https://www.jianshu.com/p/75980691aa02

```
/*任务分配问题分支界限法*/
Begin
    1  根据限界函数计算目标函数的下界down;采用贪心法得到上界up;
    2. 将待处理结点活结点表初始化为空;
    3. for(i=l;i<=n;i++)
       x[i]= 0;
    4. k=l;i=0; //为第k个人分配任务，i为第k-1个人分配的任务
    5. while (k>=l)
    5.1 x[k]=l;
    5.2 while (x[k]<=n)
    5.2.1 如果人员k分配任务x[k]不发生冲突，则
    5.2.1.1 计算目标函数值lb;
    5.2.1.2 若lb<=up,则将i，<x[k]，k>lb存储在活结点表中;
    5.2.2 x[k]=x[k]+l;
    5.3 如果k==n且叶子结点的lb值在活结点表中最小,
        则输出该叶子结点对应的最优解;
    5.4 否则，如果k==n且活结点表中的叶子结点的lb值不是最小,则
    5.4.1 up=活结点表中的叶子结点最小的lb值;
    5.4.2 将活结点表中超出目标函数界的结点删除;
    5.5 i=活结点表中lb最小的结点的x丨k]值;
    5.6 k=活结点表中lb最小的结点的k值;k++;
End
```

```
/*0-1背包问题回溯算法*/
Begin
    设Backtrakc(i)表示对第i层结点的搜索
    1. if(i>n)则返回当前解bestp,结束算法
    2. if 当前背包重量 cw+w[i]<c,进入左子树
       2.1 cw+=w[i];当前价值cp+=v[i];
       2.2 搜索下一层结点 Backtrack(i+1);
       2.3 回退,cw-=w[i],cp=v[i]];
    3. 如果Bound(i+1)>bestp,进入右子树,即Backtrack(i+1)
End
```

```
/*0-1背包问题*/
int knapSack(int W, int wt[], int val[], int n)
{
   if (n == 0 || W == 0)
       return 0;
   if (wt[n-1] > W)
       return knapSack(W, wt, val, n-1);
   else return max( val[n-1] + knapSack(W-wt[n-1], wt, val, n-1),
                    knapSack(W, wt, val, n-1)
                  );
}
```

```
/*二分查找法分治策略*/
Begin
    输入有序数组a[],查找元素x,数组最左边下标i，最右边下标j
    i->0,j->a.length
  1.while(i>=j)循环执行:
    1.1 设置 m =(i+j)/2;
    1.2 if(x==a[m]) return m;
    1.3 if(x<a[m]) j=m-1; else i=m+1;
  2.return -1;
End
```

---

https://blog.csdn.net/qq_34379645/article/details/78614284

```

```

伪代码	含义	C/C++语言
缩进	程序块	{}
/ /	行注释	/ /
←	赋值	=
=	比较运算——等于	==
≠	比较运算——不等于	！=
≤	比较运算——小于或等于	< =
≥	比较运算——大于或等于	>=
for i←1 to n do	For循环	for(i=1;i<=n;i++){}
for i←n downto 1 do	For循环	for(i=n;i>=1;i–){}
while i<n do	Wihle循环	while(i<n){}
do while i<n	Do-While循环	do {} while(i<n)
repeat until i<n	Repeat循环	
if i<n else	If-Else语句	if(i<n){} else {}
return	函数返回值	return
A[0..n-1]	数组定义	int A[n-1]
A[i]	引用数组	A[i]
SubFun()	函数调用	SubFun()