1.引论

1.2 蛮力法

思考题 1：求一个 4 位数 x 和一个一位数 y，使得 x 与 y 的乘积 z 也是 4 位数，且 x, y, z 包含了所有 1 到 9 的数字。

```
public class Ch1Ex0 {
  public static boolean checkput(boolean[] set, int k){
    while (k>0 && !set[k%10]){ set[k%10]=true; k/=10; } return k==0; }
  public static void main(String[] args){ PrintWriter fout=null;
    try{ fout=new PrintWriter(new FileWriter("out0.txt")); }
    catch(IOException e){ System.out.println("Error!"); }
    long begin=System.currentTimeMillis(), end;
    boolean[] dset=new boolean[10];
    for (int i=2; i<=7; ++i) for (int j=1234; j<=9876/i; ++j){
      Arrays.fill(dset, false); dset[0]=dset[i]=true;
      if (checkput(dset, j) && checkput(dset, i*j))
        fout.println(j+"*"+i+"="+i*j); }
    end=System.currentTimeMillis();
    fout.println("Elapsed time: "+(end-begin)); fout.close(); }
}
```

```
1738*4=6952
1963*4=7852
Elapsed time: 0
```

思考题 2：求一个 3 位数 x 和一个 2 位数 y，使得 x 与 y 的乘积 z 是 4 位数，且 x, y, z 包含了所有 1 到 9 的数字。

思考题 3：求一个 3 位数 x 使得 2x 与 3x 也是 3 位数，且它们包含了所有 1 到 9 的数字。

思考题 4：求一个 3 位数 x 和一个 2 位数 y，使得 x 与 y 的乘法算式只包含数字 2，3，5，7。

```
public class Ch1Ex1 {
  public static boolean checkput(boolean[] set, int k){
    while (k>0 && !set[k%10]){ set[k%10]=true; k/=10; } return k==0; }
  public static boolean check(boolean[] set, int k){
    while (k>0 && set[k%10]) k/=10; return k==0; }
  public static void main(String[] args){ PrintWriter fout=null;
    try{ fout=new PrintWriter(new FileWriter("out1.txt")); }
    catch(IOException e){ System.out.println("Error!"); }
    long begin=System.currentTimeMillis(), end;
    boolean[] dset=new boolean[10];
    for (int i=2; i<=7; ++i) for (int j=1234; j<=9876/i; ++j){
      Arrays.fill(dset, false); dset[0]=dset[i]=true;
      if (checkput(dset, j) && checkput(dset, i*j))
        fout.println(j+"*"+i+"="+i*j); }
    for (int i=12; i<=79; ++i) for (int j=123; j<=9876/i; ++j){
      Arrays.fill(dset, false); dset[0]=true;
      if (checkput(dset, i) && checkput(dset, j) && checkput(dset, i*j))
        fout.println(j+"*"+i+"="+i*j); }
    for (int i=123; i<=329; ++i){ Arrays.fill(dset, false); dset[0]=true;
```

```
      if (checkput(dset, i) && checkput(dset, i*2) && checkput(dset, i*3))
        fout.println(i+", "+i*2+", "+i*3); }
    Arrays.fill(dset, false); dset[2]=dset[3]=dset[5]=dset[7]=true;
    for (int i=123; i<=987; ++i) for (int j=12; j<=98; ++j)
      if (check(dset, i) && check(dset, j) && check(dset, i*(j%10))
           && check(dset, i*(j/10)) && check(dset, i*j))
        fout.println(i+"*"+j+"="+i*(j%10)+"+10*"+i*(j/10)+"="+i*j);
    end=System.currentTimeMillis();
    fout.println("Elapsed time: "+(end-begin)); fout.close(); }
}
```

```
1738*4=6952
1963*4=7852
483*12=5796
297*18=5346
198*27=5346
157*28=4396
186*39=7254
138*42=5796
159*48=7632
192, 384, 576
219, 438, 657
273, 546, 819
327, 654, 981
775*33=2325+10*2325=25575
Elapsed time: 16
```

思考题 5：实现蛮力算法 A1.2.1 和 A1.2.2，并用来求解子集和数问题。

```
public interface Copiable { Copiable cloneSelf(); void copy(Object obj); }
public interface ItrObj { boolean next(); }
public interface ComObj  extends ItrObj, Copiable {}
public interface ComProb { boolean target(Object obj); void output(Object obj);}
public class BruteForce {
   public static int find(ItrObj itr, ComProb prob, int bound){ int num=0;
      do if (prob.target(itr)){ prob.output(itr); ++num; }
      while (num<bound && itr.next());
      return num;
   }
   public static int count(ItrObj itr, ComProb prob){ int num=0;
      do if (prob.target(itr)) ++num; while (itr.next());
      return num;
   }
}
public class BinSet implements Copiable { int n; boolean[] set;
   public BinSet(int k){ n=k; set=new boolean[k];
   }
   public Copiable cloneSelf(){
```

```
      BinSet ans=new BinSet(n); ans.copy(this);return ans;
   }
   public void copy(Object obj){ BinSet ob=(BinSet)obj;
      n=ob.n; for (int i=0; i<n; ++i) set[i]=ob.set[i];
   }
   public boolean contains(int i){ return set[i]; }
}
public class LexBinSet extends BinSet implements ComObj {
   public LexBinSet(int k){ super(k); }
   public boolean next(){  boolean hasn=true; int i=0;
      while (i<n && set[i]) set[i++]=false;
      if (i>=n) hasn=false; else set[i]=true;
      return hasn;
   }
}
public class BinSumSet implements ComProb {
   int n, sum; int[] data; PrintWriter fout;
   public BinSumSet(int k, int s, int[] d, PrintWriter f){
      n=k; sum=s; data=d; fout=f;
   }
   public boolean target(Object obj){ int s=0; BinSet ob=(BinSet)obj;
      for (int i=0; i<n; ++i) if (ob.contains(i)) s+=data[i];
      return s==sum;
   }
   public void output(Object obj){ BinSet ob=(BinSet)obj;
      fout.print("The sum of ");
      for (int i=0; i<n; ++i) if (ob.contains(i)) fout.print(data[i]+", ");
      fout.println("is "+sum);
   }
   public static void main(String[] args) {
      PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("outbinset40.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(); int m=40, n=1000, sum=m*n/4;
      int[] d=new int[m]; Random rm=new Random(m);
      for (int i=0; i<m; ++i) d[i]=rm.nextInt(n);
      fout.print("The elements of the set are: ");
      for (int i=0; i<m; ++i)  fout.print(d[i]+", ");
      fout.println();
      BinSumSet x=new BinSumSet(m, sum, d, fout);
      LexBinSet lex =new LexBinSet(m);
      fout.println(BruteForce.count(lex, x));
      //if (BruteForce.find(lex, x, 1)==0) fout.println("Not found!");
      long end=System.currentTimeMillis();
      fout.println(m+", "+sum+":  Elapsed time: "+(end-begin));
      fout.close();
```

```
    }
}
```

The elements of the set are: 606, 868, 765, 604, 136, 298, 186, 282, 456, 847, 831, 904, 3, 316, 647, 560, 34, 446, 109, 403, 394, 815, 101, 386, 169, 771, 680, 142, 275, 344,
252910
30, 7500: Elapsed time: 192489

The elements of the set are: 382, 339, 237, 263, 96, 228, 39, 961, 391, 527, 166, 93, 316, 940, 409, 460, 473, 646, 454, 299, 218, 104, 799, 218, 639, 639, 971, 705, 525, 755, 648, 975, 309, 970, 388, 265, 340, 779, 635, 564,
The sum of 382, 339, 237, 228, 961, 391, 527, 93, 316, 940, 409, 460, 473, 646, 454, 299, 218, 799, 218, 639, 971, is 10000
40, 10000: Elapsed time: 21310

The elements of the set are: 117, 388, 193, 512, 751, 561, 636, 58, 116, 8, 200, 112, 100, 155, 628, 192, 552, 507, 215, 892, 755, 923, 591, 21, 662, 284, 738, 905, 16, 786, 470, 777, 186, 994, 156, 274, 672, 806, 694, 78, 923, 319, 873, 605, 595, 958, 62, 253, 194, 367,
The sum of 117, 388, 193, 512, 751, 561, 636, 116, 8, 200, 112, 100, 155, 628, 192, 552, 507, 215, 892, 755, 923, 591, 21, 662, 284, 738, 905, 786, is 12500
50, 12500: Elapsed time: 195500

