1、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 $L[1],\ldots,L[n]$ 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 $L[1],\ldots,L[n]$，并假设每个文件被访问的概率相等，试给出这 n 个文件在磁带上的存储顺序使得平均访问代价最小。

（1）简述算法采用的贪心策略。

将文件按照从小到大的顺序存储在磁带上。

（2）表述并证明问题的贪心选择性。

给定 n 个文件的长度分别是 $L[1],\ldots,L[n]$，其中访问第 k 个文件的代价为$\sum_{j=1}^{k} L[j]$，每个文件被访问的概率相等，令$j = arg\min_i L[i]$，则存在一个存储方案T，T的平均访问代价是所有存储方案中最小的，而且T中$L[j]$排在第一位.

证明：令T'是一个平均访问代价最小的存储方案，若$L[j]$在T'中排在第一位，则得证。

否则，令文件j（长度最小的文件）在T'中排在第x位，T'中文件长度依次为$L[i_1]$，$L[i_2]$，…，$L[i_n]$，其中第m个文件的访问代价为$\sum_{p=1}^{m} L[i_p]$。交换T'中排在第1的文件和文件j（在T'中排在第x个）后，得到存储方案T，往证T的平均访问代价不大于T'。

T'的平均访问代价如下：

$$C(T') = \frac{1}{n}\sum_{m=1}^{n}\sum_{k=1}^{m} L[i_k] = \frac{1}{n}\sum_{m=1}^{x-1}\sum_{k=1}^{m} L[i_k] + \frac{1}{n}\sum_{m=x}^{n}\sum_{k=1}^{m} L[i_k].$$

T的平均访问代价如下：

$$C(T) = \frac{1}{n}\sum_{m=1}^{x-1}\left(L[j] + \sum_{k=2}^{m} L[i_k]\right) + \frac{1}{n}\sum_{m=x}^{n}\sum_{k=1}^{m} L[i_k]$$

$$\le \frac{1}{n}\sum_{m=1}^{x-1}\sum_{k=1}^{m} L[i_k] + \frac{1}{n}\sum_{m=x}^{n}\sum_{k=1}^{m} L[i_k] = C(T').$$

由此得证T具有最小的平均访问代价，而且T中$L[j]$排在第一位.

（3）表述并证明问题的优化子结构。

给定包含 n 个文件的集合S，其中文件的长度分别是 $L[1],\ldots,L[n]$，访问第 k 个文件的代价为$\sum_{j=1}^{k} L[j]$，每个文件被访问的概率相等，令$j = arg\min_i L[i]$，令T是平均访问代价最小的存储方案，而且T中$L[j]$排在第一位，记$T = j, i_1, \ldots, i_{n-1}$，则$T' = i_1, \ldots, i_{n-1}$是$S - \{j\}$的平均访问代价最小的存储方案.

证明：记$C(P)$为存储代价P的平均访问代价，则有

$$C(T) = \frac{1}{n}\left(L[j] + \sum_{m=1}^{n-1}\left(L[j] + \sum_{k=1}^{m} L[i_k]\right)\right)$$

$$= L[j] + \frac{1}{n}\sum_{m=1}^{n-1}\sum_{k=1}^{m} L[i_k] =$$

$L[j] + \frac{n-1}{n} C(T')$.

若$T' = i_1, \ldots, i_{n-1}$不是$S - \{j\}$的平均访问代价最小的存储方案，则存在$T''$是$S - \{j\}$的平均访问代价最小的存储方案，通过与上式类似的推导，可以得到文件j和T''组成的存储方案的平均访问代价为

$$L[j] + \frac{n-1}{n} C(T'') < L[j] + \frac{n-1}{n} C(T') = C(T)。$$

与 T 是平均访问代价最小的存储方案矛盾，因此$T' = i_1, \ldots, i_{n-1}$是$S - \{j\}$的平均访问代价最小的存储方案。

（4）写出算法的伪代码。

略。

（5）分析算法的时间复杂度。

$O(n\log n)$.

2、程序员接到 n 项编程任务，第 i 项任务需要在时间点 E_i 之前完成，完成第 i 项任务需要的时间（预计）为 t_i。每个任务顺利完成之后，程序员将得到固定的报酬 a；如果未能按时完成某项任务，则程序员不能得到该任务的酬金。试设计一个贪心算法，为程序员安排工作计划，使其获得酬金最大。

（1）简述算法采用的贪心策略。

每次选择当前可以选择的(即$t_i \le E_i$)长度最小的任务，放入任务队列中，已经选择的任务按照最晚结束时间从小到大排序.

