算法分析第4次作业

小组编号：23

本次作业负责人:黄勖

1 算法实现题4-1 答案：

4-1 会场安排问题。

问题描述： 假设要在足够多的会场里安排一批活动，并希望使用尽可能少的会场。设计一个有效的贪心算法进行安排。（这个问题实际上是著名的图着色问题。若将每个活动作为图的一个顶点，不相容活动间用边相连。使相邻顶点着有不同颜色的最小着色数，相当于要找的最小会场数。）

算法设计： 对于给定的 k 个待安排的活动，计算使用最少会场的时间表。

数据输入： 由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 k，表示有 k 个待安排的活动。接下来的 k 行中，每行有 2 个正整数，分别表示 k 个待安排的活动的开始时间和结束时间。时间以 0 点开始的分钟计。

结果输出： 将计算的最少会场数输出到文件 output.txt。

(1) 算法描述:

1. 从输入中读取k个待安排的活动，每个活动有开始时间和结束时间。

2. 对活动按照结束时间非递减排序，如果结束时间相同，则按照开始时间非递减排序。

3. 初始化一个空的会场列表，用于存储已安排的活动。

4. 遍历排序后的活动列表：

   - 如果会场列表为空，将当前活动分配给一个新的会场。
   - 否则，遍历已有的会场列表，找到一个会场，使得当前活动的开始时间大于等于该会场中最后一个活动的结束时间，然后将当前活动分配给这个会场。如果没有找到合适的会场，新建一个会场并将当前活动分配给它。

5. 继续遍历所有活动，直到所有活动都被安排。

6. 输出已分配的会场数，即为最少会场数。

(2) 贪心选择性质证明:

该算法的贪心选择性质在于，每次选择结束时间最早的活动，并将其分配给已有的会场或创建一个新的会场。这种选择策略确保了每个会场的结束时间尽可能早，以便容纳更多的活动。

证明贪心选择性质的关键在于排序步骤，将活动按照结束时间非递减排序，如果结束时间相同，则按照开始时间非递减排序。这样选择结束时间最早的活动是一个合理的决策，因为它会释放会场以容纳更多的活动。选择其他活动可能导致需要更多的会场，因此贪心选择性质得以满足。

因此，该算法是一个有效的贪心算法，可以用于解决会场安排问题，并保证得到最少会场数的安排。

(3) 最优子结构性质证明:

假设 $(a_1, a_2, \ldots, a_k)$ 是 n 个活动的集合 $e = \{1, 2, \ldots, n\}$ 所需会场的最优解。设 a_1 中安排了 m 个相容的活动，那么也就是说 $(n-m)$ 个活动完全安排需要 $k-1$ 个会场。假设 $(n-m)$ 个活动安排只需要 $k-2$ 个会场或则更少的会场。也就是说 n 个活动安排只需要 $k-1$ 个会场或者更少的会场就可以安排完，则前后出现矛盾。

(4) 时间复杂度: $O(n^2)$

本题分工：小组共同讨论，黄勖编写

2 算法实现题4-2 答案:

4-2 最优合并问题。

问题描述：给定 k 个排好序的序列 $s_1, s_2, \cdots, s_k$，用 2 路合并算法将这 k 个序列合并成一个序列。假设采用的 2 路合并算法合并 2 个长度分别为 m 和 n 的序列需要 $m+n-1$ 次比较。试设计一个算法确定合并这个序列的最优合并顺序，使所需的总比较次数最少。

为了进行比较，还需要确定合并这个序列的最差合并顺序，使所需的总比较次数最多。

算法设计：对于给定的 k 个待合并序列，计算最多比较次数和最少比较次数合并方案。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 k，表示有 k 个待合并序列。接下来的 1 行中，有 k 个正整数，表示 k 个待合并序列的长度。

结果输出：将计算的最多比较次数和最少比较次数输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
4	78 52
5 12 11 2	