思考题 6：实现蛮力算法 A1.2.3，并用来求解 0-1 背包问题。

```
public interface OptProb extends ComProb { int val(Object obj); }
public static Copiable findleast(ComObj itr, OptProb prob){ Copiable ans=null;
      boolean found=false; int lval=0;
      do if (prob.target(itr)){
         ans=itr.cloneSelf(); lval=prob.val(itr); found=true; }
      while (!found && itr.next());
      if (found){
         while (itr.next())
            if (prob.target(itr)){ int v=prob.val(itr);
               if (v<lval){ ans.copy(itr); lval=v; } }
      prob.output(ans); }
      return ans;
   }
public class KnapBrute implements OptProb {
   public boolean target(Object obj){ int s=0; BinSet ob=(BinSet)obj;
      for (int i=0; i<Knap.n; ++i) if (ob.contains(i)) s+=Knap.weight[i];
      return s<=Knap.wlimit;
   }
   public void output(Object obj){ BinSet ob=(BinSet)obj; int w=0, v=0;
      Knap.out.print("Select: ");
      for (int i=0; i<Knap.n; ++i)
         if (ob.contains(i)){ Knap.out.print(i+", ");
            w+=Knap.weight[i]; v+=Knap.value[i];
         }
      Knap.out.println();
      Knap.out.println("wall: "+w+",   vall: "+v);
```

```
    }
    public int val(Object obj){
       BinSet ob=(BinSet)obj; int v=0;
       for (int i=0; i<Knap.n; ++i)
          if (ob.contains(i)) v+=Knap.value[i];
       return -v;
    }
    public static void main(String[] args){PrintWriter fout=null;
       try{ fout=new PrintWriter(new FileWriter("outbrutebin30.txt")); }
       catch(IOException e){ System.out.println("Error!"); }
       int len=30, n=200, wlimit=40*len;
       Random rm=new Random(len); int[] weight=new int[len];
       Random rm1=new Random(len+1); int[] value=new int[len];
       for (int i=0; i<len; ++i){
          weight[i]=rm.nextInt(n); value[i]=rm1.nextInt(n); }
       int  wall=0, vall=0;
       for (int i=0; i<len; ++i){ wall+=weight[i]; vall+=value[i]; }
       Knap.n=len; Knap.vall=vall; Knap.value=value; Knap.wall=wall;
       Knap.weight=weight; Knap.wlimit=wlimit; Knap.out=fout;
       fout.println("weights:");
       for (int i=0; i<len; ++i) fout.print(weight[i]+", "); fout.println();
       fout.println("values:");
       for (int i=0; i<len; ++i) fout.print(value[i]+", "); fout.println();
       fout.println("wall: "+wall+"  vall: "+vall+"  wlimit: "+wlimit);
       long begin=System.currentTimeMillis(), end=0;
       KnapBrute x=new KnapBrute();
       LexBinSet lex=new LexBinSet(len);
       if (BruteForce.findleast(lex, x)==null) fout.println("Not found!");
       end=System.currentTimeMillis();
       fout.println(len+":  Elapsed time: "+(end-begin));
       fout.close(); }
}
```

weights:
53, 36, 1, 61, 5, 95, 33, 55, 93, 88, 86, 82, 85, 70, 48, 76, 84, 68, 77, 86,
values:
78, 24, 20, 27, 44, 46, 24, 52, 53, 81, 25, 78, 81, 19, 14, 14, 92, 6, 32, 55,
wall: 1282 vall: 865 wlimit: 400
Select: 0, 2, 4, 9, 11, 12, 16,
wall: 398, vall: 474
20: Elapsed time: 156

weights:
6, 68, 165, 4, 136, 98, 186, 82, 56, 47, 31, 104, 3, 116, 47, 160, 34, 46, 109, 3, 194, 15, 101, 186, 169, 171, 80, 142, 75, 144,
values:
32, 156, 184, 70, 123, 149, 129, 79, 116, 140, 171, 99, 199, 165, 114, 150, 142, 184, 64, 72, 27, 11, 39, 138, 191, 69, 75, 165, 68, 175,

wall: 2778 vall: 3496 wlimit: 1200
Select: 0, 1, 2, 3, 5, 8, 9, 10, 12, 13, 14, 16, 17, 19, 21, 24, 27, 29,
wall: 1194, vall: 2436
30: Elapsed time: 248165

1.3 组合对象的枚举

思考题 1: 如何将 1，2，3，4，5，6，7，12 分别代入 a-h，使得下面的算式 成立？(减法和除法的顺序是从上倒下，从左到右)

a * b = c
= /
d e
- =
f = g + h

思考题 2: 如何将+,-, *, /分别代入 A-D,将 1，2，3，4，5，6，7，12 分别代入 a-h，使得下面的算式 成立？ (减法和除法的顺序是从上倒下，从左到右)

a A b = c
= B
d e
C =
f = g D h

```
public class Permutation  implements Copiable { int n; int[] perm;
   public Permutation(int k){ n=k; perm=new int[k];
      for (int i=0; i<n; ++i) perm[i]=i;
   }
   public Copiable cloneSelf(){
      Permutation ans=new Permutation(n);   ans.copy(this);   return ans;
   }
   public void copy(Object obj){ Permutation ob=(Permutation)obj;
      for (int i=0; i<n; ++i) perm[i]=ob.perm[i];
   }
   public int content(int i){ return perm[i]; }
}
public class LexPerm extends Permutation implements ComObj {
   public LexPerm(int k){ super(k); }
   public boolean next(){  boolean hasn=true; int i=n-1, j=n-1;
      while (i>0 && perm[i-1]>perm[i]) --i;
      if (i==0) hasn=false;
      else { --i; while (perm[j]<perm[i]) --j;
         int t=perm[i]; perm[i]=perm[j]; perm[j]=t;  ++i; j=n-1;
         while (i<j){ t=perm[i]; perm[i]=perm[j]; perm[j]=t; ++i; --j; }
      }
      return hasn;
   }
```

```
}
public class ComObjCombine implements ComObj {
   public ComObj x1, x2; Copiable initx1;
   public ComObjCombine(ComObj y1, ComObj y2){
      x1=y1; initx1=y1.cloneSelf(); x2=y2; }
   public Copiable cloneSelf(){
      return new CopiCombine(x1.cloneSelf(), x2.cloneSelf());
   }
   public void copy(Object obj){ ComObjCombine ob=(ComObjCombine)obj;
      x1.copy(ob.x1); x2.copy(ob.x2);
   }
   public boolean next(){ boolean hasn=x1.next();
      if (!hasn){ x1.copy(initx1); hasn=x2.next();
      }
      return hasn;
   }
}
public class Olym1 implements ComProb {
   int[] data; PrintWriter fout;
   public Olym1(int[] d, PrintWriter f){ data=d; fout=f;
   }
   public boolean target(Object obj){ ComObjCombine ob=(ComObjCombine)obj;
      Permutation x1=(Permutation)ob.x1, x2=(Permutation)ob.x2;
      boolean q1 = x2.content(0)==0 ?
         data[x1.content(0)]+data[x1.content(1)]==data[x1.content(2)] :
         x2.content(0)==1 ?
         data[x1.content(0)]-data[x1.content(1)]==data[x1.content(2)] :
         x2.content(0)==2 ?
         data[x1.content(0)]*data[x1.content(1)]==data[x1.content(2)] :
         data[x1.content(2)]*data[x1.content(1)]==data[x1.content(0)];
      boolean q2 = q1 && (x2.content(1)==0 ?
         data[x1.content(2)]+data[x1.content(3)]==data[x1.content(4)] :
         x2.content(1)==1 ?
         data[x1.content(2)]-data[x1.content(3)]==data[x1.content(4)] :
         x2.content(1)==2 ?
         data[x1.content(2)]*data[x1.content(3)]==data[x1.content(4)] :
         data[x1.content(4)]*data[x1.content(3)]==data[x1.content(2)]);
      boolean q3 = q2 && (x2.content(2)==0 ?
         data[x1.content(4)]+data[x1.content(5)]==data[x1.content(6)] :
         x2.content(2)==1 ?
         data[x1.content(5)]-data[x1.content(4)]==data[x1.content(6)] :
         x2.content(2)==2 ?
         data[x1.content(4)]*data[x1.content(5)]==data[x1.content(6)] :
         data[x1.content(6)]*data[x1.content(4)]==data[x1.content(5)]);
      return q3 && (x2.content(3)==0 ?
         data[x1.content(6)]+data[x1.content(7)]==data[x1.content(0)] :
```

```
            x2.content(3)==1 ?
            data[x1.content(7)]-data[x1.content(6)]==data[x1.content(0)] :
            x2.content(3)==2 ?
            data[x1.content(6)]*data[x1.content(7)]==data[x1.content(0)] :
            data[x1.content(0)]*data[x1.content(6)]==data[x1.content(7)]);
    }
    public void output(Object obj){ ComObjCombine ob=(ComObjCombine)obj;
        Permutation x1=(Permutation)ob.x1, x2=(Permutation)ob.x2;
        output(data[x1.content(0)]); fout.print(map(x2.content(0)));
        output(data[x1.content(1)]);
        fout.print(" = "); output(data[x1.content(2)]); fout.println();
        fout.print(" =       "); fout.println(map(x2.content(1)));
        output(data[x1.content(7)]); fout.print("       ");
        output(data[x1.content(3)]);  fout.println();
        fout.print(map(x2.content(3)));  fout.println("       =");
        output(data[x1.content(6)]); fout.print(" = ");
        output(data[x1.content(5)]);
        fout.print(map(x2.content(2))); output(data[x1.content(4)]);
        fout.println(); fout.println();
    }
    private String map(int k){
        return k==0 ? " + " : k==1 ? " - " : k==2 ? " * " : " / ";
    }
    private void output(int k){
        if (k<10) fout.print(" "+k); else fout.print(k);
    }
    public static void main(String[] args) {
        PrintWriter fout=null;
        try{ fout=new PrintWriter(new FileWriter("out11.txt")); }
        catch(IOException e){ System.out.println("Error!"); }
        long begin=System.currentTimeMillis();
        int[] d={1, 2, 3, 4, 5, 6, 7, 12};
        LexPerm a1=new LexPerm(8), a2=new LexPerm(4);
        ComObjCombine x=new ComObjCombine(a1, a2);
        Olym1 olym=new Olym1(d, fout);
        if (BruteForce.find(x, olym, 10)==0) fout.println("Not found!");
        long end=System.currentTimeMillis();
        fout.println("Elapsed time: "+(end-begin));
        fout.close();
    }
}
2 * 6 = 12
=       /
7       3
-       =
5 =  1 + 4
```

```
12 /  3 =  4
 =         +
 6         1
 *         =
 2 =  7 -  5

Elapsed time: 47
```