（2）表述并证明问题的贪心选择性。

设$A=\{a_1, a_2, \ldots, a_n\}$，$a_i=(t_i, T_i)$，$p=arg_i \min\limits_{1\le i\le n} t_i$. 必然存在$S=\{a_{j_1}, a_{j_2}, \ldots, a_{j_k}\}$是$A$的最优调度，$S$包含$a_p$，并且$S$中的任务按照最晚完成时间升序排列.

证明：首先证明存在优化解包含任务p. 令T是A的一个优化调度，如果T包含任务p，则得证优化解包含任务p. 否则，用任务p替换T中的第一个任务，因为任务p的长度不超过T中第一个任务的长度，T中的其余任务的完成时间不会后延，于是得到一个新的调度P，其中包含任务数量和最优解T相同，因此P也是一个最优调度，并且包含任务p.

若P中的任务按照最晚完成时间从小到大排序，则贪心选择性得证. 否则，考虑 P 中任意相邻的两个任务a_x和a_y，其中a_x在前，而且任务a_y的完成时间必然不超过E_y. 如果$E_x > E_y$，则交换任务a_x和a_y后，其他任务的完成时间不变，a_y的完成时间提前，a_x的完成时间后延至交换前a_y的完成时间. 由于交换前a_y的完成时间必然不超过$E_y < E_x$，交换后a_x必然能够按时完成，其他任务显然也可以按时完成. 因此，对于任意相邻的两个任务a_x和a_y，其中a_x在前，$E_x > E_y$，则交换任务a_x和a_y后，所有任务依旧可以按时完成，可调度的任务数量不变. 可以从P开始，不断地交换最晚完成时间逆序的相邻任务，直至得到按照完成时间排序的调度S，此时S中按时完成的任务数量与P相同，S也是优化解，即存在优化解S包含任务p，而且S中的任务按照最晚完成时间排序.

（3）表述并证明问题的优化子结构。

设$A=\{a_1, a_2, \ldots, a_n\}$，$a_i=(t_i, T_i)$，$p=arg_i \min\limits_{1\le i\le n} t_i$. S是A的最优调度，S包含a_p，并且S中的任务按照最晚完成时间升序排列.复制$A/\{a_p\}$中的元素到A'，修改A'中的元素，对于$a_q \in A'$：如果$T_q > T_p$，则$T_q \leftarrow T_q - t_p$；如果$T_q \le T_p$，则$T_q \leftarrow min\{T_q, T_p - t_p\}$；如果$T_q < t_q$，则从$A'$中去掉$a_q$，即$A' \leftarrow A'/\{a_q\}$. 从$S$中去掉$a_p$，其余任务顺序不变得到$S'$，则$S'$是$A'$的最优调度，而且$S'$中任务按照最晚完成时间升序排列.

证明：易知S'中任务按照最晚完成时间升序排列. 如果S'不是A'的最优调度，令S''是A'的最优调度，而且S''中任务按照最晚完成时间升序排列. 于是$|S''| > |S'|$. 将S''中最晚完成时间大于$T_p - t_p$任务向后延时t_p，将得到三个部分：S''中最晚完成时间不大于$T_p - t_p$的任务序列；长度为t_p的空闲时间；S''中最晚完成时间大于$T_p - t_p$的任务序列. 将a_p置于空闲时间将得到一个A的调度S'''，而且$|S'''| = |S''| + 1 > |S'| + 1 = |S|$，与$S$是$A$的最优调度矛盾！因此$S'$是$A'$的最优调度，而且$S'$中任务按照最晚完成时间升序排列.

（4）写出算法的伪代码。

Optimal-plan(A)

Input: A=$\{a_i = (t_i,\ T_i) \mid i = 1,2,\dots,\ n\}$

Output: $P = \{a_{j_1}, a_{j_2}, \dots, a_{j_k}\}$

1. $S \leftarrow \emptyset$
2. $A' = \{1, \dots, n\}$
3. While $A' \neq \emptyset$
4. $\quad p \leftarrow arg_i \min_{i \in A'} t_i$
5. $\quad S \leftarrow S \cup \{p\}$
5. $\quad$ For $q \in A'$
6. $\quad\quad$ If $T_q \leq T_p - t_p$
7. $\quad\quad\quad T_q \leftarrow min\{T_q,\ T_p - t_p\}$
8. $\quad\quad$ Else if $T_q > T_p - t_p$
9. $\quad\quad\quad T_q \leftarrow T_q - t_p$
10. $\quad\quad$ If $T_q < t_q$
11. $\quad\quad\quad A' \leftarrow A'/\{q\}$
12. $P = \{a_j | j \in S\}$并将P中元素按照T_j升序排序
15. Return P

（5）分析算法的时间复杂度。

$O(n^2)$.