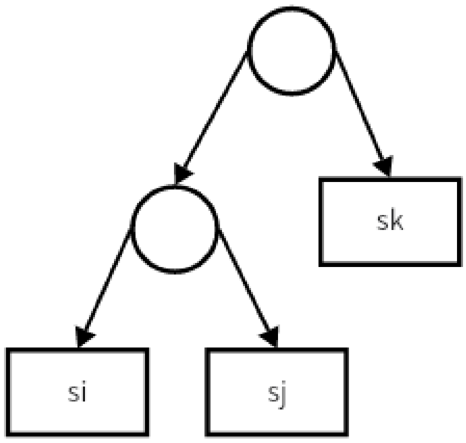
(1) 算法描述:

1. 将序列$s1, s2, \ldots\ldots, sk$按照序列长度从小到大进行堆排序
2. 每次选择最短的两个序列s_i, s_j合并为一个新的序列s_{i+j}重新插入后重复该步骤2，直到序列处理完毕

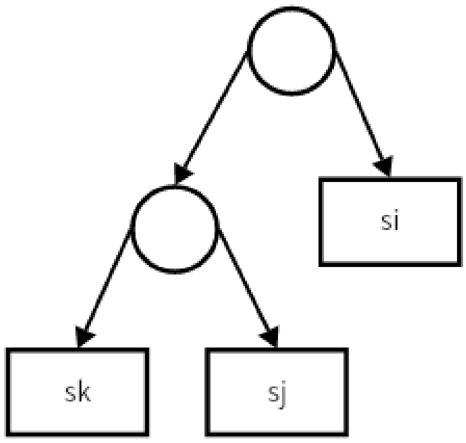
(2) 贪心选择性质证明:

原问题$s = s1, s2, \ldots, sk$

设最优解(即最少合并次数)为C且si, sj是最短的两个序列，构造二叉合并树如下:



假设还存在另一个最优解C，未先合并si, sj（$sk > si \& sk > sj$）



$C - C' = (s_i + s_j + 1) + [(s_i + s_j + 1) + s_k + 1] - (s_k + s_j + 1) + [(s_k + s_j + 1) + s_i + 1] = s_i - s_k < 0$

故C'不是最优解，C为最优解，具有贪心选择性质

(3) 最优子结构性质证明:

$s = s1, s2, \ldots, sn$，记s_i和sj合并$(si + j)$

将问题s划分为子问题$s - si, sj$和$(si + j)$

$s = (s - s_i, s_j) \cup (si + j)$

因此最优值$C_n : C_n = C_{n-2} + (s_i + s_j + 1)$

假设C_{n-2}不是最优值，则存在C'_{n-2}使得$C'_n = C'_{n-2} + (s_i + s_j + 1) < C_n$

与C_n是最优值矛盾，因此C_{n-2}是最优值

具有最优子结构性质

(4) 时间复杂度:

堆排序的时间复杂度O(nlogn)，合并O(m+n-1)

$$(n-1)(O(nlogn)+O(m+n-1))=O(n^2logn)$$

(5) 说明:

同理总比较次数最多时，每次选取最长的两个序列进行合并即可

本题分工：小组共同讨论，李嘉琪编写

3　算法实现题4-4 答案:

4-4　磁盘文件最优存储问题。

问题描述：设磁盘上有 n 个文件 $f_1, f_2, \cdots, f_n$，每个文件占用磁盘上的 1 个磁道。这 n 个文件的检索概率分别是 $p_1, p_2, \cdots, p_n$ 且 $\sum_{i=1}^{n} p_i = 1$。磁头从当前磁道移到被检信息磁道所需的时间可用这两个磁道之间的径向距离来度量。如果文件 f_i 存放在第 i（$1 \leqslant i \leqslant n$）道上，则检索这 n 个文件的期望时间是 $\sum_{1 \leqslant i \leqslant j \leqslant n} p_i p_j d(i,j)$。式中，$d(i,j)$是第 i 道与第 j 道之间的径向距离$|i-j|$。