思考题 3:蛮力算法 A1.2.3 求解旅行商问题。

```
public class TspBrute implements OptProb {
   int n; int[][] g; PrintWriter fout;
   public TspBrute(int k, int[][] d, PrintWriter f){
      n=k; g=d; fout=f;
   }
   public boolean target(Object obj){ return true;
   }
   public void output(Object obj){ Permutation ob=(Permutation)obj;
      fout.println(n+": ");
      for (int i=0; i<n; ++i){
         for (int j=0; j<n; ++j) fout.print(g[i][j]+",  "); fout.println();
      }
      for (int i=0; i<n; ++i) fout.print(ob.content(i)+", ");
   fout.println();
      fout.println(val(obj));
   }
   public int val(Object obj){ int ans=0; Permutation ob=(Permutation)obj;
      int first=ob.content(0), prev=first;
      for (int i=1; i<n; ++i){  int cur=ob.content(i);
         ans+=g[prev][cur]; prev=cur;
      }
      ans+=g[prev][first]; return ans;
   }
   public static void main(String[] args) {
      PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("outbrute12.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(); int mm=12;
      Random rm=new Random(mm); int[][] gg=new int[mm][mm];
      for (int i=0; i<mm; ++i) for (int j=0; j<mm; ++j)
         gg[i][j] = i==j ? 0 : rm.nextInt(100)+1;
      TspBrute tsp=new TspBrute(mm, gg, fout);
      LexPerm lex=new LexPerm(mm);
      BruteForce.findleast(lex, tsp);
      long end=System.currentTimeMillis();
      fout.println("Elapsed time: "+(end-begin));
      fout.close();
```

```
    }
}
12:
0,  67,  13,  57,  34,  25,  12,  51,  16,  97,  22,  69,
73,  0,  62,  8,  96,  65,  37,  32,  6,  4,  91,  83,
67,  18,  0,  67,  39,  98,  59,  78,  78,  52,  59,  8,
57,  85,  37,  0,  96,  21,  24,  79,  66,  48,  30,  96,
88,  56,  23,  53,  0,  19,  25,  92,  45,  75,  44,  56,
63,  61,  100,  26,  42,  0,  34,  40,  69,  41,  64,  24,
30,  21,  94,  38,  7,  88,  0,  81,  73,  29,  96,  82,
49,  73,  29,  1,  5,  49,  36,  0,  19,  71,  70,  25,
95,  26,  53,  87,  99,  54,  57,  56,  0,  37,  23,  60,
56,  90,  28,  8,  31,  24,  96,  22,  84,  0,  24,  12,
78,  48,  30,  9,  69,  72,  68,  39,  23,  15,  0,  7,
3,  85,  95,  6,  18,  39,  92,  45,  72,  20,  14,  0,
0, 2, 1, 8, 10, 9, 7, 3, 6, 4, 5, 11,
175
Elapsed time: 38080
```

思考题 4:用子集的递增序列表示和蛮力算法 A1.2.1 求解子集和数问题。

```
public class ComSet implements Copiable { int n, m; int[] set;
   public ComSet(int k){ n=k; m=0; set=new int[k];
   }
   public Copiable cloneSelf(){
      ComSet ans=new ComSet(n); ans.copy(this);   return ans;
   }
   public void copy(Object obj){ ComSet ob=(ComSet)obj;
      m=ob.m; for (int i=0; i<m; ++i) set[i]=ob.set[i];
   }
   public int subcardinal(){ return m; }
   public int content(int i){ return set[i]; }
}
public class LexComSet extends ComSet implements ComObj {
   public LexComSet(int k){ super(k); }
   public boolean next(){  boolean hasn=true;
      if (m==0) { set[0]=1; m=1; }
      else if (set[m-1]!=n){ set[m]=set[m-1]+1; ++m; }
      else if (m==1) hasn=false;
      else {  --m; ++set[m-1]; }
      return hasn;
   }
}
public class ComSumSet implements ComProb {
   int n, sum; int[] data; ComSet itr; PrintWriter fout;
   public ComSumSet(int k, int s, int[] d, PrintWriter f){
      n=k; sum=s; data=d; fout=f;
   }
```

```
    public boolean target(Object obj){
        int s=0; ComSet ob=(ComSet)obj; int m=ob.subcardinal();
        for (int i=0; i<m; ++i) s+=data[ob.content(i)-1];
        return s==sum;
    }
    public void output(Object obj){
        ComSet ob=(ComSet)obj; int m=ob.subcardinal();
        fout.print("The sum of ");
        for (int i=0; i<m; ++i) fout.print(data[ob.content(i)-1]+", ");
        fout.println("is "+sum);
    }
    public static void main(String[] args) {
        PrintWriter fout=null;
        try{ fout=new PrintWriter(new FileWriter("outcomset50.txt")); }
        catch(IOException e){ System.out.println("Error!"); }
        long begin=System.currentTimeMillis(); int m=50, n=1000, sum=m*n/4;
        int[] d=new int[m]; Random rm=new Random(m);
        for (int i=0; i<m; ++i) d[i]=rm.nextInt(n);
        fout.print("The elements of the set are: ");
        for (int i=0; i<m; ++i)  fout.print(d[i]+", ");
        fout.println();
        ComSumSet x=new ComSumSet(m, sum, d, fout);
        LexComSet com=new LexComSet(m);
        fout.println(BruteForce.count(com, x));
        //if (BruteForce.find(com, x, 1)==0) fout.println("Not found!");
        long end=System.currentTimeMillis();
        fout.println(m+", "+sum+":  Elapsed time: "+(end-begin));
        fout.close();
    }
}
The elements of the set are: 606, 868, 765, 604, 136, 298, 186, 282, 456, 847, 831, 904,
3, 316, 647, 560, 34, 446, 109, 403, 394, 815, 101, 386, 169, 771, 680, 142, 275, 344,
252910
30, 7500:  Elapsed time: 80621

The elements of the set are: 382, 339, 237, 263, 96, 228, 39, 961, 391, 527, 166, 93, 316,
940, 409, 460, 473, 646, 454, 299, 218, 104, 799, 218, 639, 639, 971, 705, 525, 755, 648, 975,
309, 970, 388, 265, 340, 779, 635, 564,
The sum of 382, 339, 237, 263, 96, 228, 39, 961, 391, 527, 166, 93, 316, 940, 409, 460, 473,
646, 454, 299, 218, 104, 799, 525, 635, is 10000
40, 10000:  Elapsed time: 31

The elements of the set are: 117, 388, 193, 512, 751, 561, 636, 58, 116, 8, 200, 112, 100,
155, 628, 192, 552, 507, 215, 892, 755, 923, 591, 21, 662, 284, 738, 905, 16, 786, 470, 777,
186, 994, 156, 274, 672, 806, 694, 78, 923, 319, 873, 605, 595, 958, 62, 253, 194, 367,
The sum of 117, 388, 193, 512, 751, 561, 636, 58, 116, 8, 200, 112, 100, 155, 628, 192, 552,
```

```
507, 215, 892, 755, 923, 591, 21, 662, 284, 738, 905, 16, 78, 319, 62, 253, is 12500
50, 12500:  Elapsed time: 327
```

思考题 5：用子集的递增序列表示和蛮力算法 A1.2.3 求解 0-1 背包问题。

```
public class KnapBruteCom implements OptProb {
   public boolean target(Object obj){
      int w=0; ComSet ob=(ComSet)obj; int m=ob.subcardinal();
      for (int i=0; i<m; ++i) w+=Knap.weight[ob.content(i)-1];
      return w<=Knap.wlimit;
   }
   public void output(Object obj){ int w=0, v=0;
      ComSet ob=(ComSet)obj; int m=ob.subcardinal();
      Knap.out.print("Select: ");
      for (int i=0; i<m; ++i){
         Knap.out.print((ob.content(i)-1)+", ");
         w+=Knap.weight[ob.content(i)-1]; v+=Knap.value[ob.content(i)-1];
      }
      Knap.out.println();
      Knap.out.println("wall: "+w+",   vall: "+v);
   }
   public int val(Object obj){
      int v=0; ComSet ob=(ComSet)obj; int m=ob.subcardinal();
      for (int i=0; i<m; ++i) v+=Knap.value[ob.content(i)-1];
      return -v;
   }
   public static void main(String[] args){PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("outbrutecom30.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int len=30, n=200, wlimit=40*len;
      Random rm=new Random(len); int[] weight=new int[len];
      Random rm1=new Random(len+1); int[] value=new int[len];
      for (int i=0; i<len; ++i){
         weight[i]=rm.nextInt(n); value[i]=rm1.nextInt(n); }
      int  wall=0, vall=0;
      for (int i=0; i<len; ++i){ wall+=weight[i]; vall+=value[i]; }
      Knap.n=len; Knap.vall=vall; Knap.value=value; Knap.wall=wall;
      Knap.weight=weight; Knap.wlimit=wlimit; Knap.out=fout;
      fout.println("weights:");
      for (int i=0; i<len; ++i) fout.print(weight[i]+", "); fout.println();
      fout.println("values:");
      for (int i=0; i<len; ++i) fout.print(value[i]+", "); fout.println();
      fout.println("wall: "+wall+"   vall: "+vall+"   wlimit: "+wlimit);
      long begin=System.currentTimeMillis(), end=0;
      KnapBruteCom x=new KnapBruteCom();
      LexComSet lex=new LexComSet(len);
      if (BruteForce.findleast(lex, x)==null) fout.println("Not found!");
      end=System.currentTimeMillis();
```

```
        fout.println(len+":  Elapsed time: "+(end-begin));
        fout.close(); }
}
```

weights:
53, 36, 1, 61, 5, 95, 33, 55, 93, 88, 86, 82, 85, 70, 48, 76, 84, 68, 77, 86,
values:
78, 24, 20, 27, 44, 46, 24, 52, 53, 81, 25, 78, 81, 19, 14, 14, 92, 6, 32, 55,
wall: 1282 vall: 865 wlimit: 400
Select: 0, 2, 4, 9, 11, 12, 16,
wall: 398, vall: 474
20: Elapsed time: 63

weights:
6, 68, 165, 4, 136, 98, 186, 82, 56, 47, 31, 104, 3, 116, 47, 160, 34, 46, 109, 3, 194, 15, 101, 186, 169, 171, 80, 142, 75, 144,
values:
32, 156, 184, 70, 123, 149, 129, 79, 116, 140, 171, 99, 199, 165, 114, 150, 142, 184, 64, 72, 27, 11, 39, 138, 191, 69, 75, 165, 68, 175,
wall: 2778 vall: 3496 wlimit: 1200
Select: 0, 1, 2, 3, 5, 8, 9, 10, 12, 13, 14, 16, 17, 19, 21, 24, 27, 29,
wall: 1194, vall: 2436
30: Elapsed time: 90059

