1.

设有 n 个物品，第 i 个物品的价值是 vi、重量是 wi, 假设物品可以任意分割，给定一个背包，其能容纳最大重量为 C，求该背包能容纳物品的最大价值。要求写出伪代码并分析算法正确性和复杂性。

伪代码:

```
Input: Items::Vector<Tuple<Value, Weight>>, C::Weight
Output: Result::Value

ItemHeap = new Heap(Items, x->x.v) //按价值整理成最大堆
while notEmpty(ItemHeap)
    i = ItemHeap.pop() //获取堆顶元素
    if i.w <= C
        Result += i.v
        C -= i.w
    else
        return Result + i.v * C / i.w //分割该物品塞满背包
return Result //程序执行到这里说明背包可以装下所有东西
```

正确性证明:

优化子结构: 设S是物品集合，假设X是问题的优化解，则X-{i}是C-i.w,S-{i}的优化解，其中i是价值最高的物品。因为假如Y更优的话，那么对于原问题，Y+{i}比X更优，与X是最优解矛盾

贪心选择性: 对|X|做归纳。当|X|为1时，显然选择价值最高的物品是最优解。假设|X|<k时选价值最高的k件物品时最优解，根据优化子结构的证明，只要子问题选择最优解即可，而i+1是子问题的最优解，因此|X|=k+1时也成立

复杂度分析

时间复杂度: 最好情况下背包只能装下一个东西，此时只需要建立堆O(n); 最坏情况下背包可以装下所有物品，此时一重循环每次弹堆需要logk的时间，总时间为O(nlogn)

空间复杂度: 一个最大堆，需要O(n)

2.

有 6 种硬币，面值是 1 分, 2 分, 5 分, 1 角，5 角，1 元，给定一个钱数 n，求出一个硬币组合，要求面值总和为 n 且硬币个数最少，假设每种硬币个数无限。要求写出伪代码并分析算法正确性和时间复杂性。

伪代码:

```
Input: n::Money
Output: Result::Vector<Tuple<Coin, Int>>

Coins = [1, 0.5, 0.1, 0.05, 0.02, 0.01]
for i in Coins
    if n > i
        Result += (i, n / i) //除法运算取整数商
        n = n mod i
return Result
```

正确性分析:

优化子结构: 假设X(i)表示第i中硬币的数量，{X(i)}是最优解。则对于问题S(n-1)，X[2..i]是该问题的优化解。假设有Y比X[2..i]更优，那么对于原问题，X[1]+Y比X更优，这与X是优化解矛盾。所以X[2..i]也是子问题的优化解

贪心选择性: 假设i为当前面值最大的硬币，则只要n>i，则应该选择一枚i硬币。因为假设Y为最优解且Y(i)1)枚硬币其面值为i，从而存在Y'比Y少上述k枚硬币而多一枚i硬币比Y更优，与Y最优矛盾。因此每次都应该选择面值最大的硬币

复杂度分析:

时间复杂度: 由于硬币数确定，循环次数也是确定的，时间复杂度对于钱数来说为O(1)

空间复杂度: 没有使用任何辅助数据结构，空间复杂度为O(1)

3.

存放于磁带上文件需要顺序访问。故假设磁带上依次存储了 n 个长度分别是 L[1],....,L[n]的文件，则访问第 k 个文件的代价为 $\sum_{j=1}^{k} L[j]$ 。现给定 n 个文件的长度 L[1],....,L[n]，并假设每个文件被访问的概率相等，试设计一个算法输出这 n 个文件在磁带上的存储顺序使得平均访问代价最小。。答案要求包含以下内容：（1）证明问题具有贪心选择性；（2）证明问题具有优化子结构；（3）给出算法并分析算法的时间

复杂度。

算法:

按照长度从小到大排序即可

正确性证明:

设A[n]为1...n的一个排列使得L[A[1]] <= L[A[2]] <= ... <= L[A[n]]，设X为问题的优化解。

优化子结构:

对于子问题P[2..n]，X[2..n]是其优化解，因为假设Y比X[2..n]更优，则对于原问题，Y+X[1]比X更优，这与X是优化解矛盾，因此X[2..n]是P[2..n]的优化解

贪心选择性:

假设X比A更优，则必∃ $i < j\ s.t.\ L_{X_i} > L_{X_j}$，设Y为交换X中i和j两项构成的序列

$$\sum_{k=1}^{n}\sum_{p=1}^{k} L_{X_p} - \sum_{k=1}^{n}\sum_{p=1}^{k} L_{Y_p} = (j-i)(L_{X_i} - L_{X_j}) > 0$$

因此Y的代价更小。如果Y=A，则命题成立，否则根据上述证明过程，还存在Y'比Y更优，可以根据逆序对的数量归纳得到A是优化解。

4.

设有 n 个正整数，将它们连接成一排，组成一个最大的多位整数。
例如：n=3 时，3 个整数 13，312，343，连成的最大整数为 34331213。
又如：n=4 时，4 个整数 7，13，4，246，连成的最大整数为 7424613。
输入是 n 个正整数，输出是这 n 个正整数连成的最大多位整数，要求用贪心法求解该问题。答案要求包含以下内容：（1）证明问题具有贪心选择性；（2）证明问题具有优化子结构；（3）写出算法伪代码并分析算法的时间复杂度。

伪代码:

```
Input: Numbers::Vector<Int>
Output: Result::Int

for i in sort(Numbers) //按从大到小的顺序排序
    Result.concat!(i) //将数字作为字符串拼接到结果的后面
return Result
```

正确性分析:

优化子结构: 假设序列X是一个优化解，则X[2..n]是子问题2..n的优化解，因为假设Y比X[2..n]更优，则X[1]+Y比X更优，这与X是最优解矛盾，因此X[2..n]是子问题的优化解

贪心选择性:

设A[n]为这n个数的一个排列使得A[1] <= A[2] <= ... <= A[n]，设X为问题的优化解。

假设X比A更优，则必∃ $i < j\ s.t.\ X_i > X_j$，设Y为交换X中i和j两项构成的序列，由X_i>X_j以及X_i的位数不小于X_j两点易证X的小于Y（证明过程类似第3题），从而Y是更优的解，归纳逆序对的数量即可证明X=A。

复杂度分析:

时间复杂度: 排序O(nlogn)

空间复杂度: 如果不能修改输入，则需要复制原数组以进行排序，复杂度为O(n)

5.

