**1.试证明：对于每一个无向连通图G，用prim算法能够生成一棵最小成本的生成树。**

反证法：假设prim生成的不是最小生成树

1).设prim生成的树为G0

2).假设存在Gmin使得cost(Gmin)<cost(G0) 则在Gmin中存在<u,v>不属于G0

3).将<u,v>加入G0中可得一个环，且<u,v>不是该环的最长边(这是因为<u,v>∈Gmin)

4).这与prim每次生成最短边矛盾

5).故假设不成立，命题得证.

**2. 试证明：对于每一个无向连通图G，用Kruskal算法能够生成一棵最小成本的生成树。**

证明：设G是任一无向连通图，又设T是用Kruskal算法生成的G的生成树，再设T’是G的最小成本生成树。

现要证明T和T’具有相同的成本。

设E(T)和E(T’)分别是T和T’中的边集。

若n是G中的结点数，则T和T’都有n-1条边。

若E(T)=E(T’)，则T显然就是最小成本生成树，问题得证。

若E(T)≠E(T’)，则可假设e是一条使得e∈E(T)但不属于E(T’)的最小成本的边。显然，这样的e必定存在。

把e加到T’中后，必然产生一个唯一的环。

假设e, e1, e2, …, ek是这个唯一的环。这些边ei(1≤i≤k)至少有一条不在E(T)中，如若不然，则T也将包含环e, e1, e2, …, ek，这与T是树矛盾。

设ej是e, e1, e2, …, ek环中使得ej不属于E(T)的一条边，则有c(ej)＞c(e)，其中，c(e)是边e成本函数。

因为若ej比e有更小的成本，则Kruskal算法就会在e之前考虑ej，并把ej加到T中。

也就是说，E(T)中所有比e成本小的边都在E(T’)中，并且和ej一起构不成一个环。

因此，有c(ej)＞c(e)。

现在，重新考虑具有边集E(T’) ∪ {e}的图，去掉环e, e1, e2, …, ek上的任意一条边，则可得到一棵树T’’。

特别是，如果删去边ej，则所产生的树T’’的成本将不比T’的成本大，因为c(ej)≥c(e)。

因此，可得T’’也是一棵最小成本树。

若否，则反复使用上述的转换，树T’就可以转换成T而在成本上没有任何增加。

故T是一棵最小成本生成树。

证毕。

**3.背包问题 物体重量w1.w2.w3…….证明：能用贪婪法(Greedy\_Knapsack)找到一个最优解**

证明：设x=(x1, …, xn)是Greedy\_Knapsack所生成的解。如果所有的xi等于1，显然这个解就是最优解(效益值为Σpi)。

于是，设j是使xj≠1的最小下标。由算法可知，对于l≤i＜j，xi=1；对于j＜i≤n，xi=0；对于j，0≤xj＜1。

如果x不是一个最优解，则必定存在一个可行解y=(y1, …, yn)，使得Σpiyi＞Σpixi。

不失一般性，可以假定Σwiyi=M。且设k是使得yk≠xk的最小下标。显然，这样的k必定存在。

由上面的假设，可以推得yk＜xk。这个结论可从以下三个方面，即k＜j，k=j或k＞j分别加以证明。

① 若k＜j，则xk=1。因yk≠xk，从而yk＜xk；

② 若k=j，则由于Σwixi=M，且对于l≤i＜j，有xi=yi=1，而对于j＜i≤n，有 xi=0。

若yk＞xk，显然有Σwiyi＞M，与y是可行解矛盾；

若yk=xk，与假设yk≠xk矛盾，故yk＜xk。

③ 若k＞j，则Σwiyi＞M，这是不可能的。

现在，假定把yk增加到xk，那么必须从(yk+1, …, yn)中减去同样多的量，使得所用的总容量仍是M。这导致一个新解z=(z1, …, zn)，其中zi=xi，1≤i≤k，并且有