1.4 回溯算法

思考题 1：一个 16 升的容器中装满了牛奶，另外有一个 6 升和一个 11 升的空容器，如何用它们来平分牛奶？

```
public interface Node {
     boolean target(); int subnum(); Node down(int i); void output(); }
class Link { public Node prob; public Link next;
   public Link(Node p, Link n){ prob=p; next=n; }
   public void output(){
      if (next!=null) next.output(); prob.output(); }
}
public class GraphSearch {
   private Link head; private int num, bound; private Set nodeset;
   public GraphSearch(Node p, int b, Set s) {
      head=new Link(p, null); bound=b; nodeset=s; nodeset.add(p); }
   public void depthfirst(){ Node p=head.prob, sub=null; int n=p.subnum();
      if (p.target()){ head.output(); ++num; }
      for (int i=0; num<bound && i<n; ++i)
         if ((sub=p.down(i))!=null && !nodeset.contains(sub)){
            head=new Link(sub, head);
            nodeset.add(sub);  depthfirst(); head=head.next; } }
   public boolean depthfirst1(){ LinkedList queue=new LinkedList();
      Link cur=null; queue.addFirst(head);
      while (num<bound && !queue.isEmpty()){
         cur=(Link)queue.removeFirst();
```

```
            Node curnode=cur.prob; int m=curnode.subnum();
            for (int i=0; i<m; ++i){ Node subnode=curnode.down(i);
               if (subnode!=null && !nodeset.contains(subnode)){
                  Link sub=new Link(subnode, cur);
                  queue.addFirst(sub); nodeset.add(subnode);
                  if (subnode.target()){
                     while (sub!=null){
                        sub.prob.output(); sub=sub.next;
                     }
                  ++num; }
               }
            }
         }
         return num!=0;
      }
      public boolean widefirst(){
         LinkedList queue=new LinkedList(); Link cur=null;
         queue.addLast(head);
         while (num<bound && !queue.isEmpty()){
            cur=(Link)queue.removeFirst();
            Node curnode=cur.prob; int m=curnode.subnum();
            for (int i=0; i<m; ++i){ Node subnode=curnode.down(i);
               if (subnode!=null && !nodeset.contains(subnode)){
                  Link sub=new Link(subnode, cur);
                  queue.addLast(sub); nodeset.add(subnode);
                  if (subnode.target()){ sub.output(); ++num;  }
               }
            }
         }
         return num!=0;
      }
      public int getnum(){ return num; }
   }
   public class Cup1  implements Node, Comparable{
      static int a, b, c;   static PrintWriter out;
      public static void setcup(int x, int y, int z){ a=x; b=y; c=2*z; }
      int i, j, k;
      public Cup1(int x, int y, int z){ i=x; j=y; k=z; }
      public Cup1(){ k=c; }
      public int compareTo(Object o){ Cup1 obj=(Cup1)o;
         return i<obj.i ? -1 : i>obj.i ? 1 : j<obj.j ? -1 : j>obj.j ? 1 :
               k<obj.k ? -1 : k>obj.k ? 1 : 0;
      }
      public boolean target(){ return i==0 && j==c/2 && k==c/2; }
      public int subnum(){ return 6; }
      public Node down(int x){
```

```
        if (x==0)
        return i==0 || j==b ? null : i<b-j ? new Cup1(0, j+i, k) :
                 new Cup1(i-b+j, b, k);
        if (x==1)
        return i==0 || k==c ? null : i<c-k ? new Cup1(0, j, k+i) :
                 new Cup1(i-c+k, j, c);
        if (x==2)
        return j==0 || i==a ? null : j<a-i ? new Cup1(i+j, 0, k) :
                 new Cup1(a, j-a+i, k);
        if (x==3)
        return j==0 || k==c ? null : j<c-k ? new Cup1(i, 0, k+j) :
                 new Cup1(i, j-c+k, c);
        if (x==4)
        return k==0 || i==a ? null : k<a-i ? new Cup1(i+k, j, 0) :
                 new Cup1(a, j, k-a+i);
        if (x==5)
        return k==0 || j==b ? null : k<b-j ? new Cup1(i, j+k, 0) :
                 new Cup1(i, b, k-b+j);
        return null;
    }
    public void output(){
        out.print(i+":"+j+":"+k+",   ");
    }
    public static void main(String[] args){PrintWriter fout=null;
        try{ fout=new PrintWriter(new FileWriter("out3.txt")); }
        catch(IOException e){ System.out.println("Error!"); }
        long begin=System.currentTimeMillis(), end=0;
        Cup1.out=fout; Cup1.setcup(6, 11, 8);
        fout.println("Depth First 1:");
        GraphSearch s=new GraphSearch(new Cup1(), 1, new TreeSet());
        s.depthfirst();
        fout.println("Depth First 2:");
        s=new GraphSearch(new Cup1(), 1, new TreeSet());
        s.depthfirst1();
        fout.println("Wide First:");
        s=new GraphSearch(new Cup1(), 1, new TreeSet());
        s.widefirst();
        end=System.currentTimeMillis();
        fout.println("Elapsed time: "+(end-begin));
        fout.close(); }
}
```

Depth First 1:

0:0:16, 6:0:10, 0:6:10, 6:6:4, 1:11:4, 0:11:5, 6:5:5, 0:5:11, 5:0:11, 5:11:0, 6:10:0, 0:10:6, 6:4:6, 0:4:12, 4:0:12, 4:11:1, 6:9:1, 0:9:7, 6:3:7, 0:3:13, 3:0:13, 3:11:2, 6:8:2, 0:8:8,

Depth First 2:

0:0:16, 0:11:5, 5:11:0, 6:10:0, 0:10:6, 6:4:6, 0:4:12, 4:0:12, 4:11:1, 6:9:1, 0:9:7, 6:3:7,

0:3:13, 3:0:13, 3:11:2, 6:8:2, 0:8:8,
Wide First:
0:0:16, 6:0:10, 0:6:10, 6:6:4, 1:11:4, 1:0:15, 0:1:15, 6:1:9, 0:7:9, 6:7:3, 2:11:3, 2:0:14, 0:2:14, 6:2:8, 0:8:8,
Elapsed time: 0

思考题 2: 小明拿着一个 7 升和一个 11 升的空容器来到水龙头前，他如何才能带回 6 升水呢？

```
public interface Pair { int tag1(); int tag2(); }
public class BMatrix implements Set { private boolean[][] set;
   public BMatrix(int row, int column) { set=new boolean[row][column]; }
   public int size(){ int ans=0;
      for (int i=0; i<set.length; ++i)
         for (int j=0; j<set[i].length; ++j)
            if (set[i][j]) ++ans; return ans; }
   public boolean isEmpty(){
      for (int i=0; i<set.length; ++i)for (int j=0; j<set[i].length; ++j)
         if (set[i][j]) return false; return true; }
   public boolean contains(Object o){
      int k1=((Pair)o).tag1(), k2=((Pair)o).tag2();
      if (k1>=0 && k1<set.length&& k2>=0 && k2<set[k1].length)
         return set[k1][k2]; else return false; }
   public Iterator iterator(){ return new Iterator(){
      public boolean hasNext(){ return false; }
      public Object next(){ return null; }
      public void remove(){} }; }
   public Object[] toArray(){  return null; }
   public Object[] toArray(Object a[]){  return null; }
   public boolean add(Object o){ int k1=((Pair)o).tag1(), k2=((Pair)o).tag2();
      if (k1>=0 && k1<set.length&& k2>=0 && k2<set[k1].length)
      return set[k1][k2]=true; else return false; }
   public boolean remove(Object o){
      int k1=((Pair)o).tag1(), k2=((Pair)o).tag2();
      if (k1>=0 && k1<set.length && k2>=0 && k2<set[k1].length){
         set[k1][k2]=false; return true; }
      else return false; }
   public void clear(){
      for (int i=0; i<set.length; ++i) Arrays.fill(set[i], false); }
   public boolean containsAll(Collection c){ return false; }
   public boolean addAll(Collection c){ return false; }
   public boolean removeAll(Collection c){ return false; }
   public boolean retainAll(Collection c){ return false; }
}
public class Cup2 implements Node, Pair {
   static int a, b, c; static PrintWriter out; int i, j;
   public static void setcup(int x, int y, int z){ a=x; b=y; c=z; }
   public Cup2(int x, int y){ i=x; j=y; }
   public boolean target(){ return i==c || j==c; }
```

```
    public int subnum(){ return 6; }
    public Node down(int x){
        if (x==0) return i==0 ? null : new Cup2(0, j);
        if (x==1) return j==0 ? null : new Cup2(i, 0);
        if (x==2) return i==a ? null : new Cup2(a, j);
        if (x==3) return j==b ? null : new Cup2(i, b);
        if (x==4) return i==0 || j==b ? null : i<b-j ? new Cup2(0, j+i) :
                    new Cup2(i-b+j, b);
        if (x==5) return j==0 || i==a ? null : j<a-i ? new Cup2(i+j, 0) :
                    new Cup2(a, j-a+i);
        return null;
    }
    public void output(){
        out.print(i+":"+j+",   ");
    }
    public int tag1(){ return i; }
    public int tag2(){ return j; }
    public static void main(String[] args){PrintWriter fout=null;
        try{ fout=new PrintWriter(new FileWriter("out20.txt")); }
        catch(IOException e){ System.out.println("Error!"); }
        long begin=System.currentTimeMillis(), end=0; int m=7, n=11, t=6;
        Cup2.out=fout; Cup2.setcup(m, n, t);
        fout.println("Depth First 1:");
        GraphSearch s=new GraphSearch(new Cup2(0, 0), 1, new BMatrix(m+1, n+1));
        s.depthfirst();
        fout.println("Depth First 2:");
        s=new GraphSearch(new Cup2(0, 0), 1, new BMatrix(m+1, n+1));
        s.depthfirst1();
        fout.println("Wide First:");
        s=new GraphSearch(new Cup2(0, 0), 1, new BMatrix(m+1, n+1));
        s.widefirst();
        end=System.currentTimeMillis();
        fout.println("Elapsed time: "+(end-begin));
        fout.close(); }
}
```

