第一章　作业

1. 证明下列O 、Ω 和Θ 的性质

1）　**f=O (g)当且仅当 g=Ω (f)**

证明：充分性。若 f=O (g)，则必然存在常数 $c_1>0$ 和 n_0，使得$\forall n\geq n_0$，有 $f\leq c_1*g(n)$。由于 $c_1\neq 0$，故 $g(n)\geq 1/c_1*f(n)$，故 g=Ω (f)。

必要性。同理，若 g=Ω (f)，则必然存在 $c_2>0$ 和 n_0，使得$\forall n\geq n_0$，有 $g(n)\geq c_2*f(n)$.由于 $c_2\neq 0$，故 $f(n)\leq 1/c_2*f(n)$，故 f=O (g)。

2）　**若 f=Θ (g)则 g=Θ (f)**

证明：若 f=Θ (g)，则必然存在常数 $c_1>0$，$c_2>0$ 和 n_0，使得$\forall n\geq n_0$，有 $c_1*g(n)\leq f(n)\leq c_2*g(n)$。由于 $c_1\neq 0$，$c_2\neq 0$，$f(n)\geq c_1*g(n)$可得 $g(n)\leq 1/c_1*f(n)$，同时，$f(n)\leq c_2*g(n)$，有 $g(n)\geq 1/c_2*f(n)$，即 $1/c_2*f(n)\leq g(n)\leq 1/c_1*f(n)$，故 g=Θ (f)。

3）　**O (f+g)= O (max(f, g))，对于Ω 和Θ 同样成立。**

证明：设 F(n)= O (f+g)，则存在 $c_1>0$，和 n_1，使得$\forall n\geq n_1$，有

$F(n)\leq c_1 (f(n)+g(n))$

$= c_1 f(n) + c_1 g(n)$

$\leq c_1*\max\{f,g\}+ c_1*\max\{f,g\}$

$=2\ c_1*\max\{f,g\}$

所以，F(n)= O (max(f, g))，即O (f+g)= O (max(f, g))

对于Ω 和Θ 同理证明可以成立。

4）　**log(n!)= Θ(nlogn)**

证明：

•由于 log(n!)=$\sum_{i=1}^{n}\log i$ ≤$\sum_{i=1}^{n}\log n$=nlogn，所以可得 log(n!)= O (nlogn)。

•由于对所有的偶数 n 有，

log(n!)= $\sum_{i=1}^{n}\log i \geq \sum_{i=n/2}^{n}\log i \geq \sum_{i=n/2}^{n}\log n/2 \geq$(n/2)log(n/2)=(nlogn)/2-n/2。

当 n≥4，(nlogn)/2-n/2≥(nlogn)/4，故可得∀n≥4，log(n!) ≥(nlogn)/4，即 log(n!)= Ω (nlogn)。

综合以上两点可得 log(n!)= Θ(nlogn)

2. 设计一个算法，求给定 n 个元素的第二大元素，并给出算法在最坏情况下使用的比较次数。（复杂度至多为 2n-3）

算法：

```
Void findsecond(ElemType A[])
{
    for (i=2; i<=n;i++)
       if (A[1]<A[i])
       {
          temp=A[1];
           A[1]=A[i];
           A[i]=temp;
          }
       for (i=3; i<=n;i++)
       if (A[2]<A[i])
```

```
        {
           temp=A[1];
            A[1]=A[i];
            A[i]=temp;
           }
      return A[2];
}
```

该算法使用的比较次数为：2n-3

3. 设计一个算法，求给定 n 个元素的最大和最小元素。（要求算法的复杂度至多为 1.5n）

算法：

```
void Maxmin2(A;l,r;int x;int y);
{
        if    (l=r) { x=A[l]; y=A[r]; return;}
        if    (r-l=1)
           {       if (A[l]<A[r])    { x=A[l]; y=A[r];}
                   else { x=A[r]; y=A[l];}
                   return;
           }
        else { mid:=(l+r) div 2;
                        Maxmin2(A,l,mid,x1,y1);
                        Maxmin2(A,mid+1,r,x2,y2);
```

x=min(x1,x2); y=max(y1,y2); }

}

该算法使用的比较次数为：1.5n-2

4. 给定多项式 $p(x)=a_nx_n+ a_{n-1}x_{n-1}+\ldots+a_1x+a_0$，假设使用以下方法求解：

$p=a_0$;

xpower=1;

for (i=1; i<=n; i++)

{ xpower=x * xpower;

$p=p+a_i$ * xpower;