Σwi(yi-zi)=wk(zk-yk)。

n

i=k+1

因此，对于z有

Σpizi = Σpiyi + (zk-yk)wkpk/wk -Σ(yi-zi)wipi/wi ≥ Σ piyi + [(zk-yk)wk - Σ(yi-zi)wi]\*pi/wk = Σpiyi

n

i=1

n

i=1

n

i=1

n

i=k+1

n

i=1

n

i=k+1

n

i=1

n

i=1

若Σpizi＞Σpiyi，则y不可能是最优解；若相等，且z=x，则x就是最优解；

若z≠x，则需重复上述讨论，或者证明y不是最优解，或者把y转换成x，从而证明了x也是最优解。

**4.在折半查找过程的判定树上，定义根结点到每个内部结点(查找成功的结点)的路径长度之和为内部路径长度，记为I；定义根结点到每个外部结点(查找不成功的结点)的路径长度之和外部路径长度，记为E。**  (15分)

**试证明：1.具有n个内部结点的这样的判定树，满足n = ( E - I ) / 2 。**

**2.该判定树的高度为[log2n ] + 1，其中，[log2n ]表示取log2n 的整数。**

答：1．设判定树的高度为h+1(含外部结点，且根结点的级别为1)。

根据描述折半查找过程的判定树的定义可知，判定树上不存在度为1的结点（即判定树是一个严格二叉树），且内部结点的最高层次为h；在h-1层以上是由内部结点构成的满二叉树。

不失一般性，假设第h层上有m个内部结点。则其余2h-1-m个结点是外部结点，且第h+1层上全部是外部结点。

则内部结点数n和m之间的关系为：n = 2h-1 - 1 + m ，即m = n - 2h-1 + 1

由于第i层的内部结点数为2i-1，根到第i层上的内部结点的路径程度为i-1(1≤i≤h-1)。

于是，根据外部路径长度和内部路径长度的定义可知：

I= ∑ (i-1)\*2i-1 + m\*(h-1) ，E = (2h-1-m)\*(h-1) + 2\*m\*h

h-1

i=1

所以，

E - I = (h-1)\*2h-1 -m\*(h-1) + 2\*m\*h - ∑ (i-1)\*2i-1 - m\*(h-1)

h-1

i=1

h-1

i=1

h-1

i=1

= (h-1)\*2h-1 -2m\*(h-1) + 2\*m\*h - ∑ (i-1)\*2i-1 = (h-1)\*2h-1 + 2m - ∑ (i-1)\*2i-1

h-1

i=1

= (h-1)\*2h-1 + 2(n - 2h-1 + 1) - ∑ (i-1)\*2i-1

h-1

i=1

令 S= ∑ (i-1)\*2i-1= 1\*2 + 2\*22 + 3\*23 +…+ (h-3)\*2h-3 +(h-2)\*2h-2

则，2S = 1\*22 + 2\*23 + 3\*24 +…+ (h-3)\*2h-2 +(h-2)\*2h-1

S - 2S = 2 + 22 + 23 + 24 +…+ 2h-2 - (h-2)\*2h-1

= 2\*(2h-2-1) - (h-2)\*2h-1

= 2h-1-2 - (h-2)\*2h-1

= - (h-3)\*2h-1 – 2

即，S = (h-3)\*2h-1 + 2

所以，有 E-I = (h-1)\*2h-1 + 2(n - 2h-1 + 1) - (h-3)\*2h-1 – 2 = 2n

即，n = ( E - I ) / 2 成立。

2．设具有n个结点的判定树的高度为h(含外部结点)。

对于任意的n＞1，则根判定树的构造规则可知，

判定树的第一层上有1个结点，即20=1个内部结点，

判定树的第二层上有2个结点，即21=1个内部结点，

一般地，若n≥2k，则第k层上有2k-1个内部结点。

显而易见，在高度为h的判定树的第h-2层上有2h-3个内部结点，在第h-1层上有2h-2个结点，其中部分是内部结点，部分是外部结点，且第h层的结点全部为内部结点。

