1． 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 ti ,1 ≤ i ≤ n 。

应该如何安排 n 个顾客的服务次序才能使总的等待时间达到最小？总的等待时间是各顾客等待服务的时间的总和。试给出你的做法的理由（证明）。

解：设调度f为: i1,i2,…,in，作业ik需等待，即作业1等待0，作业2等待i1，作业3等待i1+i2….，总的等待时间为：T(f)=。

使用贪心策略：服务时间短的作业先安排。算法描述略。

正确性：交换论证。不妨假设t1≤t2≤t2≤…≤tn,算法的调度f结果为1,2,..,n。设它不是最优的，存在最优调度f\*，设其最早第k项作业ik与 f不同,即f\*:1,2,..k-1,ik,ik+1…in，必有tik>=tk。 今将f\*中作业K与作业ik置换，得到调度f\*\*：1,2…k,ik+1,…ik…in。其中ik位置为j，则j>k, tik>=tk。则：

T(f\*)-T(f\*\*)=(n-k)tik+(n-j)tk-(n-k)tk-(n-j)tik=(j-k)tik+(k-j)tk=(j-k)( tik-tk)>=0

说明f\*\*也是最优调度，但它与f不同的次序项后移了一位。重复此过程最多n步，可得f最优。

1. 字符 a ~ h 出现的频率分布恰好是前 8 个 Fibonacci 数，它们的 Huffman 编码是什么？将结果推广到 n 个字符的频率分布恰好是前n个 Fibonacci数的情形。

Fibonacci 数的定义为： F0 = 1, F1 = 1, Fn = F（n−2）+ F（n−1） if n > 1

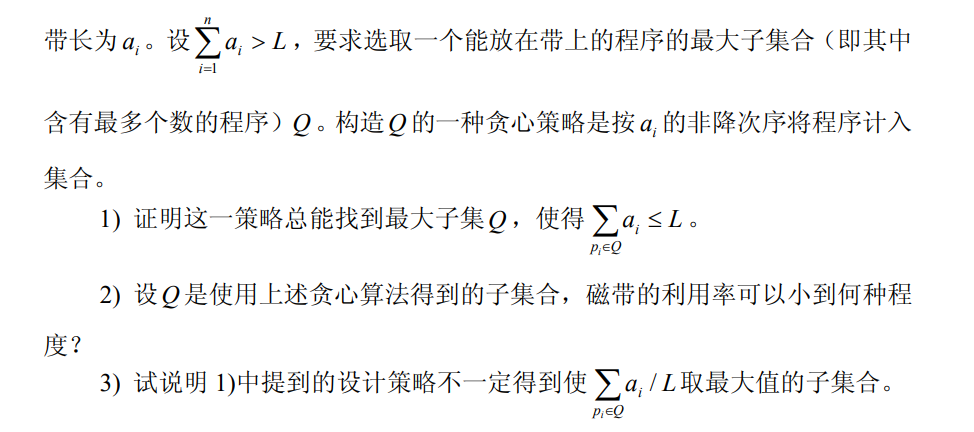
1. h的频率为：1,1,2,3,5,8,13,21,按huffman规则，编码为0000000,0000001,000001,00001,0001,001,01,1

设前k个字符符合：f0+f1+..+fk-1<fk+1成立(1<k<8已成立), 则

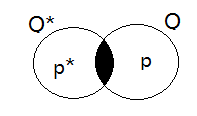
F0+f1+…+fk-1+fk<fk+1+fk=fk+2,定义字符zk= f0+f1+..+fk-1，则对任意k，zk<fk+1

经huffman编码k-1步后，得zk,fk,fk+1,….,据上，zk<fk+1,而显然fk<fk+1

Zk,fk是最小频率的两个数，合并，依次类推，每次都是新字符与下一个Fibonacci数合并，所以，f0-fn的编码为000…0,000…1,..001,01,1。其中fk的编码为000..01，共n-k个0（k>0），f0:n个0。

设 p1 , p2 ,, pn 是准备存放到长为 L 的磁带上的 n 个程序，程序 pi 需要的 。

解：



1. 反正法。

设算法得到的程序集合为Q不是最优的，Q\*是一个最优集合，则Q与Q\*无包含关系。Q不能包含Q\* 是显然的，如果Q\*包含Q ，那么对p∈Q\*\Q，如果p不小于Q的成员，算法将会在接下来的判断中将其选入(不超L)，否则，它先被检查，早应该是Q的成员。