}

求（1）该算法最坏情况下使用的加法和乘法分别为多少次？

（2）能不能对算法的性能进行提高？

解：（1）该算法最坏情况下使用的加法 n 次，乘法 2n 次

（2）改进的算法为：

```
float Horner(A, float x)
{
  p=A[n+1];
  for (j=1; j<=n; j++)
        p=x*p+A[n-j];
  return p;
}
```

该算法中使用加法 n 次，乘法 n 次

第二章

1．求解下列递推关系：

1）当 n≥1 时，f(n)=3f(n-1)；f(0)=5

解：$f(n)=3f(n-1)=3^2f(n-2)=\ldots=3^nf(n-n)=3^n*5=5*3^n$

2) 当 n≥2 时，f(n)=5f(n-1)-6f(n-2)； f(0)=1；f(1)=0

解：该递推关系的特征方程为：$x^2-5x+6=0$

特征根为：$r_1=2$；$r_2=3$

故 $f(n)=c_1*2^n+c_2*3^n$

有 $f(0)= c_1*2^0+c_2*3^0== c_1+c_2=1$ 且 $f(1)= c_1*2^1+c_2*3^1== 2c_1+c3_2=0$

可得 $c_1=3$， $c_2=-2$

故 $f(n)=3*2^n-2*3^n$

3) 当 n≥1 时，f(n)=f(n-1)+n²； f(0)=0

解：$f(n)= f(n-1)+n^2$

$= f(n-2)+ (n-1)^2+n^2$

$=\ldots.$

$= f(0)+1^2+2^2+\ldots+ (n-1)^2+n^2$

$=1^2+2^2+\ldots+ (n-1)^2+n^2$

$=1/6\ n(n+1)(2n+1)$

4) 当 n≥1 时，f(n)=2f(n-1)+n²； f(0)=1

解：设 $f(n)=2^nf'(n)$，且 $f'(0)= f(0)=1$

则 $2^n f'(n)=2*(2^{n-1}f'(n-1))+n^2$

即 $f'(n)=f'(n-1)+\frac{n^2}{2^n}$

$=f'(0)+\sum_{i=1}^{n}\frac{i^2}{2^i}$

$=1+\sum_{i=1}^{n}\frac{i^2}{2^i}$

所以 $f(n)=2^n*(1+\sum_{i=1}^{n}\frac{i^2}{2^i})=2^n*(10-\frac{n+6}{2^n})=10*2^n-(n+6)$

5) 当 n≥1 时，f(n)=nf(n-1)+1； f(0)=1

解：$f(n)=n!*(f'(0)+\sum_{i=1}^{n}\frac{1}{i!})$

$=n!*(1+\sum_{i=1}^{n}\frac{1}{i!})$

2.求解下面的递推式：当 n≥2 时，f(n)=4f(n/2)+n； f(1)=1。假设 n 为 2 的幂，用直接展开法求解递推式。

解：令 $n=2^k$

$f(n)=4f(n/2)+n$

$=4*(4f(\frac{n}{2^2})+\frac{n}{2})+n$

$=4^2 f(\frac{n}{2^2})+2n+n$

$=4^3 f(\frac{n}{2^3})+2^2 n+2n+n$

$=\ldots.$

$=4^k f(\frac{n}{2^k})+2^{k-1}n+\cdots+2n+n$

$=4^k f(1)+(2^{k-1}+\cdots+2+1)n$

$=n^2+(2^k-1)n$

$=2n^2-n$

3.求解下面的递推式：当 n≥2 时，f(n)=9f(n/3)+n²； f(1)=1。假设 n 为 3 的幂，用直接展开法求解递推式。

解：令 $n=3^k$

$$
\begin{aligned}
f(n) &= 9f(\frac{n}{3})+n^2 \\
&= 9(9f(\frac{n}{3^2})+(\frac{n}{3})^2)+n^2 \\
&= 9^2 f(\frac{n}{3^2})+n^2+n^2 \\
&= 9^3 f(\frac{n}{3^3})+n^2+n^2+n^2 \\
&= \cdots \\
&= 9^k f(\frac{n}{3^k})+kn^2 \\
&= n^2+n^2\log_3 n
\end{aligned}
$$

4. 法求解递推式的上界：当 n≥4 时，$f(n)=f(\lfloor\frac{n}{4}\rfloor)+f(\lfloor\frac{3n}{4}\rfloor)+n$； 当 n<4 时，f(n)=4。

解：

由于递推式为 $f(n)=\begin{cases}4 & n<4\\ f(\lfloor\frac{n}{4}\rfloor)+f(\lfloor\frac{3n}{4}\rfloor)+n & n\geq 4\end{cases}$