Depth First 1:

0:0, 7:0, 7:11, 0:11, 7:4, 0:4, 4:0, 4:11, 7:8, 0:8, 7:1, 0:1, 1:0, 1:11, 7:5, 0:5, 5:0, 5:11, 7:9, 0:9, 7:2, 0:2, 2:0, 2:11, 7:6,

Depth First 2:

0:0, 0:11, 7:4, 0:4, 4:0, 4:11, 7:8, 0:8, 7:1, 0:1, 1:0, 1:11, 7:5, 0:5, 5:0, 5:11, 7:9, 0:9, 7:2, 0:2, 2:0, 2:11, 7:6,

Wide First:

0:0, 7:0, 0:7, 7:7, 3:11, 3:0, 0:3, 7:3, 0:10, 7:10, 6:11,

Elapsed time: 0

思考题 3: 小明拿着一个 7 升,一个 13 升和一个 19 升的空容器来到水龙头前,他如何才能在两个容器中各盛 10 升水呢?

```
public class Cup3  implements Node, Comparable {
```

```
static int len;  static int[] vol; static PrintWriter out; int[] data;
public Cup3(int[] x){ data=x; }
public int compareTo(Object o){ Cup3 obj=(Cup3)o; int i=0;
   while (i<len && data[i]==obj.data[i]) ++i;
   return i<len ? data[i]-obj.data[i] : 0;
}
public boolean target(){ return data[1]==10 && data[2]==10; }
public int subnum(){ return len*(len-1)+2*len; }
public Node down(int i){ Cup3 ans=null; boolean q1=true, q2=true, q3=true;
   int type=0, x=0, y=0, a=0, b=0;
   if (i>=len*(len-1)+len){ type=1; x=i-len*(len-1)-len;
      q1=data[x]!=0;
   }else if (i>=len*(len-1)){ type=2; y=i-len*(len-1);
      q2=data[y]!=vol[y];
   }else{ type=3; a=i/(len-1); b=i%(len-1); if (b>=a) ++b;
      q3=data[a]!=0 && data[b]!=vol[b];
   }
   if (q1 && q2 && q3){
      int[] d=new int[len]; System.arraycopy(data, 0, d, 0, len);
      if (type==1) d[x]=0;
      if (type==2) d[y]=vol[y];
      if (type==3){
         if (data[a]<vol[b]-data[b]){ d[a]=0; d[b]=data[a]+data[b]; }
         else { d[a]=data[a]-vol[b]+data[b]; d[b]=vol[b]; }
      }
      ans=new Cup3(d);
   }
   return ans;
}
public void output(){
   for (int i=0; i<len-1; ++i) out.print(data[i]+":");
   out.print(data[len-1]+",   ");
}
public static void main(String[] args){PrintWriter fout=null;
   try{ fout=new PrintWriter(new FileWriter("out30.txt")); }
   catch(IOException e){ System.out.println("Error!"); }
   long begin=System.currentTimeMillis(), end=0; int m=7, n=11, t=6;
   Cup3.out=fout; final int[] v={7, 13, 19};
   int k=3; Cup3.len=k; Cup3.vol=v;
   fout.println("Depth First 1:");
   GraphSearch s=new GraphSearch(new Cup3(new int[k]), 1, new TreeSet());
   s.depthfirst();
    fout.println("Depth First 2:");
   s=new GraphSearch(new Cup3(new int[k]), 1, new TreeSet());
   s.depthfirst1();
   fout.println("Wide First:");
```

```
        s=new GraphSearch(new Cup3(new int[k]), 1, new TreeSet());
        s.widefirst();
        end=System.currentTimeMillis();
        fout.println("Elapsed time: "+(end-begin));
        fout.close(); }
}
```

Depth First 1:

0:0:0, 7:0:0, 0:7:0, 0:0:7, 7:0:7, 0:7:7, 0:0:14, 0:13:1, 7:6:1, 0:6:8, 6:0:8, 6:8:0, 1:13:0, 7:7:0, 7:13:0, 0:13:7, 7:6:7, 0:6:14, 6:0:14, 1:0:19, 0:1:19, 7:1:12, 0:8:12, 7:8:5, 2:13:5, 2:0:18, 0:2:18, 7:2:11, 0:9:11, 7:9:4, 3:13:4, 3:0:17, 0:3:17, 7:3:10, 0:10:10,

Depth First 2:

0:0:0, 0:0:19, 0:13:19, 7:13:19, 7:13:0, 7:0:13, 0:0:13, 0:13:13, 0:7:19, 0:7:0, 7:7:0, 7:0:7, 7:13:7, 7:1:19, 7:1:0, 0:1:0, 0:0:1, 0:13:1, 7:13:1, 7:0:14, 7:13:14, 7:8:19, 7:8:0, 7:0:8, 7:13:8, 7:2:19, 7:2:0, 0:2:0, 0:0:2, 0:13:2, 7:13:2, 7:0:15, 7:13:15, 7:9:19, 7:9:0, 7:0:9, 7:13:9, 7:3:19, 7:3:0, 0:3:0, 0:0:3, 0:13:3, 7:13:3, 7:0:16, 7:13:16, 7:10:19, 7:10:0, 7:0:10, 7:13:10, 7:4:19, 7:4:0, 0:4:0, 0:0:4, 0:13:4, 7:13:4, 7:0:17, 7:13:17, 7:11:19, 7:11:0, 7:0:11, 7:13:11, 7:5:19, 7:5:0, 0:5:0, 0:0:5, 0:13:5, 7:13:5, 7:0:18, 7:13:18, 7:12:19, 7:12:0, 0:12:7, 7:12:7, 0:12:14, 7:12:14, 2:12:19, 2:12:0, 2:0:12, 2:13:12, 2:6:19, 2:6:0, 7:6:0, 0:6:7, 7:6:7, 0:6:14, 6:0:14, 6:13:1, 6:0:1, 6:1:0, 6:1:19, 7:1:18, 0:1:18, 7:1:11, 0:1:11, 7:1:4, 0:1:4, 4:1:0, 4:1:19, 4:13:7, 4:0:7, 4:7:0, 4:7:19, 4:13:13, 4:0:13, 0:4:13, 7:4:13, 1:4:19, 1:4:0, 0:4:1, 7:4:1, 0:4:8, 7:4:8, 0:4:15, 7:4:15, 3:4:19, 3:4:0, 3:0:4, 3:13:4, 3:0:17, 3:13:17, 3:11:19, 3:11:0, 3:0:11, 3:13:11, 3:5:19, 3:5:0, 3:0:5, 3:13:5, 3:0:18, 3:13:18, 3:12:19, 3:12:0, 3:0:12, 3:13:12, 3:6:19, 3:6:0, 3:0:6, 3:13:6, 7:9:6, 0:9:6, 6:9:0, 6:9:19, 6:13:15, 6:0:15, 6:13:2, 6:0:2, 6:2:0, 6:2:19, 6:13:8, 6:0:8, 6:8:0, 0:8:6, 7:8:6, 0:8:13, 7:8:13, 1:8:19, 1:8:0, 1:0:8, 1:13:8, 1:2:19, 1:2:0, 1:0:2, 1:13:2, 1:0:15, 1:13:15, 1:9:19, 1:9:0, 1:0:9, 1:13:9, 1:3:19, 1:3:0, 1:0:3, 1:13:3, 1:0:16, 1:13:16, 1:10:19, 1:10:0, 1:0:10, 0:1:10, 7:1:10, 0:1:17, 7:1:17, 5:1:19, 5:1:0, 5:0:1, 5:13:1, 5:0:14, 5:13:14, 5:8:19, 5:8:0, 5:0:8, 5:13:8, 5:2:19, 5:2:0, 5:0:2, 5:13:2, 5:0:15, 5:13:15, 5:9:19, 5:9:0, 5:0:9, 5:13:9, 5:3:19, 5:3:0, 5:0:3, 5:13:3, 5:0:16, 5:13:16, 5:10:19, 5:10:0, 5:0:10, 5:13:10, 5:4:19, 5:4:0, 5:0:4, 5:13:4, 5:0:17, 5:13:17, 5:11:19, 5:11:0, 5:0:11, 5:13:11, 5:5:19, 5:5:0, 5:0:5, 5:13:5, 5:0:18, 5:13:18, 5:12:19, 5:12:0, 5:0:12, 5:13:12, 5:6:19, 5:6:0, 5:0:6, 5:13:6, 7:11:6, 0:11:6, 6:11:0, 6:11:19, 6:13:17, 6:0:17, 6:13:4, 6:0:4, 6:4:0, 6:4:19, 6:13:10, 6:0:10, 6:10:0, 6:10:19, 6:13:16, 6:0:16, 6:13:3, 6:0:3, 6:3:0, 6:3:19, 6:13:9, 7:12:9, 0:12:16, 7:5:16, 4:5:19, 4:5:0, 4:0:5, 4:13:5, 4:0:18, 0:4:18, 7:4:11, 0:4:11, 7:4:4, 0:4:4, 4:4:0, 4:4:19, 4:13:10, 7:10:10,

Wide First:

0:0:0, 7:0:0, 0:7:0, 7:7:0, 0:7:7, 7:7:7, 0:7:14, 0:2:19, 7:2:12, 0:9:12, 7:9:5, 3:13:5, 3:0:5, 0:3:5, 5:3:0, 5:3:19, 7:3:17, 0:10:17, 7:10:10,

