《算法设计与分析》第4次作业　　A

姓名：叶宏庭　　学号：71118415

算法分析题

题目1：给定n个物品，物品价值分别为$P_1, P_2, ..., P_n$，物品重量分别为$W_1, W_2, ..., W_n$，背包容量为M。每种物品可部分装入到背包中。输出$X_1, X_2, ..., X_n$，$0 \leq X_i \leq 1$，使得$\sum_{1 \leq i \leq n} P_i X_i$最大，且$\sum_{1 \leq i \leq n} W_i X_i \leq M$。试设计一个算法求解该问题，答案需包含以下内容：证明该问题的贪心选择性，描述算法思想并给出伪代码。

答：

采用贪心算法，首先计算出每个物品的单位重量价值($V_i = P_i/W_i$)，判断当前背包的剩余容量，从单位价值大的物品开始装。假设剩余容量大于最大单位价值的物品的重量时，将其全部放入，若下于，则放入该物品至背包填满。　正确

贪心选择性：从最大单位价值开始装，装最大能装数量。

证明：

1. $k = 1$时，第一步装入单位价值最大的物品，使得背包内的价值最大；　第一步写详细点，下同

2. 假设前k步时加入背包的价值最大，对于第k+1步，若选择非单位价值最大的物品，那么被这个物品占据的空间的价值一定小于装入最大价值物品的价值，所以装入最大单位价值物品能够保证得到当前的最大总价值，即最优局部解。

综上：本算法可以得到最大总价值，最优解。

```
while remain != 0:
 if remain >= W[max_pervalue_good] : 若剩余空间大于单位价值最高的物品的总量，则全部装入 #
  X[max_pervalue_good] = 1;
  remain -= W[max_pervalue_good];
 else:
  X[max_pervalue_good] = remain / W[max_pervalue_good]; 空间不足，能装多少装多少 #
  remain = 0;
```

正确

题目2：假设你是一位很棒的家长，想要给你的孩子们一些小饼干。但是，每个孩子最多只能给一块饼干。对每个孩子 i，都有一个胃口值 g_i，这是能让孩子们满足胃口的饼干的最小尺寸；并且每块饼干 j，都有一个尺寸 s_j。如果 $s_j \geq g_i$，我们可以将这个饼干 j 分配给孩子 i，这个孩子会得到满足。你的目标是尽可能满足越多数量的孩子，并输出这个最大数值。注意：你可以假设胃口值为正。一个小朋友最多只能拥有一块

饼干。试设计一个算法求解该问题，答案需包含以下内容：证明该问题的贪心选择性，描述算法思想并给出伪代码。示例如下：
输入: [1,2], [1,2,3]
输出: 2
解释: 你有两个孩子和三块小饼干，2个孩子的胃口值分别是1,2。你拥有的饼干数量和尺寸都足以让所有孩子满足。所以你应该输出2。
答：

贪心选择性：让胃口小的先选，选择一块符合他胃口的最小饼干。 正确

证明：1. $k = 1$时，即只有一个小孩，所以让他选择即可。

2. 假设前k个小孩已经选择好饼干，第k+1个小孩去选饼干，他如果没有选择饼干，第k+2个小孩拿到了饼干，那么把k+2的饼干给k+1，一样可以满足k+1这个小孩，且拿到饼干的小孩总数没有变，所以k+1小孩去选饼干必然会得出一个最优解。

算法思想：让胃口小的先选，且只选满足自己胃口的最小饼干，这样不浪费，能够保证得出更优的解。

```
for i=1 to n:
 while (s[j] < g[i]) : 找到符合胃口的最小饼干 #
  j++;
 if (s[j] > g[i]) :
  counter++;
```

这个写的有点简单了

题目3：给定一个区间的集合，找到需要移除区间的最小数量，使剩余区间互不重叠。可以认为区间的终点总是大于它的起点，另外像区间 [1,2] 和 [2,3] 这样边界相互“接触”的区间，可以认为没有相互重叠。试设计一个算法求解该问题，答案需包含以下内容：证明该问题的贪心选择性，描述算法思想并给出伪代码。示例如下：
输入: [[1,2], [2,3], [3,4], [1,3]]
输出: 1
解释: 移除 [1,3] 后，剩下的区间没有重叠。
答：

该问题与活动安排问题为同一问题，我们只需找到最大能留多少个区间即可。

贪心选择性：按照区间的右端点排序，右端点小的先放入最后的集合中。 正确

证明： 1. 若第一个区间不放入结果集合中，那么结果集合$(S - B_1) \bigcup A_1$ 也是这个问题的一个最优解。(其中B_1为结果集合中的第一个区间，A_1表示原始集合的第一个区间(经过排序处理之后))

2. 假设前k个区间已经选出，此时在选择B_{k+1}时，若我们不选择A_{n+1}(剩余未选区间中右端点最小的区间),而是选择了A_{n+i}(i¿1),那么我们用A_{n+1}代替A_{n+i}，也得出了一个最优解。

综上：选择右端点最小的区间加入结果集合，一定能得出一个最优解。

```
# A[i].left    A[i].right 分别表示 $A[i]这个区间的左右端点$
for i=1 to n:
 if A[i].left >= S.end: 选择的区间左端点要小于结果集合的右端点
  S.append(A[i]) 将     $A[i]加入结果集合$
```

这个还是有点太简单了，最好能按照写可运行代码的流畅来写伪代码。