这里 $\frac{1}{4}+\frac{3}{4}=1$

故作猜想 $f(n)=2f(\frac{n}{2})+n$ 的解为： $f(n)=n\log n+4n$

故对原递推式做猜想 $f(n)\leq cn\log n+4n$

由于 $f(n)=f(\lfloor\frac{n}{4}\rfloor)+f(\lfloor\frac{3n}{4}\rfloor)+n$

$$\le c\left\lfloor\frac{n}{4}\right\rfloor\log\left\lfloor\frac{n}{4}\right\rfloor+4\left\lfloor\frac{n}{4}\right\rfloor+c\left\lfloor\frac{3n}{4}\right\rfloor\log\left\lfloor\frac{3n}{4}\right\rfloor+4\left\lfloor\frac{3n}{4}\right\rfloor+n$$

$$\le c\frac{n}{4}\log\frac{n}{4}+4\frac{n}{4}+c\frac{3n}{4}\log\frac{3n}{4}+4\frac{3n}{4}+n$$

$$=\frac{1}{4}cn(\log n-\log 4)+\frac{3}{4}cn(\log n+\log\frac{3}{4})+5n$$

$$=cn\log n-(2-\frac{3}{4}\log 3)cn+5n$$

若使 f(n)满足上界为$cn\log n+4n$则必有

$cn\log n-(2-\frac{3}{4}\log 3)cn+5n\le cn\log n+4n$

即$-(2-\frac{3}{4}\log 3)cn+n\le 0$

所以$c\ge\frac{1}{2-\frac{3}{4}\log 3}=1.23$

故$f(n)\le 1.23n\log n+4n$，即上界为$1.23n\log n+4n$

4. 设计算法，求解问题：有一楼梯共 M 级，刚开始时你再第一级，若每次只能跨上一级或二级，要走上第 M 级，共有多少种走法？

```
int  fa(int  m)
{
    int  z;
    if  (m==1)  z=1;
   else  if  (m==2)  z=2;、
   else  z=fa(n-1)+fa(n-2);
   return  z;
}
```

5. 设计算法，一个射击运动员打靶，靶一共有 10 环，连开 10 枪打中 90 环的可能性有多少种？

这道题的思路与字符串的组合很像，用递归解决。一次射击有 11 种可能，命中 1 环至 10 环，或脱靶。

函数功能： 求解 number 次打中 sum 环的种数

函数参数： number 为打靶次数，sum 为需要命中的环数，result 用来保存中间结果，total 记录种数

```
void ShootProblem_Solution(int number, int sum, vector<int> &result, int &total)
{
I    if(sum < 0 || number * 10 < sum)
     //加 number * 10 < sum 非常重要，它可以减少大量的递归，类似剪枝操作
     return;
     if(number == 1) //最后一枪
     {
          if(sum <= 10) //如果剩余环数小于 10，只要最后一枪打 sum 环就可以了
          {
               for(unsigned i = 0; i < result.size(); i++)
               cout<<result[i]<<' ';
               cout<<sum<<endl;
               total++;
```

```
                return;
            }
            else
                return;
        }
        for(unsigned i = 0; i <= 10; i++) //命中 0-10 环
        {
            result.push_back(i);
            ShootProblem_Solution(number-1, sum-i, result, total); //针对剩余环数递归求解
            result.pop_back();
        }
    }

    void ShootProblem(int number, int sum)
    {
        int total = 0;
        vector<int> result;
        ShootProblem_Solution(number, sum, result, total);
        cout<<"total nums = "<<total<<endl;
    }

    int main()
```

```
{
    ShootProblem(10, 90);
    return 0;
}
```

6. 设计算法，求解猴子吃桃问题：有一群猴子摘来了一批桃子，猴王规定每天只准吃一半加一只（即第二天吃剩下的一半加一只，依此类推），第九天正好吃完，问猴子们摘来了多少桃子？

思路：可假设有第十天，则第十天剩余的桃子数目为 0，由此递推可得每一天剩余的桃子数目。第一天的桃子数目即为猴子摘桃子的总数。假设有第 10 天，则第 10 天吃 0 个桃子，倒推出前一天的个数 x，x[9]=2(x[10]+1)。

```
void main()
{
    int i=0;
    int num[11];
    num[10]=0;
    for(i=9;i<=1;i--)
        num[i]=2*(num[i+1]+1);
    cout<<num[1];
}
```