Elapsed time: 31

思考题 4: 小明有两个 100 毫升的容器中装满了牛奶，另外有一个 40 毫升和一个 70 毫升的空碗，如何用它们在两个碗中各盛 30 毫升的牛奶，并且一点儿牛奶也不泼掉呢？

```
public class Cup5  implements Node, Comparable {
   static int len;  static int[] vol; static PrintWriter out; int[] data;
   public Cup5(int[] x){ data=x; }
   public int compareTo(Object o){ Cup5 obj=(Cup5)o; int i=0;
```

```
      while (i<len && data[i]==obj.data[i]) ++i;
      return i<len ? data[i]-obj.data[i] : 0;
   }
   public boolean target(){ return data[0]==30 && data[1]==30; }
   public int subnum(){ return len*(len-1); }
   public Node down(int i){ Cup5 ans=null; boolean q=true;
      int a=i/(len-1), b=i%(len-1); if (b>=a) ++b;
         q=data[a]!=0 && data[b]!=vol[b];
      if (q){
         int[] d=new int[len]; System.arraycopy(data, 0, d, 0, len);
         if (data[a]<vol[b]-data[b]){ d[a]=0; d[b]=data[a]+data[b]; }
         else { d[a]=data[a]-vol[b]+data[b]; d[b]=vol[b]; }
         ans=new Cup5(d);
      }
      return ans;
   }
   public void output(){
      for (int i=0; i<len-1; ++i) out.print(data[i]+":");
      out.print(data[len-1]+",   ");
   }
   public static void main(String[] args){PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("out50.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(), end=0;
      Cup5.out=fout; final int[] v={40, 70, 100, 100};
      int k=4; Cup5.len=k; Cup5.vol=v;
      fout.println("Depth First 1:");
      final int [] a={0, 0, 100, 100};
      GraphSearch s=new GraphSearch(new Cup5(a), 1, new TreeSet());
      s.depthfirst(); fout.println("Depth First 2:");
      s=new GraphSearch(new Cup5(a), 1, new TreeSet());
      s.depthfirst1();  fout.println("Wide First:");
      s=new GraphSearch(new Cup5(a), 1, new TreeSet());
      s.widefirst(); end=System.currentTimeMillis();
      fout.println("Elapsed time: "+(end-begin));
      fout.close(); }
}
```

```
Depth First 1:
0:0:100:100,   40:0:60:100,   0:40:60:100,   40:40:20:100,   10:70:20:100,   0:70:30:100,
40:30:30:100,   0:30:70:100,   30:0:70:100,   30:70:0:100,   40:60:0:100,   0:60:40:100,
40:20:40:100,   0:20:80:100,   20:0:80:100,   20:70:10:100,   40:50:10:100,   0:50:50:100,
40:10:50:100,   0:10:90:100,   10:0:90:100,   40:0:90:70,   0:40:90:70,   0:30:100:70,
30:0:100:70,   30:70:30:70,   0:70:60:70,   40:30:60:70,   10:30:60:100,   10:70:60:60,
0:70:70:60,   40:30:70:60,   10:30:100:60,   0:40:100:60,   40:0:100:60,   40:70:30:60,
40:70:0:90,   0:70:40:90,   40:30:40:90,   0:30:80:90,   30:0:80:90,   10:0:100:90,
0:10:100:90,   40:10:60:90,   0:50:60:90,   40:50:20:90,   20:70:20:90,   20:0:90:90,
```

```
0:20:90:90,  40:20:50:90,  0:60:50:90,  40:60:10:90,  30:70:10:90,  30:60:10:100,
30:60:100:10,  20:70:100:10,  0:70:100:30,  40:30:100:30,  40:70:60:30,  40:70:90:0,
30:70:100:0,  40:60:100:0,  0:60:100:40,  40:20:100:40,  0:20:100:80,  20:0:100:80,
40:0:80:80,  0:40:80:80,  40:40:40:80,  10:70:40:80,  0:70:50:80,  40:30:50:80,
0:30:90:80,  30:0:90:80,  30:70:20:80,  40:60:20:80,  0:60:60:80,  40:20:60:80,
20:20:60:100,  20:70:60:50,  0:70:80:50,  40:30:80:50,  20:30:100:50,  20:30:50:100,
20:70:50:60,  40:50:50:60,  0:50:90:60,  40:10:90:60,  30:10:100:60,  30:70:40:60,
40:60:40:60,  0:60:80:60,  40:60:80:20,  30:70:80:20,  10:70:100:20,  40:40:100:20,
40:70:70:20,  40:0:70:90,  0:40:70:90,  40:40:30:90,  10:70:30:90,  10:60:30:100,
40:60:30:70,  0:60:70:70,  40:20:70:70,  10:20:100:70,  10:70:50:70,  40:40:50:70,
40:70:50:40,  0:70:90:40,  40:30:90:40,  30:30:100:40,
Depth First 2:
0:0:100:100,  0:70:100:30,  30:70:100:0,  30:70:0:100,  40:70:0:90,  40:0:70:90,
40:70:70:20,  40:40:100:20,  40:40:20:100,  40:70:20:70,  40:0:90:70,  10:0:90:100,
10:70:90:30,  10:60:100:30,  10:60:30:100,  40:60:30:70,  0:60:70:70,  40:20:70:70,
40:20:100:40,  40:70:50:40,  40:10:50:100,  40:10:100:50,  40:70:40:50,  0:70:80:50,
0:20:80:100,  40:20:80:60,  20:20:100:60,  20:20:60:100,  20:70:60:50,  20:30:100:50,
20:30:50:100,  40:30:50:80,  0:30:90:80,  40:30:90:40,  30:30:100:40,
Wide First:
0:0:100:100,  40:0:60:100,  40:60:0:100,  0:60:40:100,  40:60:40:60,  30:70:40:60,
30:30:40:100,
Elapsed time: 16
```

思考题 5：布线问题(例 1).

```
public class Lining1 implements Node, Pair {
   static final int[][] move={{0, 1}, {1, 0}, {0, -1}, {-1, 0}};
   static int m, n, num;  static PrintWriter out;
   static boolean[][] board, an;  int x, y;
   public Lining1(){ x=0; y=0; }
   public Lining1(int i, int j){ x=i; y=j; }
   public boolean target() { return x==m-1 && y==n-1; }
   public int subnum(){ return 4; }
   public Node down(int i){
      int x1=x+move[i][0], y1=y+move[i][1];
      return x1>=0 && x1<m && y1>=0 && y1<n && !board[x1][y1] ?
            new Lining1(x1, y1) : null;
   }
   public void output(){ an[x][y]=true;  ++num;
   }
   public int tag1(){ return x; }
   public int tag2(){ return y; }
   public static void outputans(){
      for (int i=0; i<m; ++i){
         for (int j=0; j<n; ++j)
            if (board[i][j]) out.print(" * ");
            else if (an[i][j]) out.print(" @ ");
```

```
            else out.print("   ");
         out.println();
      }
      out.println(num);
   }
   public static void main(String[] args){PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("outlabel-200-200.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(), end=0;
      Lining1.out=fout; int k=4;
      Random r=new Random(100); boolean[] map=new boolean[k]; map[0]=true;
      int row=200, column=200; boolean[][] bd=new boolean[row][column];
      boolean[][] ans=new boolean[row][column];
      for (int i=0; i<row; ++i) for (int j=0; j<column; ++j)
         bd[i][j]=map[r.nextInt(k)];
      bd[0][0]=false; bd[row-1][column-1]=false;
      Set set=new BMatrix(row, column);
      Lining1.m=row; Lining1.n=column; Lining1.board=bd; Lining1.an=ans;
      //fout.println("Depth First 1:");
      //GraphSearch s=new GraphSearch(new Lining1(), 1, set);
      //s.depthfirst(); outputans();
      //fout.println("Depth First 2:");
      //GraphSearch s=new GraphSearch(new Lining1(), 1, set);
      //s.depthfirst1(); outputans();
      //fout.println("Wide First:");
      //GraphSearch s=new GraphSearch(new Lining1(), 1, set);
      //s.widefirst(); outputans();
      fout.println("Label Search:");
      LabelSearch s=new LabelSearch(new Lining1(), 1, set);
      s.depthfirst(); outputans();
      end=System.currentTimeMillis();
      fout.println("Elapsed time: "+(end-begin));
      fout.close(); }
}
```

Depth First 1:

```
+ + # * * # * * * * * # # * * # * * * *
* + + + + # * * * # * * * # * * # * # #
* * * * + # + + + # # * # + + + + + + +
* * * * + + + # + + + + + + # + + + + +
# * * # * * # * * # * * # # * + # * # #
* * * * * # * * * * * * # * * + + # * *
* * # # * # * * # # * * * * * * + + + +
* * * * * * * * * # # # * * * * * * * +
* * # * * * # # * * * # * * * * * # * +
* # * * # * * * # * * * * * * # # * * +
```

41

Elapsed time: 0

Depth First 2:

```
+ * # * * # * * + + + # # * * # * * * *
+ * * * * # * * + # + + * # * * # * # #
+ * * * * # * * + # # + # * * * * * * *
+ + * * * * * # + * + + * * # * * * * *
# + * # * * # + + # + * # # * * # * # #
+ + * * * # + + * * + * # * + + + # * *
+ * # # * # + * # # + + + * + * + * * *
+ + + + * + + * * # # # + * + * + + + *
* * # + + + # # * * * # + * + * * # + *
* # * * # * * * # * * * + + + # # * + +
```