磁盘文件的最优存储问题要求确定这 n 个文件在磁盘上的存储位置，使期望检索时间达到最小。试设计一个解此问题的算法，并分析算法的正确性与计算复杂性。

算法设计：对于给定的文件检索概率，计算磁盘文件的最优存储方案。

数据输入：由文件 input.txt 给出输入数据。第 1 行是正整数 n，表示文件个数。第 2 行有 n 个正整数 a_i，表示文件的检索概率。实际上第 k 个文件的检索概率应为 $a_k \Big/ \sum_{i=1}^{n} a_i$。

结果输出：将计算的最小期望检索时间输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
5	0.547396
33 55 22 11 9	

(1) 算法描述:

- 排序规则:
  - 先对文件检索的概率从大到小排序，设排序后有 $p_1 \geq p_2 \geq \ldots \geq p_n$.
- 贪心策略: f_1 占中心磁道，f_2 和 f_3 分居 f_1 的两侧，f_4 在 f_2 的左侧，f_5 在 f_3 的右侧，……

(2) 贪心选择性质证明:

设第 i 个文件检索概率为 P_i, 第 x 个文件与其他文件的期望时间为

$$T_x=\sum_{i=1}^{n} p_x \cdot p_i \cdot |x-i|$$

要求 T 最小，则让每一项的 T_x 都尽为最小值，T_x 中 p_x 为定值，所以应让 $\sum_{i=1}^{n} p_i \cdot |x-i|$ 最小，那么就让概率最大的文件乘以最小的检索时间，较大的文件乘以较小的检索时间即可，即证明两个数组的最小乘积和。

证明如下：

要求两个数组的最小成绩和，假设现在有 $nums1$ 中的任意两个元素为 a, b, $nums2$ 中的元素为 x, y

, 并且有 $a \le b$, $x \le y$, 则有以下两种乘积组合:
$temp1 = a \cdot x + b \cdot y$
$temp2 = a \cdot y + b \cdot x$
比较两种乘积组合的大小，有:
$temp2 - temp1 = a \cdot (y-x) + (x-y) = a \cdot (y-x) - b \cdot (y-x) = (a-b) \cdot (y-x) \le 0$
即:
$temp2 \le temp1$
因此选择第二种组合，即选取 $nums1$ 中较小元素与 $nums2$ 中较大元素进行相乘.

(3) 最优子结构性质证明:

在选择最大检索概率后，原问题转化为对剩下文件安排的子问题，即如果 A 是原问题的最优解，则 $A' = A - \{p_{\max}\}$ 是剩余文件安排问题的最优解。
如果存在解 B' 优于 A'，则 $B' + \{p_{\max}\}$ 要优于 $A' + \{p_{\max}\} = A$，与 A 的最优性质矛盾.

(4) 时间复杂度: $O(n^2)$

本题分工：小组共同讨论，黄勖编写

4 算法实现题4-6 答案:

4-6　最优服务次序问题。

问题描述： 设有 n 个顾客同时等待一项服务，顾客 i 需要的服务时间为 t_i（$1 \leqslant i \leqslant n$）。应如何安排 n 个顾客的服务次序才能使平均等待时间达到最小？平均等待时间是 n 个顾客等待服务时间的总和除以 n。

算法设计： 对于给定的 n 个顾客需要的服务时间，计算最优服务次序。

数据输入： 由文件 input.txt 给出输入数据。第 1 行是正整数 n，表示有 n 个顾客。接下来的 1 行中，有 n 个正整数，表示 n 个顾客需要的服务时间。

(1) 算法描述:

1. 将所有顾客按照服务时间从小到大堆排序

2. 计算每个顾客的等待时间$wi = (t1 + t2 + \ldots + ti - 1)$

   其中ti表示第i个顾客的服务时间

3. 计算平均等待时间$W = (w1 + w2 + \ldots + wn)/n$，其中$wi$表示在顾客$i$之前等待的顾客所需服务时间之和

4. 输出服务次序和平均等待时间

(2) 贪心选择性质证明:

假设顾客i等待时间为wi，平均等待时间W

$$W = \frac{\sum_{i=1}^{n} w_i}{n}$$

最优服务A中t_1满足条件：$t_1 <= t_i \ (1 < i \leq n)$

服务序列$A = t_1, t_2, \ldots, t_i \ldots, t_n$

反证法证明：假设t_1不是最小的，不妨设$t_1 > t_i (i > 1)$

设存在另一最优服务序列$B = t_i, t_2, \ldots, t_1 \ldots, t_n$

$$WA - WB = (t_1 - t_i) \times (i - 1) > 0$$

即$WA > WB$，这与A是最优服务相矛盾

最优服务次序问题满足贪心选择性质

(3) 最优子结构性质证明:

若A是原问题T的最优解，则$A' = t_2, \ldots, t_i, \ldots, t_n$是服务次序问题子问题T'的最优解

假设A'不是子问题T'的最优解，其子问题的最优解为B'，则有$TB' < TA'$

而根据TA的定义得：$TA' + t_1 = TA$

因此$TB' + t_1 < TA' + t_1 = TA$

即存在一个比最优值TA更短的总等待时间，而这与TA为问题T的最优值相矛盾

因此A'是子问题T'的最优值，满足最优子结构性质

(4) 时间复杂度:

排序时间复杂度为$O(nlogn)$，计算时间复杂度为$O(n)$

算法时间复杂度主要来自排序：

$$O(nlogn)$$

本题分工：小组共同讨论，李嘉琪编写

5 算法实现题4-8 答案：

4-8　*d* 森林问题。

问题描述：设 T 是一棵带权树，树的每条边带一个正权，S 是 T 的顶点集，T/S 是从树 T 中将 S 中顶点删去后得到的森林。如果 T/S 中所有树的从根到叶的路长都不超过 d，则称 T/S 是一个 d 森林。

① 设计一个算法求 T 的最小顶点集 S，使 T/S 是 d 森林（提示：从叶向根移动）。

② 分析算法的正确性和计算复杂性。

③ 设 T 中有 n 个顶点，则算法的计算时间复杂性应为 $O(n)$。

算法设计：对于给定的带权树，计算最小分离集 S。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 n，表示给定的带权树有 n 个顶点，编号为 $1, 2, \cdots, n$。编号为 1 的顶点是树根。接下来的 n 行中，第 $i+1$ 行描述与 i 个顶点相关联的边的信息。每行的第 1 个正整数 k 表示与该顶点相关联的边数。其后 $2k$ 个数中，每 2 个数表示 1 条边。第 1 个数是与该顶点相关联的另一个顶点的编号，第 2 个数是边权值。$k=0$，表示相应的结点是叶结点。文件的最后一行是正整数 d，表示森林中所有树的从根到叶的路长都不超过 d。

结果输出：将计算的最小分离集 S 的顶点数输出到文件 output.txt。如果无法得到所要求的 d 森林则输出 "No Solution!"。

输入文件示例	输出文件示例
input.txt	output.txt
4	1
2 2 3 3 1	
1 4 2	
0	
0	
4	

(1) 算法描述:

- 贪心策略：使用贪心的思想，从叶子节点向上遍历，每个节点的 `priceOfson` 记录该节点子节点中路径长度最大值，若 `priceOfson` 大于 `d` 则删除该节点，每一次遍历访问无子节点或子节点均被访问或删除，且未被标记的节点，直到所有节点均被访问，得到最优解。

(2) 贪心选择性质证明:

从叶子节点回溯，到达某一层的节点如果路径长度超过 `d` 时，假设我们删除这个节点的子节点，那么如果存在从另一个叶子节点另一条路径回溯路径长度超过 `d`，我们还需要删除另一个节点。也即是，对于这个节点的子节点而言：我们可能会删除 2 个以上的节点。如果我们直接删除这个节点，那么这一个点的子节点无需被删除，只需要删除一次超过 `d` 的这一个节点即可。

(3) 最优子结构证明:

对于森林而言，如果要求整体满足为 `d` 森林且删除的顶点数量最少，应该使得每一个子森林删除的顶点数量最少。如果 A 是原问题的最优解，则 $A' = A - n(\text{子树})$ 是剩余问题的最优解。如果存在解 B' 优于 A'，则 $B' + n(\text{子树})$ 要优于 $A' + n(\text{子树}) = A$，与 A 的最优性质矛盾。