第三章

1.设计一个分治算法，判定两棵给定的二叉树 T_1 和 T_2 是否相同。

采用先序遍历算法来实现，其中访问结点的操作为“判断两个结点是否相同”，如果 t1 和 t2 所指的结点均为空或均不为空且数据域的值相等，则继续先序比较它们的左、右子树，否则返回 0。具体算法如下：

```
int EqualBtr(bitreptr t1, bitreptr t2)
{
    if(t1 == NULL && t2 == NULL)
        return (1);
    if((t1==NULL && t2!=NULL) || t1!=NULL && t2== NULL) ||
(t1->data!= t2->data))
        return (0);
    hl = EqualBtr (t1->Lchild, t2->Lchild);
    hr = EqualBtr (t1->Rchild, t2->Rchild);
    if(hl == 1 && hr == 1)
        return 1;
    else
        return 0;
}
```

2.设计一个分治算法，求给定二叉树的高度。

设根结点为第一层的结点，所有 k 层结点的左、右孩子结点在 k+1 层。因此，可以通过先序遍历计算二叉树中每个结点的层次，其中最大值即为该二叉树的深度。具体算法如下：

```
void Btdepth(bitreptr t, int k, int &h)
//指针 t 指向二叉树的根结点，k, h 的初值均设为 0
{
    if (t! = NULL)
    {
        k++;       //表示结点的层次
        if (k > h)
            h = k;
        Btdepth(t->Lchild, k, h);
        Btdepth(t->Rchild, k, h);
    }
}
```

3.设计一个分治算法，在一个具有 n 个数的数组中找出第二大元素，并给

出算法的复杂度。

```
typedef struct MyTwoMax
{
int Max;
int SecMax;
}MyMax;
MyMax  getSecMax(int  A[],int  low,int  high)//分治求数值中最大的两个元素
{
    MyMax lm,rm,mm;
    int mid;
    if(high-low<=1)
    {
        if(high-low==1)
        {
            mm.Max=max(A[low],A[high]);
            mm.SecMax=min(A[low],A[high]);
            return mm;
        }
       else
      {
            mm.Max=A[low];
            mm.SecMax=0;
```

```
                return mm;
            }
        }
        else
        {
            mid=(low+high)/2;
            lm=getSecMax(A[],low,mid);
            rm=getSecMax(MyInt,mid+1,high);
         }
         mm.Max=max(lm.Max,rm.Max);
         mm.SecMax=max(min(lm.Max,rm.Max),max(lm.SecMax,rm.SecMa
x));
         return mm;
    }
```

算法复杂度为：$f(n)=\begin{cases}1 & n=2\\ 2f(\frac{n}{2})+3 & n>2\end{cases}$，故算法复杂度为$2n-3$

5. 设计一个算法，求出给定 5 个元素的中间元素。要求最坏情况下使用 6 次比较。

(课上讲过)

6. 用分治法，求两个大整数 X=5287，Y=3462 的乘积。

解：

1）

x=5287　　a=52　　b=87　　a-b= -35

y=3462　　c=34　　d=62　　d-c= 28

ac=(1768)

bd=(5394　)

(a-b)(d-c)=(-980)

$xy=ac10^4+[(ac + bd + (a-b)(d-c)]10^2+ bd$

=17680000+（1768+5394-980）*100+5394

=18303594

2）

a=52　　a_1=5　　b_1=2　　a_1-b_1=3

c=34　　c_1=3　　d_1=4　　d_1-c_1=1

a_1c_1=15　　b_1d_1=8　　$(a_1-b_1)(d_1-c_1)$=3

ac=1500+(15+8+3)10+8=1768

3）

b=87　　a_2=8　　b_2=7　　a_2-b_2=1

d=62　　c_2=6　　d_2=2　　d_2-c_2= -4

a_2c_2=48　　b_2d_2=14　　$(a_2-b_2)(d_2-c_2)$= -4

bd=4800+(48+14-4)10+14=5394

4）

|a-b|=35　　a_3=3　　b_3=5　　a_3-b_3= -2

|d-c|=28　　c_3=2　　d_3=8　　d_3-c_3=6

a_3c_3=6　　b_3d_3=40　　$(a_3-b_3)(d_3-c_3)$= -12

| (a-b)(d-c) |=600+(6+40-12)10+40=980

第四章

1.求下列两个序列的最长公共子序列 A=“xzyzzyx”， B=“zxyyzxz”，并给出一个最优解。

解： $$L[i,j]=\begin{cases}0 & \text{if } i=0 \text{ or } j=0\\ \mathrm{L[i-1,j-1]}+1 & \text{if } i\succ 0\ ,\ j\succ 0 \text{ and } a_i=b_j\\ \max\{L[i,j-1],L[i-1,j]\} & \text{if } i\succ 0\ ,\ j\succ 0 \text{ and } a_i\neq b_j\end{cases}$$