57

Elapsed time: 15

Wide First:

```
+ + # * * # * * * * * # # * * # * * * *
* + + + + # * * * # * * * # * * # * # #
* * * * + # + + + # # * # * * * * * * *
* * * * + + + # + + + + * * # * * * * *
# * * # * * # * * # * + # # * * # * # #
* * * * * # * * * * * + # * * * * # * *
* * # # * # * * # # * + + + + + + + + +
* * * * * * * * * # # # * * * * * * * +
* * # * * * # # * * * # * * * * * # * +
* # * * # * * * # * * * * * * # # * * +
```

31

Elapsed time: 0

思考题 6: 用完全 2 叉树表示子集，用算法 1.4.4 和算法 1.1.5 求解子集和数问题。

```
public class TreeSearch { private Link head; private int num, bound;
   public TreeSearch(Node p, int b) { head=new Link(p, null); bound=b; }
   public void depthfirst(){ Node p=head.prob, sub=null; int n=p.subnum();
      if (p.target()){ head.output(true); ++num; }
      for (int i=0; num<bound && i<n; ++i)
         if ((sub=p.down(i))!=null){
            head=new Link(sub, head); depthfirst(); head=head.next; }
   }
   public boolean depthfirst1(){ LinkedList queue=new LinkedList();
      Link cur=null; queue.addFirst(head);
      hile (num<bound && !queue.isEmpty()){ cur=(Link)queue.removeFirst();
         Node curnode=cur.prob; int m=curnode.subnum();
         for (int i=0; i<m; ++i){ Node subnode=curnode.down(i);
            if (subnode!=null){ Link sub=new Link(subnode, cur);
               queue.addFirst(sub);
               if (subnode.target()){
                  while (sub!=null){
                     sub.prob.output(); sub=sub.next;
```

```
                    }
                ++num; }
            }
        }
        return num!=0;
    }
    public int getnum(){ return num; }
    public boolean widefirst(){ LinkedList queue=new LinkedList();
        Link cur=null; queue.addLast(head);
        while (num<bound && !queue.isEmpty()){ cur=(Link)queue.removeFirst();
            Node curnode=cur.prob; int m=curnode.subnum();
            for (int i=0; i<m; ++i){ Node subnode=curnode.down(i);
                if (subnode!=null){ Link sub=new Link(subnode, cur);
                    queue.addLast(sub);
                    if (subnode.target()){ sub.output(); ++num;  }
                }
            }
        }
        return num!=0;
    }
}
public class BinNode implements Node {
    static int n, sum; static int[] data, rsum; static PrintWriter f;
    int level, cursum; boolean isleft;
    public BinNode(int lev, int cs, boolean il){ level=lev; cursum=cs;
    isleft=il; }
    public boolean target(){ return level==n; }
    public int subnum(){ return level<n ? 2 : 0; }
    public Node down(int i){ BinNode ans=null;
        if (i==0 && cursum+rsum[level+1]>=sum)
            ans=new BinNode(level+1, cursum, true);
        if (i==1 && cursum+data[level]<=sum)
            ans=new BinNode(level+1, cursum+data[level], false);
        return ans;
    }
    public void output(){
        if (!isleft) f.print(data[level-1]+", ");
    }
    public static void main(String[] args) {
        PrintWriter fout=null;
        try{ fout=new PrintWriter(new FileWriter("outbinnode50b.txt")); }
        catch(IOException e){ System.out.println("Error!"); }
        long begin=System.currentTimeMillis(); int m=50, n=1000, sum=m*n/4;
        int[] d=new int[m]; int[] rightsum=new int[m+1]; Random rm=new Random(m);
        for (int i=0; i<m; ++i) d[i]=rm.nextInt(n); //Arrays.sort(d);
```

```
        //for (int i=0, j=m-1; i<j; ++i, --j) { int t=d[i]; d[i]=d[j]; d[j]=t; }
        for (int i=m-1; i>=0; --i) rightsum[i]=d[i]+rightsum[i+1];
        fout.print("The elements of the set are: ");
        for (int i=0; i<m; ++i)  fout.print(d[i]+", ");
        fout.println();
        BinNode.n=m; BinNode.data=d; BinNode.sum=sum;
       BinNode.rsum=rightsum; BinNode.f=fout;
       TreeSearch s=new TreeSearch(new BinNode(0, 0, true), Integer.MAX_VALUE);
       fout.println(s.getnum());
        //TreeSearch s=new TreeSearch(new BinNode(0, 0, true), 1);
        //s.depthfirst();
        //s.depthfirst1();
        long end=System.currentTimeMillis();
        fout.println(m+", "+sum+":  Elapsed time: "+(end-begin));
        fout.close();
    }
}
Binnode, depthfirst:
The elements of the set are: 606, 868, 765, 604, 136, 298, 186, 282, 456, 847, 831, 904,
3, 316, 647, 560, 34, 446, 109, 403, 394, 815, 101, 386, 169, 771, 680, 142, 275, 344,
252910
30, 7500:  Elapsed time: 10639
ordered:
The elements of the set are: 3, 34, 101, 109, 136, 142, 169, 186, 275, 282, 298, 316, 344,
386, 394, 403, 446, 456, 560, 604, 606, 647, 680, 765, 771, 815, 831, 847, 868, 904,
252910
30, 7500:  Elapsed time: 25756
Reverse ordered:
The elements of the set are: 904, 868, 847, 831, 815, 771, 765, 680, 647, 606, 604, 560,
456, 446, 403, 394, 386, 344, 316, 298, 282, 275, 186, 169, 142, 136, 109, 101, 34, 3,
252910
30, 7500:  Elapsed time: 1310

Binnode, depthfirst1:
The elements of the set are: 606, 868, 765, 604, 136, 298, 186, 282, 456, 847, 831, 904,
3, 316, 647, 560, 34, 446, 109, 403, 394, 815, 101, 386, 169, 771, 680, 142, 275, 344,
252910
30, 7500:  Elapsed time: 15631
Ordered:
The elements of the set are: 3, 34, 101, 109, 136, 142, 169, 186, 275, 282, 298, 316, 344,
386, 394, 403, 446, 456, 560, 604, 606, 647, 680, 765, 771, 815, 831, 847, 868, 904,
252910
30, 7500:  Elapsed time: 38985
Reverse ordered:
The elements of the set are: 904, 868, 847, 831, 815, 771, 765, 680, 647, 606, 604, 560,
456, 446, 403, 394, 386, 344, 316, 298, 282, 275, 186, 169, 142, 136, 109, 101, 34, 3,
```

```
252910
30, 7500:  Elapsed time: 2028
```

思考题 7: 用递增排列树表示子集,用算法 1.4.4 和算法 1.1.5 求解子集和数问题,用算法 1.4.6 寻找包含元素最少的解。

```
public class ComNode implements Node {
   static int n, sum; static int[] data, rsum; static PrintWriter f;
   public int curnode; int cursum;
   public ComNode(int cn, int cs){ curnode=cn; cursum=cs; }
   public boolean target(){ return cursum==sum; }
   public int subnum(){ return n-curnode-1; }
   public Node down(int i){ ComNode ans=null;
      int cn=curnode+i+1, cs=cursum+data[cn];
      if (cs<=sum && cs+rsum[cn+1]>=sum) ans=new ComNode(cn, cs);
      return ans;
   }
   public void output(){
      if (curnode>=0) f.print(data[curnode]+", ");
   }
   public static void main(String[] args) {
      PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("outcomnode50b.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(); int m=50, n=1000, sum=m*n/4;
      int[] d=new int[m]; int[] rightsum=new int[m+1]; Random rm=new Random(m);
      for (int i=0; i<m; ++i) d[i]=rm.nextInt(n); //Arrays.sort(d);
      //for (int i=0, j=m-1; i<j; ++i, --j) { int t=d[i]; d[i]=d[j]; d[j]=t; }
      for (int i=m-1; i>=0; --i) rightsum[i]=d[i]+rightsum[i+1];
      fout.print("The elements of the set are: ");
      for (int i=0; i<m; ++i)  fout.print(d[i]+", ");
      fout.println();
      ComNode.n=m; ComNode.data=d; ComNode.sum=sum;
      ComNode.rsum=rightsum; ComNode.f=fout;
      TreeSearch s=new TreeSearch(new ComNode(-1, 0), 1);
      //s.depthfirst();
      s.depthfirst1();
      //s.widefirst();
      long end=System.currentTimeMillis();
      fout.println(m+", "+sum+":  Elapsed time: "+(end-begin));
      fout.close();
   }
}
```

```
Comnode, depthfirst:
The elements of the set are: 606, 868, 765, 604, 136, 298, 186, 282, 456, 847, 831, 904,
3, 316, 647, 560, 34, 446, 109, 403, 394, 815, 101, 386, 169, 771, 680, 142, 275, 344,
252910
30, 7500:  Elapsed time: 6505
```

ordered:
The elements of the set are: 3, 34, 101, 109, 136, 142, 169, 186, 275, 282, 298, 316, 344, 386, 394, 403, 446, 456, 560, 604, 606, 647, 680, 765, 771, 815, 831, 847, 868, 904, 252910
30, 7500: Elapsed time: 15569
Reverse ordered:
The elements of the set are: 904, 868, 847, 831, 815, 771, 765, 680, 647, 606, 604, 560, 456, 446, 403, 394, 386, 344, 316, 298, 282, 275, 186, 169, 142, 136, 109, 101, 34, 3, 252910
30, 7500: Elapsed time: 983

The elements of the set are: 606, 868, 765, 604, 136, 298, 186, 282, 456, 847, 831, 904, 3, 316, 647, 560, 34, 446, 109, 403, 394, 815, 101, 386, 169, 771, 680, 142, 275, 344,
606, 868, 765, 604, 136, 298, 186, 282, 456, 847, 831, 904, 3, 34, 680,
30, 7500: Elapsed time: 0