(4) 时间复杂度: $O(n)$

本题分工：小组共同讨论，黄勖编写

6 算法实现题4-9 答案：

4-9 虚拟汽车加油问题。

问题描述：一辆虚拟汽车加满油后可行驶 n km。旅途中有若干加油站。设计一个有效算法，指出应在哪些加油站停靠加油，使沿途加油次数最少。并证明算法能产生一个最优解。

算法设计：对于给定的 n 和 k 个加油站位置，计算最少加油次数。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 2 个正整数 n 和 k，表示汽车加满油后可行驶 $n$km，且旅途中有 k 个加油站。接下来的 1 行中有 k+1 个整数，表示第 k 个加油站与第 k–1 个加油站之间的距离。第 0 个加油站表示出发地，汽车已加满油。第 k+1 个加油站表示目的地。

结果输出：将计算的最少加油次数输出到文件 output.txt。如果无法到达目的地，则输出"No Solution"。

输入文件示例	输出文件示例
input.txt	output.txt
7 7	4
1 2 3 4 5 1 6 6	

(1) 算法描述:

1. 初始化变量`refill_count`记录加油次数，`curpos` 表示当前汽车所在的位置，`tank`表示汽车当前油箱中的油量。
2. 遍历加油站位置
   - 若汽车能够到达当前加油站，则不在当前位置加油继续前进；
   - 如果汽车无法到达当前加油站，则在上一个加油站加满油，更新`refill_count`，并更新`curpos`和`tank`
3. 返回`refill_count`

(2) 贪心选择性质证明:

假设k个加油站位置分别为$p_1, p_2, \ldots\ldots p_k$，任意相邻的$p_i$到$p_j$的距离为$d(i,j)$

设存在最优解$A = p_1, \ldots\ldots, p_i, p_j, \ldots p_m$，保证任意相邻的pi和pj满足$n < d(i,j)$

假设还存在另一个最优解$B = p_1, \ldots\ldots, p_i, p_j, p_k \ldots, pm$允许其中存在某一相邻的$p_i, p_j, p_k$满足$n \geq d(i,k)$，则

则对于加油次数有$T(A) < T(B)$与B是最优解相矛盾

因此满足贪心选择性质

(3) 最优子结构性质证明:

原问题T的最优解$A = p_1, \ldots\ldots, p_i, p_j, \ldots p_m$，保证任意相邻的$p_i$和$p_j$满足$n < d(i,j)$

$p_1, p_2, \ldots\ldots, p_k$中，p1一定选择，找到位置t保证t>1且$d(1,t) \geq n \&\& d(1,t-1) \leq n$

则子问题T'的$A-A_1=p_t,\ldots\ldots,p_i,p_j,\ldots p_m$，$(d(1,t)\geq n\&\&d(1,t-1)\leq n)$,$A_1$表示序列$A$的第一个元素

$$T(A)=T(A-A_1)+d(A_1,t)$$

假设$T(A-A_1)$不是最优解，则存在B使得$T(A)'=T(B)+d(A_1,t)<T(A)$

与$T(A)$是最优解相矛盾，因此$T(A-A_1)$是最优解

满足最优子结构性质

(4) 时间复杂度:

遍历k个加油站，堆每个加油站检查汽车是否能够到达该加油站，因此算法的时间复杂度为

$$O(k)$$

本题分工：小组共同讨论，李嘉琪编写

7 算法实现题4-11 答案:

4-11 删数问题。

问题描述：给定 n 位正整数 a，去掉其中任意 $k\leqslant n$ 个数字后，剩下的数字按原次序排列组成一个新的正整数。对于给定的 n 位正整数 a 和正整数 k，设计一个算法找出剩下数字组成的新数最小的删数方案。

算法设计：对于给定的正整数 a，计算删去 k 个数字后得到的最小数。

数据输入：由文件 input.txt 提供输入数据。文件的第 1 行是 1 个正整数 a。第 2 行是正整数 k。

结果输出：将计算的最小数输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
178543	13
4	

(1) 算法设计:

1. 将正整数a表示为一个字符串以便操作。
2. 从左到右遍历字符串，对于每一位数字，如果后面的数字比当前数字大，则记录该数字下标，最后将当前数字删除(即最大最前面的那一个数字)，重复这一过程直到删除k个数字。
3. 将剩下的数字按原次序排列组成一个新的正整数。

(2) 贪心选择性质证明:

从左到右遍历字符串，如果后面的数字比当前数字大，删除当前数字可以使得新数最小。如果不删除当前数字，后面的数字比它大的可能会导致新数变大，所以贪心地删除当前数字是合理的。

(3) 最优子结构性质证明:

在进行了贪心选择后，原问题 T 就变成了对 N 如何删去 $k-1$ 个数的问题 T'，是原问题的子问题。若 $A=(x_j, A')$ 是原问题 T 的最优解，则 A' 是子问题 T' 的最优解，其最优值为 $T_{A'}$。

证明：假设 A' 不是子问题 T' 的最优解，其子问题的最优解为 B'，其最优值记为 $T_{B'}$，则有 $T_{B'} < T_{A'}$。根据 T_A 的定义可知：$T_A = T_{A'} + x_j \cdot 10^{(n-j)}$，而 $T_{B'} < T_{A'}$，因此有 $T_{B'} + x_j \cdot 10^{(n-j)} < T_{A'} + x_j \cdot 10^{(n-j)} = T_A$。即存在一个由数 a 删去的 $n-1$ 位数比最优值 T_A 更小。这与 T_A 为问题 T 的最优值相矛盾。因此，A' 是子问题 T' 的最优解。

因此，删数问题满足最优子结构性质。

(4) 时间复杂度： $O(n)$

本题分工：小组共同讨论，黄勖编写

8 算法实现题4-15 答案：

4-15 最优分解问题。

问题描述： 设 n 是一个正整数。现在要求将 n 分解为若干互不相同的自然数的和，且使这些自然数的乘积最大。

算法设计： 对于给定的正整数 n，计算最优分解方案。

数据输入： 由文件 input.txt 提供输入数据。文件的第 1 行是正整数 n。

结果输出： 将计算的最大乘积输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
10	30

(1) 算法描述:

1. 初始化变量$last=2$，初始化数组`elements`保存每个分解的值
2. 循环处理n：

   若$n-last>last$：$last++$，$n-=last$并且保存到数组`elements`中

   若$n-last \le last$：从后往前循环n-last个元素，每个`elements`的元素值加1，结束循环
3. 遍历数组`elements`求最大乘积

(2) 性质证明:

***已知性质**：对任意$a+b=const$，当$|a-b|$取min时，ab取到max

A表示将n分解为k个数的乘积：$A=n1,n2,\ldots n_i,n_{i+1}\ldots,nk$保证任意相邻$n_i,n_{i+1}$的差值 $1 \le d(n_i,n_{i+1}) \le 2$

假设存在另一情况B得到最大乘积乘积$B=n1,n2,\ldots n_i'\ldots,nk$存在相邻$n_i',n_{i+1}'$的差值$d(n_i',n_{i+1}')>2$

则根据***已知性质**有$n_i n_{i+1} > n_i' n_{i+1}'$

$$A-B=\prod_{j=1}^{k}n_j-(\prod_{j=1}^{i-1}n_j\times\prod_{j=i+2}^{k}n_j\times n_i'\times n_{i+1}')=\prod_{j=1}^{i-1}n_j\times\prod_{j=i+2}^{k}n_j\times(n_in_{i+1}-n_i'n_{i+1}')>0$$

与$\prod B>\prod A$矛盾

因此$\prod A$为最大值

(3) 时间复杂度:

考虑最坏情况下分解到最优一个数的值小于前一个元素，需要从后往前遍历，故时间复杂度为

$$O(n^2)$$

本题分工：小组共同讨论，李嘉琪编写