（再画一个图，Q\*大圆 Q小圆包含情况 这么理解——找到Q和Q\*的重合情况）

取Q\*使得|Q∩Q\*|最大。令p是Q\Q\*中最小的，则Q中比p更小的程序一定在Q\*中(阴影部分)。则任给p\*∈Q\*\Q，必有p\*>=P（>阴影中的一部分）。否则，若p\*<P，ｐ\*在P前被算法检查，

（找到p和p\*的大小关系）

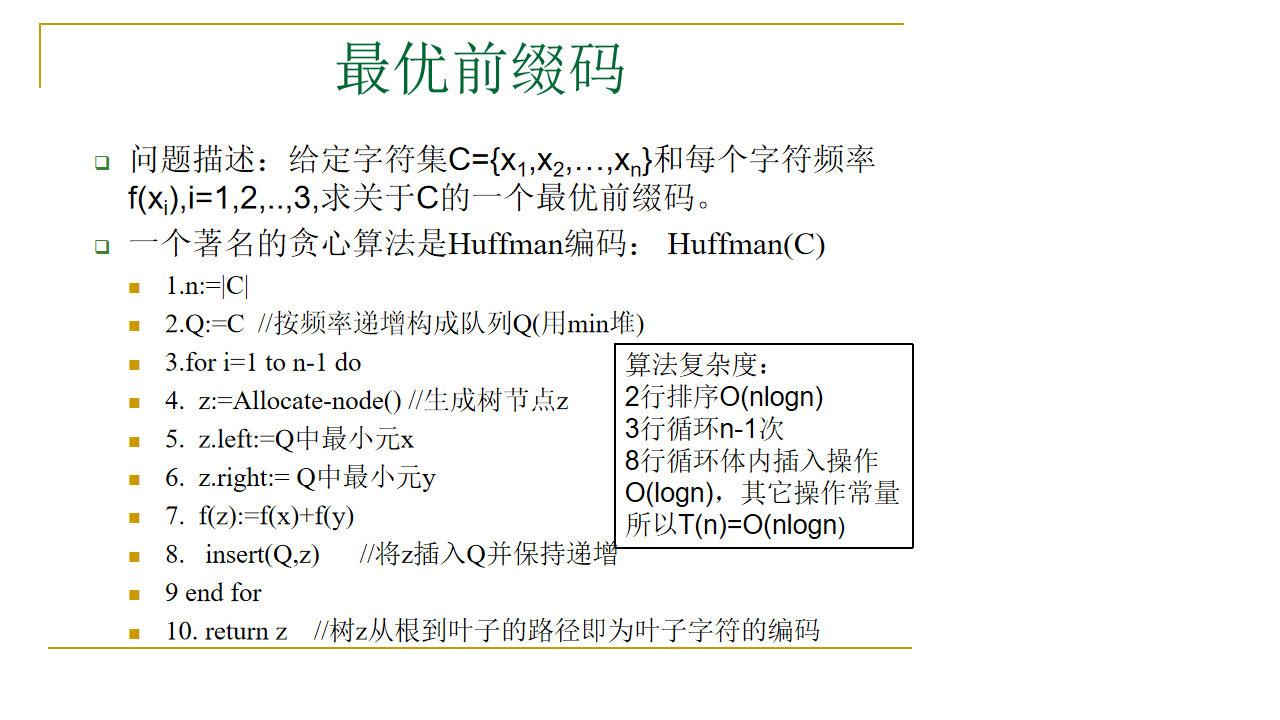
此时磁带装入的程序均在Q∩Q\*即Q\* 中，P\*不被选入，说明此时磁带已装不下p\* ，但P\*在Q\*里，说明P\*在Q∩Q\*之外可以装入，矛盾。