相等时取←优先。故由下图可知其中一个最优解为：zyyx；最长公共子序列的长度为 4。

		y_j	z	x	y	y	z	x	z
	i	0	1	2	3	4	5	6	7
	j								
	0	0	0	0	0	0	0	0	0
x	1	0	0	1 ←	1 ←	1 ←	1	1 ←	1
z	2	0	**1** ←	1 ←	1 ←	1	2 ←	2 ←	2
y	3	0	1 ←	1	**2**	2 ←	2 ←	2 ←	2
z	4	0	1 ←	1	2 ←	2	3 ←	3	3
z	5	0	1 ←	1	2 ←	2	3 ←	3	4

y 6 0 1 ← 1 2 **3** ← 3 ← 3 4

↑ ↖ ↑ ↑ ↖

x 7 0 1 2 ← 2 3 3 **4** ← 4

2. 求对下列 5 个矩阵连乘：M1:4×5；M2:5 ×4 ；M3:4 ×6；M4:6 ×4；M5:4 ×5

1）求这 5 个矩阵连乘时所需要的最少乘法次数；

2）给出一个最优解。

解： $C[i,j]=\min_{i<k\le j}\{C[i,k-1]+C[k,j]+r_i r_k r_{j+1}\}$

1）单个矩阵相乘：C[1,1]==C[2,2]=0 ……=C[5,5]=0

2）相邻两个矩阵相乘：

C[1,2]=min (1<k≤2) {C[1,1]+C[2,2]+r1*r2*r3} =4*5*4=80

C[2,3]=min (2<k≤3) {C[2,2]+C3,3]+r2*r3*r4} =5*4*6=120

C[3,4]=min (3<k≤4) {C[3,3]+C[4,4]+r3*r4*r5} =4*6*4=96

C[4,5]= min (4<k≤5) {C[4,4]+C[5,5]+r4*r5*r6} =6*4*5=120

3）三个矩阵相乘：

C[1,3]= min (1<k≤3) {C[1,1]+C[2,3]+r1*r2*r4， C[1,2]+C[3,3]+r1*r3*r4}

=min{0+120+4*5*6 ， 80+4*4*6}=176

C[2,4]= min (2<k≤4) {C[2,2]+C[3,4]+r2*r3*r5， C[2,3]+C[4,4]+r2*r4*r5}

=min{96+5*4*4 ， 120+5*6*4}=176

C[3,5]= min (3<k≤5) {C[3,3]+C[4,5]+r3*r4*r6， C[3,4]+C[5,5]+r3*r5*r6}

=min{120+4*6*5 ， 96+4*4*5}=176

4）四个矩阵相乘：

C[1,4]= min (1<k≤4) {C[1,1]+C[2,4]+r1*r2*r5， C[1,2]+C[3,4]+r1*r3*r5，

C[1,3]+C[4,4]+r1*r4*r5 }

=min{176+4*5*4 ， 80+96+4*4*4，176+4*6*4}=240

C[2,5]= min (2<k≤5) {C[2,2]+C[3,5]+r2*r3*r6， C[2,3]+C[4,5]+r2*r4*r6，

C[2,4]+C[5,5]+r2*r5*r6 }

=min{176+5*4*5 ，120+120+5*6*5，176+5*4*5}=276

5）五个矩阵连乘：

C[1,5]= min (1<k≤5) {C[1,1]+C[2,5]+r1*r2*r6， C[1,2]+C[3,5]+r1*r3*r6，

C[1,3]+C[4,5]+r1*r4*r6， C[1,4]+C[5,5]+r1*r5*r6 }

=min{276+4*5*5, 80+176+4*4*5, 176+120+4*6*5,

240+4*4*5}

=320

最优解为：（（M1*M2）*（M3*M4））*M5

3. 求解下面旅行推销员问题，并给出最优周游路线。

$$D=\begin{bmatrix}0 & 2 & 4 & 6\\ 2 & 0 & 1 & 2\\ 5 & 9 & 0 & 1\\ 9 & \infty & 2 & 8\end{bmatrix}$$