于是，有

h-2

i=1

∑ 2i-1 = 2h-1-1 ＞ n

即有，2h-1≥ n

由于n＞1，h-1≥log2n

由于h是整数，所以有

h-1 = [log2n]，[log2n]表示取不大于log2n的整数值。

也即有h = [log2n] + 1。

**5. 试证明，当且仅当 *lim* f(n)/g(n)=0时，f(n)=o(g(n))。**

n→∞

证明：

首先证明，若f(n)=o(g(n))，则*lim* f(n)/g(n)=0。 5分

n→∞

根据小写o的定义，当f(n)=o(g(n))成立时，有f(n)=Ο(g(n))且f(n)≠Ω(g(n))。

即存在正常数c和n0，使得对于所有的n≥n0，有f(n)≤c\*g(n)。

考虑到f(n)的非负性，对于所有n≥n0，应有g(n)≥0，且若g(n)=0，则有f(n)≡0。于是，根据大写Ο的定义及大写Ο的比率定理可知，

*lim* f(n)/g(n)存在且等于0。

n→∞

所以，不失一般性，可以假设当n≥n0时有g(n)＞0。根据大写Ο的比率定理可知，f(n)/g(n)的极限存在且*lim* f(n)/g(n)≤c。

n→∞

令*lim* f(n)/g(n) = a，其中0≤a≤c。若a=0，则问题得证。

n→∞

假设a＞0，则根据极限的有关知识，可导出矛盾。因为有 *lim* g(n)/f(n) = *lim* (1/(f(n)/g(n)) = 1/a=c’。 c’是一个正的常数。

n→∞

n→∞

根据极限定义有，对于给定的任意小的常数ε＞0，存在n1，使得对于一切n≥n1，有| g(n)/f(n)-a|≤ε，也即有 g(n)/f(n)≤ε+a=b，从而有f(n)≥b\*g(n)。

于是，取m=max(n0,n1),则存在这样的正常数c，b，使得对于一切的n≥m，有f(n)≤c\*g(n)和f(n)≥b\*g(n)都成立，与小写o的定义矛盾。

所以，*lim* f(n)/g(n) = 0，

n→∞

再证，若*lim* f(n)/g(n) = 0，则f(n)=o(g(n))。 5分

n→∞

根据极限定义，对于任意小给定的ε＞0，存在n0，使得对于一切的n≥n0，且考虑到f(n)的非负性，所以有

| f(n)/g(n)| = f(n)/g(n)≤ε，即f(n)≤ε\*g(n)，取c=ε，满足大写Ο的定义。

即f(n)=Ο(g(n))

而*lim* g(n)/f(n) = *lim* 1/(f(n)/g(n))→∞，所以根据大写Ο比率定理，有

n→∞

n→∞

f(n)≠Ω(g(n))，即f(n)=o(g(n))。证毕。

**6.已知f(n)=О(log2n)，求证：f(n)=О(logkn)，k＞1。**

证明：

因为log2n= logkn / logk2，其中k＞1， 所以有 logk2＞0 2分

因为 f(n)＝О(log2n)，所以 存在正常数c和n0，使得对于一切n大于n0 有

f(n)≤c\*log2n＝(c/logk2) \* logkn

令c1＝c/logk2，则对于一切n大于n0 有 2分

有f(n)≤c1\*log2n

所以，f(n)=О(logkn) 2分

**7.对于函数f(n)和g(n)，若*lim* g(n)/f(n)存在，则f(n)＝Ω(g(n))当且仅当存在确定的常数c，有*lim* g(n)/f(n)≤c。**

证明：先证充分性：若f(n)=Ω(g(n))，则*lim* g(n)/f(n)≤c。 5分

n→∞

根据Ω的定义，当f(n)=Ω(g(n))成立时，即存在正常数a和n0，使得对于所有的n≥n0，有f(n)≥a\*g(n)。

考虑到f(n)的非负性，以及a为正常数，所以有：

g(n)/f(n)≤1/a，

因为*lim* g(n)/f(n)存在，可令c=1/a，所以有因为*lim* g(n)/f(n)≤c。 充分性成立。

n→∞

n→∞

再证明必要性（有两种方法：一是根据定义推理；二是利用反证法）。

因为*lim* g(n)/f(n)存在，可设为a，且有*lim* g(n)/f(n)≤c。

n→∞

n→∞

于是，根据极限定义有，对于给定的任意小的常数ε＞0，存在n1，使得对于一切n≥n1时，有

|g(n)/f(n)-a|≤ε，也即有-ε≤g(n)/f(n)-a≤ε，a-ε≤g(n)/f(n)≤ε+a，所以有

g(n)/f(n)≤ε+a，考虑到f(n)的非负性，所以有f(n)≥c\*g(n)，其中c=1/(ε+a)>0为常数。再取n0=n1,

也就是说，存在正常数c和n0，使得对于一切的n≥n0，有f(n)≥c\*g(n)，从而有f(n)=Ω(g(n))。

证毕。 5分。

**8.证明n!=О(nn)**  (5分)

证明：对于任意的n>0,有

n!=1×2×…×n ≤ n×n×…×n=nn

令c=1，即有

n! ≤nn

所以，根据大○定义，有

n!=О(nn)