将Q\*中的P\*换成P，显然可以装入，得到Q\*\*，也是最优的，但|Q\*\*∩Q|>| Q\*∩Q|，与Q\*的选取矛盾。得证。

(2)磁带利用率为该是集合Q中所有程序的长度之和/L，可以为0，如果一个也装入不了。

(3)反例，l=8,n=4,a1=1,a2=2,a3=3,a4=4,算法装入p1,p2,p3,带长浪费2，而装入p1,p3,p4，带长浪费0。

4． 写出 Huffman 编码的伪代码，并编程实现。



设有一条边远山区的道路AB，沿着道路AB分布着n所房子。这些房子到A的距离分别是d1,d2,…,dn(d1<d2<…<dn)。为了给所有房子的用户提供移动电话服务，需要在这条道路上设置一些基站。为了保证通讯质量，每所房子应该位于距离某个基站的4Km范围内。设计一个算法找基站的位置，并且使得基站的总数最少。证明算法的正确性。

解：使用贪心法：令a1,a2,…表示基站的位置。

贪心策略：首先令a1=d1+4，对d2,d3,…dn依次检查，找到下一个不能被该基站覆盖的房子。如果dk<=a1+4但dk+1>a1+4,那么第k+1个房子不能被基站覆盖，于是取a2=dk+1+4作为下一个基站的位置，照此下去，直到检查完dn为止。

算法伪码：Location()

输入：距离d1,d2,…,dn的数组d[1..n]:满足d[1]<d[2]<…<d[n]

输出：基站位置的数组a[..]

a[1]:=d[1]+4

K:=1

for j:=2 to n

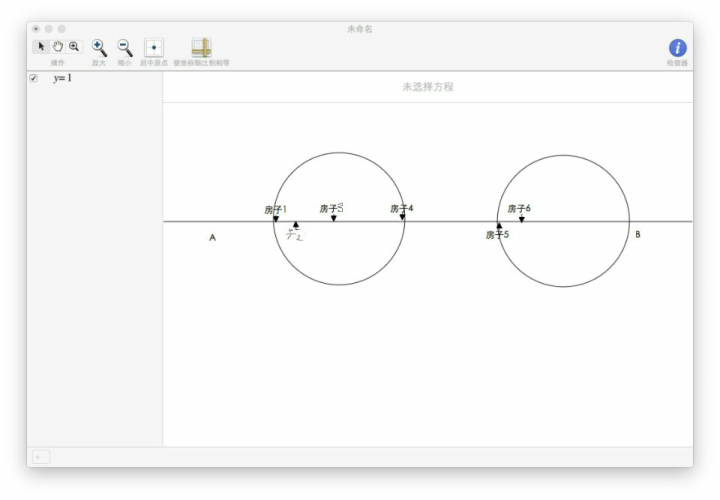
If d[j]>a[k]+4

k:=k+1

a[k]:=d[j]+4

Return a

大概想法是 从第一个房子开始 + 4000米 ，然后跳过这个区域，再找下一个房子，从这个房子开始 + 4000米



算法的正确性证明使用归纳法：对任何正正整数k，存在最优解包含算法前k步选择的基站位置。

证明：k=1，存在最优解包含a[1]。若不然，有最优解OPT，其第一个基站位置是b[1]，b[1]≠a[1]。那么d1-4≤b[1]<d1+4=a[1]。B[1]覆盖的是距离在[d1,b[1]+4]之间的房子。A[1]覆盖的是距离[d1,a[1]+4]的房子，因为b[1]<a[1],b[1]覆盖的房子都在a[1]覆盖的区域内，用a[1]替换b[1]，得到的仍旧是最优解。

假设对于k存在最优解A包含算法前k步的选择的基站，即A={a[1],a[2]…a[k]}∪B，其中a[1],a[2]…a[k]覆盖了距离d1,d2,…,dj的房子，那么B是关于L={dj+1,dj+2,…,dn}的最优解。否则，存在关于L的最优解B\*，那么用B\*替换B得得到A\*，且|A\*|<|A|，与A是最优解矛盾。根据归纳假设，L有一个最优解B/={a[k+1],…,}，|B/|=|B|。于是，A/={a[1],a[2],…,a[k]}∪B/= {a[1],a[2],…,a[k+1]…}也是最优解，且|A|=|A/|，从而证明了结论对k+1也真。证毕。算法复杂度 T(n)=O(n)。

2. 有n个的进程，Test每次测试时间很短，可以忽略不计，即，如果Test在时刻t测试，那么它将对满足s[i]<=t<=d[i]的进程p1,p2,..,pn，进程pi的开始时间为s[i]，截止时间为d[i]。可以通过检测程序Test来测试正在运行所有进程同时取得测试数据。问：如何安排测试时刻，使得对每个进程至少测试一次，且Test测试的次数达到最少？设计算法并证明正确性，分析算法复杂度。