解：

1）

f(v2:Φ)=d21=2 f(v3:Φ)=d31=5 f(v4:Φ)=d41=9

2）

f(v2: {v3})= d23+f(v3:Φ)=1+5=6　f(v2: v4)= d24+f(v4:Φ)=2+9=11

f(v3: {v2})= d32+f(v2:Φ)=9+2=11　f(v3: v4)= d34+f(v4:Φ)=1+9=10

f(v4: {v2})= d42+f(v2:Φ)=∞　　f(v4: v3)= d43+f(v3:Φ)=2+5=7

3）

f(v2: {v3 ,v4})= min{d23+f(v3: {v4 }), d24+f(v4: {v3}) }= min{1+10, 2+7}=9

f(v3: {v2 ,v4})= min{d32+f(v2: {v4 }), d34+f(v4: {v2}) }= min{9+11, 1+∞}=20

f(v4: {v2 ,v3})= min{d42+f(v2: {v3}), d43+f(v3: {v2}) }= min{ ∞ +6, 2+11}=13

4）

f(v1: {v2 ,v3 ,v4})= min{d12+f(v2: {v3 ,v4}), d13+f(v3: {v2 ,v4}) , d14+f(v4: {v2 ,v3}) }

= min{2+9, 4+20, 6+13}=11

最佳周游路线为：V1→V2→V4→V3→V1

4.用动态规划法，求解 0-1 背包问题，已知背包容量为 22，5 件物品的体积分别为 3,5,7,8,9，价值分别为 4,6,7,9,10，求该背包的最大价值及物品选择情况。

解：
$$V[i,j]=\begin{cases}0 & \text{if } i=0 \text{ or } j=0\\ V[i-1,j] & \text{if } j\prec s_i\\ \max\{V[i-1,j],V[i-1,j-s_i]+v_i\} & \text{if } i\succ 0 \text{ and } j\geq s_i\end{cases}$$

由下表可知，该背包的最大价值是 25，最优解为：（0,1,0,1,1）

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	0	0	0	4	4	6	6	6	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
3	0	0	0	4	4	6	6	7	10	10	11	11	13	13	13	17	17	17	17	17	17	17	17
4	0	0	0	4	4	6	6	7	10	10	11	13	13	15	15	17	19	19	20	20	22	22	22
5	0	0	0	4	4	6	6	7	10	10	11	13	14	15	16	17	19	20	20	21	23	23	25

第五章

1. 用贪心法求解部分背包问题，已知 n=3，C=40，(w1,w2,w3)=(28,15,24), (p1,p2,p3)=(35,25,24) .

解：

r1=35/28=1.25; r2=25/15=1.67; r3=24/24=1

可知 r2>r1>r3

故从第二件物品开始贪心选择：

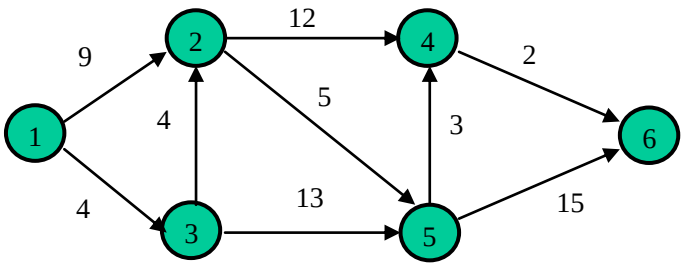
判定条件　背包已用部分　　背包剩余部分　　背包总价值　　物品放入情况

C>w2　　15　　25　　25　　(0, 1, 0)

C-w2<w1　　40　　0　　25+(25/28)*35　　(0.893, 1, 0)

即，背包最大价值为 56.25，放置情况为(0.893, 1, 0)

