分支限界法感想

分支限界法，也有一个通俗的名字，广度优先搜索。也是本书中相对简单的一种算法。分支是使用广度优先策略，依次生成扩展结点的所有分支。限界是在结点扩展过程中，计算结点的上界，搜索的同时剪掉某些分支。分支限界法就是把问题的可行解展开，再由各个分支寻找最佳解。与回溯法类似，分支限界法也是在解空间中搜索得到解；不同的是，分支限界法会生成所有扩展结点，并舍弃不可能通向最优解的结点，然后根据广度优先/最小耗费优先，从活结点中选择一个作为扩展结点，使搜索向解空间上有最优解的分支推进。

需要注意的是，限界函数很大程度上决定了算法的效率。同一问题可以设计不同的限界函数。队列式分支限界法中，常以约束条件作为限界函数，满足约束条件才可入队，不满足约束条件的舍弃。优先队列式分支限界法中，还可以设计一个启发函数作为限界函数。
对于有约束的问题，队列法和优先队列法均可以求解；对于无约束问题， 宜使用优先队列法。

分支限界法也有一个固定的解题套路。如下面的伪代码所示。

```
while(队列不空){
     队头出队
     if(队头是问题的解){
          跳出循环，结束
     }

     扩展结点入队
}
```

具体的操作中，有两种不同的队列。

- 队列式分支限界法：按照队列先进先出（FIFO）原则选取下一个节点为扩展节点。
- 优先队列式分支限界法：按照优先队列中规定的优先级选取优先级最高的节点成为当前扩展节点

和回溯法一样，在算法开始之前，要在大脑中定义问题的解空间树，想好约束条件，再去实现代码。

作为一种搜索算法，广度优先搜索每生成一个子结点，就要提供指向它们父亲结点的指针。当解出现时候，通过逆向跟踪，找到从根结点到目标结点的一条路径。当然不要求输出路径，就没必要记父亲。生成的结点要与前面所有已经产生结点比较，以免出现重复结点，浪费时间和空间，还有可能陷入死循环。如果目标结点的深度与“费用”（如：路径长度）成正比，那么，找到的第一个解即为最优解，这时，搜索速度比深度搜索要快些，在求最优解时往往采用广度优先搜索；如果结点的“费用”不与深度成正比时，第一次找到的解不一定是最优解。广度优先搜索的效率还有赖于目标结点所在位置情况，如果目标结点深度处于较深层时，需搜索的结点数基本上以指数增长。