1.现有一台计算机，在某个时刻同时到达了 n 个任务。该计算机在同一时间只能处理一个任务，每个任务都必须被不间断地得到处理。该计算机处理这 n 个任务需要的时间分别为 $a_1,a_2,\ldots,a_n$。将第 i 个任务在调度策略中的结束时间记为 e_i。请设计一个贪心算法输出这 n 个任务的一个调度使得用户的平均等待时间 $1/n\sum e_i$ 达到最小。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）根据贪心选择性和优化子结构用伪代码写出算法；

（4）分析算法的时间复杂度

答：1>贪心选择性证明：

设任务集为 A，需要时间最短的任务为 ai，只需证明存在任务集 A 的一个最优调度即使平均等待时间最短的调度，第一个需要处理的任务是 ai。

设 S 是 A 的一个最优调度，S 的第一个调度任务为 ak，若 ak 为 ai，则命题成立。

如果 ak≠ai，令 B 为 A 的另一个调度，其调度顺序将 S 中的 ak 和 ai 对调。S 的调度时间 T(S)，B 的调度时间为 T(B)，S 中 ak 和 ai 之间的任务集为 C。

S 调度 ak 之后的任务和 B 调度中时间相同

B 调度中 C 等待时间相对于 S 调度来讲，由于由于 ai<=ak，调度时间减少。

ak 和 ai 的调度时间和在 B 中为 $ai+ai+\sum_C aj+ak$,在 S 中位 $ak+ai+\sum_C aj+ak$ ，由于由于 ai<=ak，B 相对 S 调度时间减少

故 T(S)>=T(B)，又因为 S 为最优调度，故 T(S)<=T(B)，所以 T(S)=T(B)，故 B 为最优调度。

2>优化子结构：

只需证明 S 为最优调度，则 S-ai 为 A-ai 的最优调度

若 S-ai 不是 A-ai 的最优调度，设 A-ai 的最优调度为 X，则 T(X)<T(S-ai),则 ai+（n-1）T(X)< ai+（n-1）T(S-ai)，这与 S 是最优调度矛盾。故 S-ai 是最优调度

3>伪代码：

```
MergeSort(A,0,n);
cout<<"最优调度为："<<endl;
for(int i=0;i<n;i++)
 cout<<A[i]<<'\t';
```

4>时间复杂度:O(nlgn)

2. 现有面值为 1 角、5 分、2 分、1 分的硬币，每种硬币的个数都是无限的。给出一个贪心算法，使得对任意给定的面值为 n（n>18）分的纸币能够将它兑换成币值相等的硬币且使用硬币个数最少。证明算法的正确性并分析其复杂度。

答：1.以分为单位，伪代码如下：

```
int count=0;
while(n>=10){count++;n-=10;}
while(n>=5){count++;n-=5;}
while(n>=2){count++;n-=2;}
while(n>=1){count++;n-=1;}
return count;
```

算法的正确性：

贪心选择性：

即证明当 n>=10 时，选择面值为 1 角的，设 S 为最优选择

若当 n>=10 时，S 选择面值为 1 角的，贪心选择性成立。

若 S 中当 n>=10 时，不选择面值为 1 角的，选择 X 当 n>=10 时，S 选择面值为 1 角的，则 X 中多出的 1 角在 S 中需要其它面值兑换，使数量增加，故 S 的数量 C(S)>=C(X)。

又因为 S 为最优选择，故 C(S)<=C(X)，故 C(S)=C(X)，所以 X 为最优选择。

优化子结构证明：

设选择 1 角的个数为 m，S 为最优选择，X 为 S 中选择 1 角的选择，即证 S-X 为 n-10m 在面值 5 分，2 分，1 分的选择。

若 S-X 不是最优选择 n-10m 在面值 5 分，2 分，1 分的选择，存在最优选择 B

则 C(B)<C(S-X)，加上原有 1 角的选择有 m+C(B)<m+C(S-X)=C(S)

与 S 是最优选择矛盾，故 S-X 为 n-10m 在面值 5 分，2 分，1 分的选择

所以算法是正确的。

时间复杂度为 O(n)

3.给定 k 个排好序的有序序列 $s_1, s_2, \ldots, s_k$,现在用 2 路归并排序算法对这些有序序列排序。假定用 2 路归并排序算法对长度分别为 m 和 n 的有序序列排序要用 $m+n-1$ 次比较操作。设计一个贪心算法合并 $s_1, s_2, \ldots, s_k$ 使得所需的比较操作次数最少。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；

