第五章 回溯法

5.1 回溯算法基本思想

- 回溯法是一种通用的解决问题的办法，本质上就是一种穷举，并且是一种避免重复的穷举。
- 所有回溯算法与走迷宫具有相同的本质。

迷宫问题

开始　第一次回溯　第二次回溯　第三次回溯　其他回溯

结束

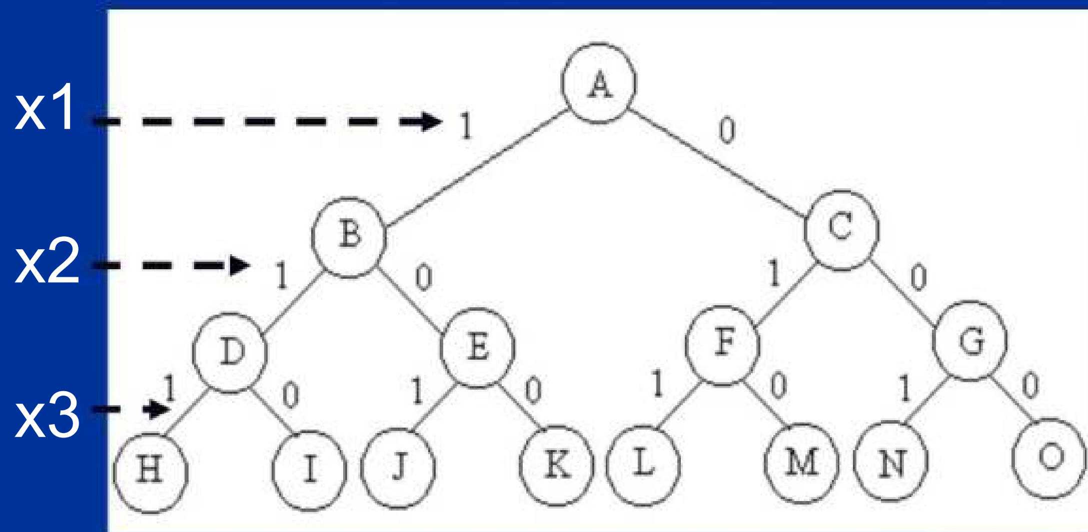
回溯：往回退，追根溯源。

求解的目标:确定每一个十字路口的**选择**，最后走出迷宫。

问题的解空间树

湖北经济学院计科系

- 问题的解向量：回溯算法希望一个问题的解能够表示成一个n元式(x1,x2,…,xn)的形式。
- 解空间树：由解元素x1,x2…xn取不同值所构成组合(或排列)序列，迷宫中xi表示第i个十字路口做出的选择。



Eg. n=3时的0-1背包问题解空间树为左图所示二叉树

显式约束：对分量xi的取值限定。

隐式约束：为满足问题的解对不同分量之间施加的约束。

湖北经济学院计科系

- 回溯算法的本质：确定问题的解空间树后，深度优先遍历此空间树。但是往往问题的解空间树是无法完全存储的（eg），所以遍历有它的特点。
- 通常情况下的处理策略是：一边生成解空间树，一边遍历。

解空间树的生成方法

- 深度优先的解空间树生成法：从根结点R开始（R是第一个扩展结点），生成它的一个儿子结点C(即确定了x1)；然后以C作为新的扩展结点，在完成了以C为根的子树的穷尽搜索之后(即递归，是在x1保持不变的情况下，分别求解x2…xn的所有可能组合)，将R重新变成扩展结点（即回溯到R结点处），继续生成R的下一个儿子（即让x1取另外一个值，直至没有值可取即会回退到R的父结点，此时以R为根的树遍历结束）。
- 扩展结点：一个准备生成其儿子的结点

湖北经济学院计科系

湖北经济学院计科系

- 回溯算法正是按深度优先解空间树生成法，结点生成到哪里，就遍历到哪里。
- 此过程可以表达成如下递归形式和非递归形式。
- 递归形式的表达：

```
BackTrack(k)  //当前已经具备k-1个解元素，准备求Xk
{  if(x1,x2,…,xk-1) 是问题的解   输出；
   else  {
          计算第k个解元素的候选集合Sk;
            //Sk相当于第k个十字路口的选择
          while(Sk != 空)
            { xk=Sk中的一个元素；//Xk已求出
              Sk=Sk-{xk};
              BackTrack(k+1);  //递归进一步求Xk+1 }
          }//while结束，即表示第k个路口已穷举完
}
```