2. 用 dijkstra 算法求解下图的单源最短路径问题，设源点为 1。



解：

1） X={1}, Y={2,3,4,5,6}

λ[1]=0, λ[2]=9, λ[3]=4, λ[4]=∞, λ[5]=∞, λ[6]=∞

2） 选取 Y 集合中标记λ[]的最小值为λ[3]=4, 将顶点 3 加入到 X 集合，即

X={1, 3}, Y={2, 4,5,6}

修改和顶点 3 相关顶点的λ[]值，即

λ[2]=8, λ[4]=∞, λ[5]=17, λ[6]=∞

3） 选取 Y 集合中标记λ[]的最小值为λ[2]=8, 将顶点 2 加入到 X 集合，即

X={1, 2, 3}, Y={4,5,6}

修改和顶点 2 相关顶点的λ[]值，即

λ[4]=20, λ[5]=13, λ[6]=∞

4） 选取 Y 集合中标记λ[]的最小值为λ[5]=13, 将顶点 5 加入到 X 集合，即

X={1, 2, 3, 5}, Y={4, 6}

修改和顶点 5 相关顶点的λ[]值，即

λ[4]=16, λ[6]=28

5） 选取 Y 集合中标记λ[]的最小值为λ[4]=16, 将顶点 4 加入到 X 集合，即

X={1, 2, 3, 4, 5}, Y={6}

修改和顶点 4 相关顶点的λ[]值，即

λ[6]=18

6） 选取 Y 集合中标记λ[]的最小值为λ[6]=18, 将顶点 6 加入到 X 集合，即

X={1, 2, 3, 4, 5, 6}, Y=∅

故从源点到各个顶点的最短路径为：

1→3→2: 8

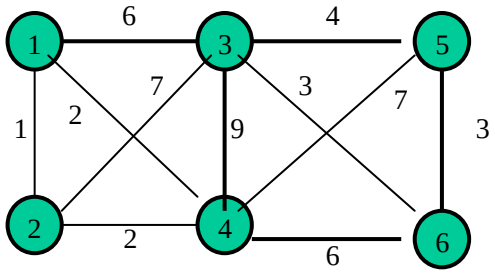
1→3: 4

1→3→2→5: 13

1→3→2→5→4: 16

1→3→2→5→4→6: 18

3. 分别用 Kruscal 和 Prim 方法求下图的最小耗费生成树。



解：

1）Prim 方法：（选取从 X 集合中顶点出发到 Y 集合中顶点终止的边里面

权值最小的，并将该终点加入到 X 集合）

①X={1}， Y={2, 3, 4, 5, 6}

可选边为:（1,2,1），（1,3,6），（1,4,2）

权值最小边为（1,2,1）

故 X={1,2}， Y={3, 4, 5, 6}

②X={1,2}， Y={3, 4, 5, 6}

可选边为:（1,3,6），（1,4,2），（2,3,7），（2,4,2）

权值最小边为:（1,4,2）

故 X={1,2,4}， Y={3, 5, 6}

③X={1,2,4}， Y={3, 5, 6}

可选边为:（1,3,6），（2,3,7），（4,3,9），（4,5,7），（4,6,6）

权值最小边为:（1,3,6）

故 X={1,2,3,4}， Y={5, 6}

④X={1,2,3,4}， Y={5, 6}

可选边为:（3,5,4），（3,6,3），（4,5,7），（4,6,6）

权值最小边为:（3,6,3）

故 X={1,2,3,4,6}， Y={6}

⑤X={1,2,3,4,6}， Y={6}

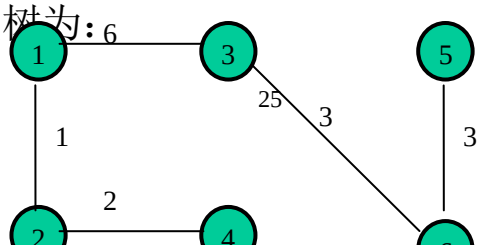
可选边为:（3,5,4），（4,5,7），（6,5,3）

权值最小边为:（6,5,3）

故 X={1,2,3,4,5,6}， Y=∅