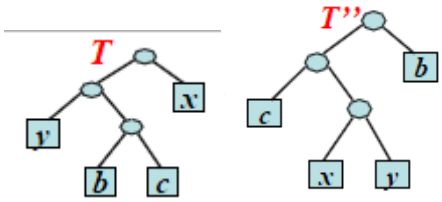
（3）根据贪心选择性和优化子结构用伪代码写出算法；

（4）分析算法的时间复杂度。

答：1>贪心选择性：

问题类似 Huffman 树，面值即为频率，合并的先后决定参与合并的次数，即为树的深度。

序列 s1,s2, •••••• sn 中的最小两个序列为 x,y，则存在最优合并树 T，使 x 和 y 具有最大长度。



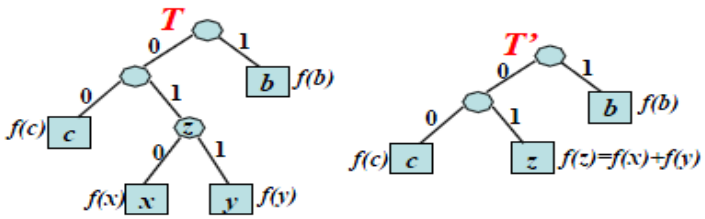
若 x 和 y 是具有最大深度的两个叶节点，则问题得证。

若 x 和 y 不是具有最大深度的两个叶节点，设 b 和 c 是具有最大深度的两页节点，设除 x,y,a,b 四节点子树外，其它的合并代价和 T''一样，设为 H，T 的合并次数为 Cost(T)=b+c+b+c+y+b+c+y+x=3(b+c)+2y+x，Cost(T'')= 3(x+y)+2c+b，因为 x<b,x<c,y<b,y<c，故 Cost(T'')< Cost(T)

又因为 T 为最优合并树，故 Cost(T'')>Cost(T)故 Cost(T'')=Cost(T)

即 T''为最优合并树。

2>优化子结构

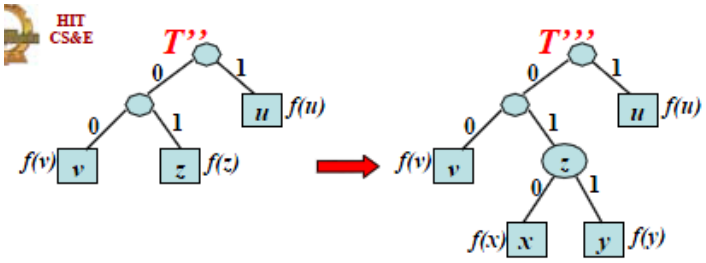


如上图所示只需证如果 T 为最优合并树，则 T’是最优合并树。

Cost(T)- Cost(T’)=z=x+y

即 Cost(T)= Cost(T’)+x+y

若 T’不是最优合并树，存在最优合并树 T’’,由 T’’可得到合并树 T’’’



易得 Cost(T’’’) = Cost(T’’) +x+y 且 Cost(T’) > Cost(T’’)

故 Cost(T’’’) >Cost(T)

与 T 是最优合并树矛盾，故 T’是最优合并树。

3>伪代码如下：

```
MergeSort(S,0,n);
for(int i=0;i<n-1;i++)
{
    z <- AllocateNode();
    left(z) <- x <- min(S);     //从S中删除x
    right(z) <- y <- min(S);    //从S中删除y
    z=x+y;
    Insert(S,z)
    return min(S);              //返回根节点
}
```

4>时间复杂度为 O(nlgn)

4.给定两个大小为 n 的正整数集合 A 和 B。对于 A 到 B 的一个一一映射 f，不妨设 $f(a_i)=b_i$（i=1,…,n），则 f 的代价为 $\sum_{i=1}^{n} a_i^{b_i}$ 。试设计一个贪心算法，找出从 A 到 B 的代价最大的一一映射。你的答案应包括：

（1）用伪代码写出算法；

（2）证明算法的正确性；

答：1>伪代码如下：

```
MergeSort(A,0,n);
MergeSort(B,0,n);
sum=0;
for(int i=n-1;i>=0;i++)
{
    sum+=pow(A[i],B[i]);
}
```

2>贪心选择性：

设 x 是 A 中最大的元素，y 是 B 中最大的元素，只需要证明存在最优选择包含 x^y

设 M 是最优选择，如果 M 包含 x^y，则结论得证。

若 M 不包含 x^y，包含 x^a，b^y，构造选择 N，包含 x^y，b^a，其余同 M 相同且其余部分和为 H，则有 a<=y,b<=x

x^y+b^a-(x^a+b^y)=x^a(x^(y-a)-1)- b^a(b^(y-a)-1)>=0

故 N 的总代价 Cost(N)>=Cost(M)