设 x1, x2, …., xn 是实数轴上的 n 个点，若用单位长度的闭区间覆盖这些点，至少需要多少单位长度闭区间？

```
Input: X::Vector<Real>
Output: Result::Int

current = -∞
for i in sort(X) //从小到大排序
    if i <= current + 1
        continue
    else
        current = i
        Result += 1
return Result
```

6.

考虑下述最小生成树算法，初始时，G 中的每个顶点被视为一个单结点的树，不选择任何边，在每一步，为每棵树选择一条最小权的边 e，是的 e 只有一个顶点在 T 中，如果必要的话，出去所选边的备份，当只得到一棵树或者所有边都被选中了，那么终止算法。证明算法的正确性并且求出算法的最大步数。

没看懂

7.

G=(V, E)是一个具有 n 个顶点 m 条边的连通图，且可以假设边的代价为正且各不相同，设，定义 T 的瓶颈边是 T 中代价最大的边，G 的一个生成树 T 是一棵最小瓶颈生成树，如果不存在 G 的生成树 T'是的它具有代价更小的瓶颈边。问:(1)G 的每棵最小瓶颈树一定是 G 的一棵生成树吗？证明或者给出反例; (2) G的每棵生成树都是 G 的最小瓶颈树吗？证明或者给出反例。

题目补充: 问题中的生成树均指最小生成树

(1). 否，反例如下:

```
a - b: 2
b - c: 2
a - c: 1
```

对于这个图，a-b-c是一棵最小瓶颈树，它的瓶颈为2，但是它的代价为4，存在c-a-b是代价为3的更小的生成树。

(2). 是，因为假如T是瓶颈为x的最小生成树，且存在一颗瓶颈树T'，其所有边的代价都小于x。考虑图S为T删去一条代价为x的边构成的图，由于T'是连通的，T'中存在一条边e可以使S连通，且根据T'的定义，e<x。因此S+e是比T更小的一棵生成树，这与T是最小生成树矛盾。因此T也是G的最小瓶颈树

8.

给定 n 个自然数 d1, d2, …, dn, 设计算法，在多项式时间确定是否存在一个无向图 G，使它的结点度数准确地就是 d1, d2, …, dn， 要求 G 中在任意两个结点之间至多有一条边，且不存在一个结点到自身的边。

```
Input: D::Vector(Int)
Output: G 度数为D的图，或者null如果不存在这样的图

unmetNodes = []
for i in sort(D) // 按从大到小排序
    p = new Node(i)
    G.addNode(p)
    for j in unmetNodes
        if p.degree <= 0
            break
        j.degree -= 1
        p.degree -= 1
        G.addEdge(j,p)
        if j.degree <= 0
            unmetNodes.delete(j)
    if p.degree > 0
        unmetNodes.add(p)
if empty(unmetNodes)
    return G
else
    reutrn null
```

9.

考虑一种特殊的 0-1 背包问题，有 n 个物品，每个物品价值和重量都相等，背包能容纳的最大重量是 C, 回答下列问题:

若物品的重量(价值)分别是 1, 2, …,2^n，证明该 0-1 背包问题可以用贪心法求解并写出该贪心法。

请写出一个物品重量(价值)序列，使得上述贪心法无法得到最优解。

(1).

优化子结构: 假设X是原问题的优化解，则X[2..n]是2,4,..,2^n ，背包重量为C-1的背包问题的优化解，因为假设Y是更优的解，那么对于原问题，Y+1是比X更优的解，这与X是最优解矛盾。

贪心选择性: 设$C \geq 2^k$，如果不选2^k，则最大可选的价值只有$1+2+..+2^{k-1}=2^k-1<2^k$，不如选择2^k。由k的任意性知这个论断对于任意子问题也成立。

伪代码:

```
Input: n::Int, C::Weight
Output: X::Vector<Bool>

for i in n..1
    if 2^i > C
        X[i] = 1
        C -= 2^i
return X
```

(2).

2,3,4...n-1的序列就无法用贪心法求解。例如当n=5,C=5时，可选的物品为2,3,4，此时贪心算法会选择4，而最优解为2+3

10.

考虑下述“逆贪心”算法，输入是连通有权无向图 G,用邻接表描述

```
sort the edges E of G By weight
for i ← 1 to |E|
    e ← ith heaviest edge in E
    if G \ e is connected
        remove e from G
```

(1). 该算法的最坏运行时间是多少？在什么情况下发生?
(2). 证明这个算法可以找到 G 的最小生成树。

(1). 对于边n来说，最坏运行时间为O(n^2)，在权重最小的边位于最小生成树之内的时候发生

(2).

显然输出的是生成树，下证其为最小生成树。

假设T'是比输出的T更小的生成树，任取在T'中但不在T中的边e，则T+e中存在环。由于算法先删重边，所以e必然是该环中最重的边，否则T中会包含e。由e的任意性知T'中的任何边都不能替换T中的边，因此T是不比T'大的生成树，这与T'比T小矛盾。因此T是最小生成树。