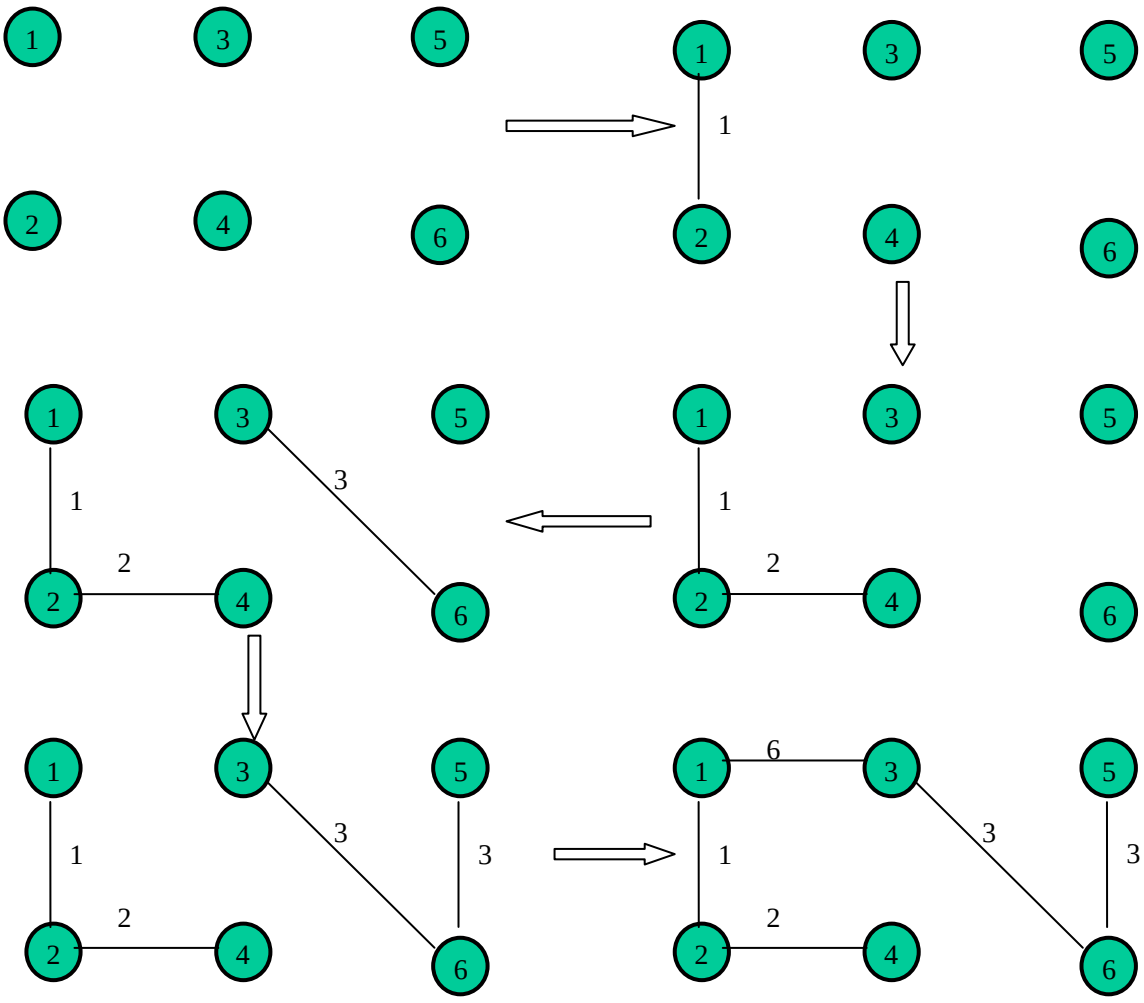
故最小耗费生成树的总耗费为：1+2+6+3+3=15

构造的最小耗费生成树为:



2）Kruscal 方法：（每次从边集里面选取边的权值最小且不和已有边构成回路）

由图示可知：最小耗费生成树的总耗费为：1+2+3+3+6=15，共可以生成 4 棵最小耗费生成树。



第六章

1. 用回溯法求解{1,2,3,4,5}这 5 个自然数中任取 3 个数的组合。

解：

从 n 个不同元素中取 r 个不重复的元素组成一个子集，而不考虑其元素的顺序，称为从 n 个中取 r 个的无重组合，例如 R = {1,2,3,4,5}, n = 5, r = 3 则搜索空间树（略）：

无重组合为：{1,2,3}; {1,2,4}; {1,2,5};{1,3,4}; {1,3,5}; {1,4,5};{2,3,4};{2,3,5};{2,4,5};{3,4,5}

```
int n, r;
int C[5];
char used[5];

void combine(int pos, int h)
{
    int i;     //如果已选了 r 个元素了，则打印它们
   if (pos==r)
   {
       for (i=0; i<r; i++)
           cout<< C[i];
       cout<< endl;
       return;
   }
```

```
    for (i=h; i<=n-r+pos; i++) //对于所有未用的元素
        if (!used[i])
        {
            C[pos] = i+1;       //把它放置在组合中
            used[i]++;          //使用该元素
            combine(pos+1,i+1);     //搜索第 i+1 个元素
            used[i]--; //恢复递归前的值
        }
}
```

2. 用回溯法求 6 皇后的解。

解：搜索空间树（略）；可行解为：X_1=(2,4,6,1,3,5); X_2=(3,6,2,5,1,4); X_3=(4,1,5,2,6,3); X_4=(5,3,1,6,4,2)

3. 设计一个回溯算法，生成 1,2,3,…,n 的全排列。

```
    void Permutatons1(int n)
    {
       for (j=1; j<=n; j++)
            p[j]=j;
       perm1(1);
    }
    void Perm1(int m);
    {
```

```
    if (m=n)    output p[1..n]
    else
    {    for (j=m; j<=n; j++)
          {      interchange p[j] and p[m];
                 perm1(m+1);
                //comment:At this point p[m..n]=m,m+1,…,n
                 interchange p[j] and p[m];
          }
   }
}
```