)和g(n, t)和h(i)根据所求问题可以求得。上述算法是递归算法，程序执行过程中会对上述操作重复得进行，当程序执行完成，也就是搜索完成，则在此过程中完成了对完整的解空间的树的构造。

（在子集树中，h(i)的取值只有0和1，在一些问题中，边的取值是很容易获得的，比如图的着色问题，h(i)的取值就是颜色的取值）

3.回溯算法是如何构造这样的一条路径的？

我们已经知道，回溯算法是对解空间树进行深度优先搜索，深度优先搜索进行搜索时如果成功则会找到一条从根结点到叶结点的路（路是有一条一条互相连接的边组合而成的，边的形成在2中已经介绍），x[t] = h(i)用来记录每条边的取值。获得了路径也就获得了问题的解。

4.递归回溯算法是如何进行回溯的？

递归机制实现的

5．迭代回溯算法是如何进行回溯的？