# 16.1-2

这个方法与书上的方法相比就是反过来罢了。假设我们选了一个活动，我们想让剩下的活动能选得最多，意思就是我要让剩下的时间越多，反过来想就是本次选择的活动时间要越少。如果所有活动的结束时间按递增排好序，那么从左到右扫描一遍，等同于所有活动的开始时间按递减排好序，从左到右扫描一遍，伪代码如下

GREEDY-ACTIVITY-SELECTOR（S，f）

1. n length[S]

2. A

3. k 1

4. for m 2 up to n

5. if s[m] f[k]

6. A A{}

7. km

8. return A

此算法总是考虑当下最优的选择，不考虑将来的效应，而且与书上的经典算法针对同一个集合总是求得相同的解，所以求得的是最优解。

# 16.2-6

1. 求出价质比

2. 将价质比的中位数作为主元，小于这个数的价质比放在L数组中，等于的放在E数组中，大于的放在G数组中，求出每个数组的总重

3.

伪代码如下

LINEAR-TIME-FRACTION-KNAPSACK(V, W, )

1. let A[1..n] be a new array

2. for i = 1 to n

3. A[i] = V[i] / W[i]

4. return INNER(A, , sum)

INNER(A, , sum)

1. let L, E, G be a new array

2. n = A.length / 2

3. put

4. if ()

5. A=G

6. INNER(A, , sum)

7. else if ()

8. sum = sum

9. return sum

10. else

11. sum = sum +

12.

13. INNER(L, , sum)

# 16.3-7

可以用个最小堆来做，每个字符出现的频率作为关键字，伪代码如下：

TERNARY-HUFFMAN(C)

1. add a new node with frequency 0 to C if C.length is even
2. n = C.length
3. H = C
4. for i = 1 to
5. allocate a new node as z
6. z.left = H.min
7. z.mid = H.min
8. z.right = H.min
9. z.freq = z.left+z.mid+z.right
10. INSERT(H,z)
11. return H.MIN