证毕。 5分。

**9.某算法的计算时间可用下面的递推关系式描述。试采用迭代方式求解该关系式，并用大写О表示解(要求给出详细的推导过程)。(6分)**

a n=1，a为常数

T(n)=

2T(n/2) + c\*n n>1，c为常数

首先考虑n是2的整次幂，即对于某个正整数k，有n=2k。

T(n) = 2T(n/2) + c\*n = 2(2T(n/4) + c\*n/2) + c\*n = 4T(n/4) + 2c\*n = 4(2T(n/8) + c\*n/4) + 2c\*n = …

= 2kT(1) + k\*c\*n = a\*n + c\*n\*log2n

所以，T(n)= О(nlogn)。 3分

当n不是2的幂时，如何计算呢？考虑到T(n)是n的不减函数，所以有

T(2k)≤T(2k+1)。当2k≤n≤2k+1时，T(n)≤T(2k+1)，所以仍有T(n)=О(nlogn)。 2分

**10、某算法的计算时间可用下面的递推关系式描述。试采用迭代的方式求解该关系式，并用大写О表示解(要求给出详细的推导过程)。(6分)**

b n=1，b为常数

T(n)=

3T(n/2) + c\*n n>1，c为常数

（提示：xlogby = ylogbx）

首先考虑n是2的整次幂，即对于某个正整数k，有n=2k。

T(n) = 3T(n/2) + c\*n = 3(3T(n/4) + c\*n/2) + c\*n = 32T(n/22) + 3 \*c\*n/2 + c\*n

= 33T(n/23) + (3/2)2 \*c\*n + 3/2 \*c\*n + c\*n = …

= 3kT(1) +((3/2)k-1 + (3/2)k-2 + … + (3/2)2 + 3/2 + 1)c\*n = b\*3k + 2\*c(3k-2k)

≤ (b+c)\*3k =О(3k) （4分）

因为，k=log2n，而xlogby = ylogbx

所以，T(n)= О(nlog23)。

**11.在已知根R(i,j)（0≤i＜j≤n）的情况下写一个构造最优二分检索树T的算法。**

**证明这样的树可以在О(n)时间内构造出来。**

证明：

因为R(i,j)（0≤i＜j≤n）中记录了构造最优二分检索树Tij时满足最小成本的k值。

根据最优二分检索树的定义知道，给定的n个标识符是单调有序的（无重复）。

若Tij的根为ak，则aK的左子树上的标识符是{ai,ai+1,…,ak-1}，而则aK的右子树上的标识符是{ak+1,ak+2,…,aj},且当i=j时，Tij是外部结点。

根据上述思路，可依据如下算法构造一棵最优二分检索树。

采用二叉链表作为存储结构，结点的定义如下：

**主程序如下: 2分**

**main OBST0(OBSTree &T，int n) {**

**//数组A[]和R[][]是全局变量**

**if(n<1) return error；**

**OBSTCreate(T,0,n)；**

**}**

**时间复杂度证明: 2分**

**T(n) = T(k) + T(n-k) + c c为常数 0≤k≤n**

**= …**

**= (n+1)\*T(0) + n\*c**

**因为T(0)=0**

**所以T(n)= c\*n**

**即，T(n)=** О**(n)**

**证毕**

**因为T(0)=0**

**所以T(n)= c\*n**

**即，T(n)=** О**(n)**

**证毕**

typedef struct OBSTNode {

char data; //标识符类型，可以是字符串

struct OBSTNode \*lchild, \*rchild;

} OBSTNode,\* OBSTree

Void OBSTCreate(OBSTree P, int i,int j) {

int k； OBSTree S；

if (i〈〉j) {

k=r[i][j]； //数组R[][]是全局变量

new(S)； //产生一个新的结点,让S指向该结点

S->data=a[k]； //数据A[]是全局变量

If(!P) {P=S}； //产生根结点

Else {

if(S->data < P->data) { //新产生的结点是P结点的左孩子

P->lchild=S； P = S；}

else { //新产生的结点是P结点的右孩子

P->rchild=S； P = S；}

} ；

OBSTCreate(P,i,k-1)；

OBSTCreate(P,k+1,j)；

}

}// OBSTCreate 6分