又因为 M 为最优选择，故 Cost(N)<=Cost(M)，故 Cost(N)=Cost(M)，故 N 为最优选择

结论得证。

优化子结构：

设 x 是 A 中最大的元素，y 是 B 中最大的元素，M 是最优选择，M-x^y 是 A-x，B-y 的最优选择。

如果 M-x^y 不是 A-x，B-y 的最优选择，其存在最优选择 N，则 Cost(N)>Cost(M-x^y)

同时加上 x^y，得 Cost(N)+ x^y>Cost(M)，这与 M 是最优选择矛盾

故 M-x^y 是 A-x，B-y 的最优选择。

故算法是正确的。

5. 一个 DNA 序列 X 是字符集{G, T, A,C}上的串，其上有大量信息冗余。设 x 是 X 的子串，x 及其冗余形式在 X 内在出现的起、止位置构成了一系列等长区间$[p_1,q_1],\ldots,[p_m,q_m]$。试设计一个贪心算法找出$[p_1,q_1],\ldots,[p_m,q_m]$中互不相交的区间的最大个数，即确定 x 的独立冗余度。

（1）用伪代码写出算法；

（2）证明算法的正确性；

答：1>伪代码如下：

```
MergeSort(q,0,n);      //根据qi的大小对序列进行排序
A <- {1}
for(j=1,i=2;i<n;i++)
{
    if(pi<qj)
    {
        A <- A U {i};
        j <- i;
    }
}
```

2>贪心选择性：

即证存在最优选择包含 q 中最小的元素。

设 M 为最优选择，qi 为 q 最小的元素，如果 M 包含 qi，问题得证。

如果 M 不包含 qi，其第一个元素为 qj，第二个元素为 qk，设 N 为另一组合，N 的第一个元素为 qi，剩下元素同 M 相同，则有如下关系：

qi<qj 且 qj<pk，所以 qi<pk，所以选择 M 的互不相交个数 Count(S)=Count(X)

所以 N 为序列的最优选择，问题得证。

优化子结构：

设[pi,qi]为最优选择 M 的第一个元素，则 M - [pi,qi]为序列 X’={j∈X|pj>qi}的最优选择。

若 M - [pi,qi]不是序列 X’={j∈X|pj>qi}的最优选择，设最优选择为 N，

则 Count(N)>Count(M - [pi,qi])，由于序列 X’中 pj>qi，故 M - [pi,qi]和 N 中均可以加上[pi,qi]

所以 1+Count(N)> Count(M - [pi,qi])+1=Count(M)

与 M 是最优选择矛盾，故 M - [pi,qi]为序列 X’={j∈X|pj>qi}的最优选择。

6.从哈尔滨到上海的高速公路上有若干个加油站。如果汽车从一个加油站出发时油箱是满的，则汽车可以顺利达到下一个加油站。试设计一个汽车加油方案使得汽车在整个行驶路程中加油的次数最少，证明所给方案的正确性。

答：1>解决方案：

根据实际问题，加油站之间距离不等，若要使加油次数最少，只需做到如果车内的油能够到达下一站点就不加油，即做贪心选择。

2>正确性证明：

站点分别记为 1,2, •••••• n，站点集合记为 A，我们只需证明问题贪心算法的正确性，即满足贪心选择和优化子结构即可。

贪心选择：

设从初始位置最远可以前行到 i 站点，则存在最优选择 M，M 前行到 i 站点首次加油。

M 前行到 i 站点首次加油，问题得证。

若 M 不前行至 i 站点进行加油，在加油站 j 进行首次加油，易得 j<i，存在加油组合 N，使 N 中除第一个加油站点外，剩余的加油站点均一致。

因为就 j<i，故从 j 站点可以到达的下一加油站 k 从 i 站点也可以同样到达。

故加油次数 Count(M)=Count(N)，故 N 也为最优选择，问题得证。

最优子结构：

设最优选择 M 包含从初始位置前行到 i 站点首次加油，只需证 M-i 为 A - { 1,2，•••，i } 的最优选择。

如果 M-i 不是 A - { 1,2，•••，i }的最优选择，其最优选择为 N,则 Count(N)<Count(M-i)

在分别加上在 i 站点的加油次数得：1+Count(N)<1+Count(M-i)= Count(M)

这与 M 是最优选择矛盾，故 M-i 为 A - { 1,2，•••，i }的最优选择。

所以方案是正确的。