湖北经济学院计科系

- 非递归形式的表达：

```
BackTrack(X)
{ 计算解向量X的第一个元素x1的候选集合S1；
  k=1;          //S1即第一个十字路口处没有走过的支路
 While(k>0)
  {  while  Sk != 空
       {  Xk=Sk中的一个元素；       //第k个路口已确定
          Sk=Sk-{Xk};          //已走过的支路不会重复走
          if(X=(x1,x2,…xk)是问题的解) 输出；
          k++; 计算候选集合Sk; }
     k--;                  //回溯，即回退到第k-1个十字路口
  }
}
```

湖北经济学院计科系

- 例如：输出1~n的全排列，递归表达如下（pailie0.c）

```
void f(int k) //已经具备X1~Xk-1,函数目的：求解X[k]
{int i;
  if(k-1==n)  //找到问题的一个解
  {   cout<<endl<<"The "<<++count<<":";
      for(i=1;i<n+1;i++)  cout<<X[i]<<“  ”; } //输出解向量

  else  for(i=1;i<n+1;i++) //候选集为1~n中没有用过的数
    if(!used[i])   //used[i]=0表示i没有使用过，否则已使用
    { X[k]=i;
       used[i]=1;  //i已经使用过
      f(k+1);       //递归求Xk+1
      uese[i]=0;  //f(k+1)结束表示解空间树中第k层子树遍历完
   }//即X1~Xk确定为某种组合的所有可能排列结果都已输
     //出，例如{1,2,3,4},{1,2,4,3}即为x1,x2确定为1,2的所有可能
} //排列结果，所以f(k+1)结束后，会在X1~Xk-1不变的情况
   //下，看看除了i以外Xk还有没有其它可用的值
```

此处有多种写法，写成限界函数(xianjie(k,i),判断Xk能否取i)最好

非递归的算法过程如下

湖北经济学院计科系

```
void f1( void )
{ k=1;from=1; l=1;
  while(k>0)
  { for( ;i<n+1;i++) if(!used[i] && k<=n)
          {X[k]=l;used[i]=1;if(k==n)输出;k++;}


    k--; i=X[k]+1;   //回溯的处理，让第X[k]的取
   值范围变为l~n，实际上就是X[k]+1~n
  }
}
```

湖北经济学院计科系

- 采用这种思路框架来设计程序的好处：

  1）保证回溯思路的正确性，并且减轻回溯过程的思考工作。一律看成是求解向量X的过程。

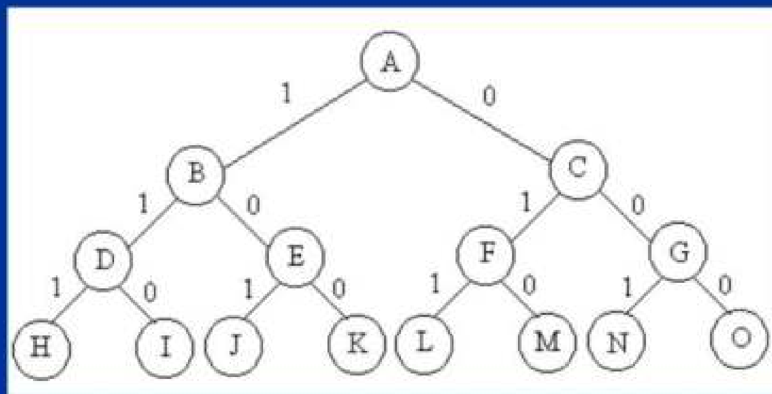
  2）把限界函数分离出来，便于程序的编写和调试，每一个问题的约束条件都不一样，但是他们回溯的思路都是一样的。所以这样处理类似于模块化 。

关键点：限界函数的设计，限界函数完全取决于问题的显式约束和隐式约束。

5.2 解空间树的两种形态

子集树：即将一批元素进行组合（与划分成若干个集合等价）的问题，例如0-1背包、礼物分配、集装箱装载、图的着色等问题

湖北经济学院计科系




```
void backtrack (int t)
{
  if (t>n) output(x);
  else      for (int
   i=0;i<=1;i++)
         if (xianjie(t, i))
        {       x[t]=i;
                 backtrack(t+1);
           }
}
```

xianjie函数**检查Xt**能否取**i**

时间复杂度：$\Theta\ (a^n)$
a为被划分集合个数

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/745244123101011321