使用贪心法。

贪心策略：把进程按截止时间排序。取第一个进程的截止时间作为第一个测试点，然后顺序检查后续能够被这个测试点检测的进程(这些进程的开始时间小于测试点)，直到找到下一个不能够被这个测试点测试到的进程为止。取这个进程的截止时间作为下一个测试点，……,知道检查完所有的进程为止。算法描述：

Test()

输入：s[1..n]，d[1..n]；输出：t[]，顺序选定的测试点构成的数组

将进程按d[i]递增顺序排序，使d[1]≤d[2]≤…≤d[n]

struct Process {

int start, end;} p[N];

bool cmp(Process a, Process b){

return a.end < b.end;}

int main(){

int k, n;

while(k--) {

sort(p, p + n, cmp);

int cnt = 0;

for(int i = 0, j; i < n; ) {

for(j = i; p[j].start <= p[i].end && j < n; j++);

i = j;

cnt++;

}

printf("%d\n", cnt);

}

return 0;}

用归纳法证明其正确性:

任给k，存在最优解包含算法前k步选择的结果。

证明：k=1，设s={t[i1],t[i2]，…}是最优解，则一定t[i1]<t[1],否则t[i1]不能测试d[1]的进程。设pu是在时刻t[i1]被检测到的任意进程，则s[u]≤t[i1]≤d[u]，从而s[u] ≤t[i1]<t[1]=d[1] ≤d[u]，因此，pu也可以在t[1]时刻被测试，在s中将t[i1]替换为t[1]也是最优解。

假设结论对k成立，则存在最优解T={t[1],t[2],…,t[k]}∪T/。设算法前k步选择的测试点不能测到的进程构成集合Q⊆P，P是全部进程集合，那么T/是问题Q的最优解。根据归纳假设，存在Q的最优解T\*包含测试点t[k+1]，即T\*={t[k+1]}∪T//，因此，{t[1],t[2],…,t[k]}∪T\*={t[1],t[2],…,t[k],t[k+1]} ∪T//也是原问题的最优解，得证。

算法最坏时间复杂度：排序O(nlogn)，检查O(n)，所以T(n)=O(nlogn)。

1. 设有作业集合J={1,2,…,n}，每项作业的加工时间都是1，所有作业的截止时间是D。若作业i在D之后完成，则称为被延误的作业，需赔偿罚款m(i)(i=1,2,..n)，这里D和m(i)都是正整数，且n项m(i)彼此不等。设计一个算法求出使总罚款最小的作业调度算法，证明算法的正确性并分析时间复杂度。

使用贪心法。贪心规则：优先安排前Ｄ个罚款最多的作业。

正确性证明：交换论证。设算法选择的作业调度ｆ的安排次序是<i1,i2,…,in>，那么罚款为F(f)=，任给k<=D<j，m(ik)>=m(ij)。显然最优调度没有空闲时间，不妨设作业是连续安排的。每项作业的加工时间都是1，在D之前可完成D项作业，在D之后安排的n-D项作业iD+1,iD+2,…,in是被罚款的作业。

设算法得到的安排不是最优的，则存在一个最优调度f\*，它的前D个作业包含了至少1个作业ij，j>D，从而至少有一个作业ik，k<=D被安排在了D之后。交换ik,ij得到新的调度f\*\*，则F(f\*)-F(f\*\*)=m(ik)-m(ij)≥0，说明f\*\*也是最优调度，但f\*\*与f的前D个相同作业多了1个，依次进行，可得最优作业与f相同，得证。

算法描述：

算法A:按m[i]非升排序，依次选择作业即可。但T(n)=O(nlogn).

——用算法A就可以了

算法B:m\*:=select(m[],n-D)

Partition(m[],A,B,m\*)

{ i1,i2,…,iD }：=B，{iD+1,iD+2,…,in}:=A+{m\*}

算法复杂度T(n)=O(n)。

1. 举出反例证明：本章开始例1贪心规则找零钱算法(目标：零币数量最少；规则：尽量先找币值大的)，在零钱种类不合适时，贪心算法结果不正确。

零钱系统币值为25,10,1元，找30元。

按贪心算法30=25+1X5共6枚。但30=10\*3只要3枚