Comnode, depthfirst1:
The elements of the set are: 606, 868, 765, 604, 136, 298, 186, 282, 456, 847, 831, 904, 3, 316, 647, 560, 34, 446, 109, 403, 394, 815, 101, 386, 169, 771, 680, 142, 275, 344, 252910
30, 7500: Elapsed time: 10062
ordered:
The elements of the set are: 3, 34, 101, 109, 136, 142, 169, 186, 275, 282, 298, 316, 344, 386, 394, 403, 446, 456, 560, 604, 606, 647, 680, 765, 771, 815, 831, 847, 868, 904, 252910
30, 7500: Elapsed time: 25382
Reverse ordered:
The elements of the set are: 904, 868, 847, 831, 815, 771, 765, 680, 647, 606, 604, 560, 456, 446, 403, 394, 386, 344, 316, 298, 282, 275, 186, 169, 142, 136, 109, 101, 34, 3, 252910
30, 7500: Elapsed time: 1451

The elements of the set are: 904, 868, 847, 831, 815, 771, 765, 680, 647, 606, 604, 560, 456, 446, 403, 394, 386, 344, 316, 298, 282, 275, 186, 169, 142, 136, 109, 101, 34, 3,
765, 680, 647, 606, 604, 456, 446, 403, 394, 386, 344, 316, 298, 275, 186, 169, 142, 136, 109, 101, 34, 3,
30, 7500: Elapsed time: 16

Comnode, widefirst:
The elements of the set are: 904, 868, 847, 831, 815, 771, 765, 680, 647, 606, 604, 560, 456, 446, 403, 394, 386, 344, 316, 298, 282, 275, 186, 169, 142, 136, 109, 101, 34, 3,
904, 868, 847, 831, 815, 771, 765, 647, 606, 446,
30, 7500: Elapsed time: 31

40，内存不足!

思考题 8：用算法 1.4.7 求解跳马问题，从一个角开始，最后可以不跳回起点。

```
public class LabelSearch {
   private Link head; private int num, bound; private Set nodeset;
   public LabelSearch(Node p, int b, Set s) {
      head=new Link(p, null); bound=b; nodeset=s; nodeset.add(p); }
   public void depthfirst(){ Node p=head.prob, sub=null; int n=p.subnum();
      if (p.target()){ head.output(true); ++num; }
      for (int i=0; num<bound && i<n; ++i)
         if ((sub=p.down(i))!=null && !nodeset.contains(sub)){
            head=new Link(sub, head); nodeset.add(sub); depthfirst();
            head=head.next; nodeset.remove(sub); } }
   public int getnum(){ return num; }
}
public class Knight1 implements Node, Pair {
   static int m, n; static PrintWriter out;
   static int[][]move=
      {{2,1},{1,2},{-1,2},{-2,1},{-2,-1},{-1,-2},{1,-2},{2,-1}};
   static int[][] board;  int level, x, y;
   public Knight1(){ level=1; }
   public Knight1(int xx, int yy){ level=1; x=xx; y=yy; }
   public Knight1(int lev, int xx, int yy){ level=lev; x=xx; y=yy; }
   public boolean target(){ return level==m*n; }
   public int subnum(){ return level<m*n ? 8 : 0; }
   public Node down(int i){ int x1=x+move[i][0], y1=y+move[i][1];
      return x1>=0 && x1<m && y1>=0 && y1<n ?
         new Knight1(level+1, x1, y1) : null;  }
   public void output(){ board[x][y]=level;
      if (target()){
         for (int i=0; i<m; ++i){
            for (int j=0; j<n; ++j){
               if (board[i][j]<10) out.print("0");
               out.print(board[i][j]+", "); }
            out.println(); }
         out.println(); }
   }
   public int tag1(){ return x; }
   public int tag2(){ return y; }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("node1-66.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int m=6, n=6; BMatrix mnset=new BMatrix(m, n);
      int[][] board=new int[m][n];
      Knight1.m=m; Knight1.n=n; Knight1.out=fout; Knight1.board=board;
      long begin=System.currentTimeMillis();
      LabelSearch x=new LabelSearch(new Knight1(0, 0), 1, mnset);
      x.depthfirst(); if (x.getnum()==0) fout.println("No Solution!");
```

```
        long end=System.currentTimeMillis();
        fout.println(m+", "+n+":  Elapsed time: "+(end-begin));
        fout.close();
    }
}
01, 16, 07, 26, 11, 14,
34, 25, 12, 15, 06, 27,
17, 02, 33, 08, 13, 10,
32, 35, 24, 21, 28, 05,
23, 18, 03, 30, 09, 20,
36, 31, 22, 19, 04, 29,

6, 6:  Elapsed time: 63

01, 38, 31, 08, 19, 36, 15,
32, 29, 20, 37, 16, 07, 18,
39, 02, 33, 30, 09, 14, 35,
28, 25, 40, 21, 34, 17, 06,
41, 22, 03, 26, 45, 10, 13,
24, 27, 48, 43, 12, 05, 46,
49, 42, 23, 04, 47, 44, 11,

7, 7:  Elapsed time: 1857

01, 60, 39, 34, 31, 18, 09, 64,
38, 35, 32, 61, 10, 63, 30, 17,
59, 02, 37, 40, 33, 28, 19, 08,
36, 49, 42, 27, 62, 11, 16, 29,
43, 58, 03, 50, 41, 24, 07, 20,
48, 51, 46, 55, 26, 21, 12, 15,
57, 44, 53, 04, 23, 14, 25, 06,
52, 47, 56, 45, 54, 05, 22, 13,

8, 8:  Elapsed time: 2121
```

思考题 9：用算法 1.4.7 求解跳马问题，最后要求跳回起点。

```
public class Knight1a extends Knight1 {
   public Knight1a(){ super(2, 2, 1); board[0][0]=1; board[1][2]=m*n; }
   public Knight1a(int lev, int xx, int yy){ super(lev, xx, yy); }
   public boolean target(){ return level==m*n-1 && edge(x, y, 1, 2); }
   public Node down(int i){ int x1=x+move[i][0], y1=y+move[i][1];
      return x1>=0 && x1<m && y1>=0 && y1<n && !(x1==1 && y1==2) ?
            new Knight1a(level+1, x1, y1) : null;  }
   public boolean edge(int x1, int y1, int x2, int y2){
      return Math.abs(x1-x2)==2 && Math.abs(y1-y2)==1 ||
         Math.abs(x1-x2)==1 && Math.abs(y1-y2)==2; }
```

```
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("node1a-66.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int m=6, n=6; BMatrix mnset=new BMatrix(m, n);
      int[][] board=new int[m][n];
      Knight1.m=m; Knight1.n=n; Knight1.out=fout; Knight1.board=board;
      long begin=System.currentTimeMillis();
      LabelSearch x=new LabelSearch(new Knight1a(), 1, mnset);
      x.depthfirst(); if (x.getnum()==0) fout.println("No Solution!");
      long end=System.currentTimeMillis();
      fout.println(m+", "+n+":  Elapsed time: "+(end-begin));
      fout.close();
   }
}
01, 14, 21, 30, 35, 12,
22, 29, 36, 13, 20, 31,
15, 02, 23, 32, 11, 34,
28, 05, 08, 17, 24, 19,
07, 16, 03, 26, 33, 10,
04, 27, 06, 09, 18, 25,

6, 6:  Elapsed time: 64865
```

思考题 10：用算法 1.4.7 求解布线问题，跟思考题 5 的程序比较运行时间。

```
Label Search:
 +  +  #  *  *  #  *  *  *  *  *  #  #  *  *  #  *  *  *  *
 *  +  +  +  +  #  *  *  *  #  *  *  *  #  *  *  #  *  #  #
 *  *  *  *  +  #  +  +  +  #  #  *  #  +  +  +  +  +  +  +
 *  *  *  *  +  +  +  #  +  +  +  +  +  +  #  +  +  +  +  +
 #  *  *  #  *  *  #  *  *  #  *  *  #  #  *  +  #  *  #  #
 *  *  *  *  *  #  *  *  *  *  *  *  #  *  *  +  +  #  *  *
 *  *  #  #  *  #  *  *  #  #  *  *  *  *  *  *  +  +  +  +
 *  *  *  *  *  *  *  *  *  #  #  #  *  *  *  *  *  *  *  +
 *  *  #  *  *  *  #  #  *  *  *  #  *  *  *  *  *  #  *  +
 *  #  *  *  #  *  *  *  #  *  *  *  *  *  *  #  #  *  *  +
41
Elapsed time: 0

180×200: Depth First 1:
547
Elapsed time: 15
180×200: Depth First 2:
9891
Elapsed time: 47
180×200: Wide First:
```

379
Elapsed time: 47
180×200: Label Search:
547
Elapsed time: 31

188×200: Depth First 1:
583
Elapsed time: 31
188×200: Depth First 2:
10283
Elapsed time: 47
188×200: Wide First:
391
Elapsed time: 47
188×200: Label Search:
583
Elapsed time: 31

190×200: Depth First 1:
堆栈溢出!
190×200: Depth First 2:
0
Elapsed time: 93
190×200: Wide First:
0
Elapsed time: 47
190×200: Label Search:
堆栈溢出!

192×200: Depth First 1:
堆栈溢出!
192×200: Depth First 2:
10485
Elapsed time: 31
192×200: Wide First:
393
Elapsed time: 63
192×200: Label Search:
10 分钟未停机，终止实验。

200×200: Depth First 1:
593
Elapsed time: 31
200×200: Depth First 2:
10947

```
Elapsed time: 47
200×200: Wide First:
399
Elapsed time: 46
200×200: Label Search:
```
10 分钟未停机，终止实验。

思考题 11：用算法 1.4.8 求解子集和数问题。

```
public interface Mono{ boolean target(); int subnum();
     boolean down(int i); void up(); void output(); }
public class BackSearch { Mono prob; int num, bound;
   public BackSearch(Mono p, int b) { prob=p; bound=b; }
   public void depthfirst(){ int n=prob.subnum();
      if (prob.target()){ prob.output(); ++num; }
      for (int i=0; num<bound && i<n; ++i)
         if (prob.down(i)){ depthfirst(); prob.up(); } }
   public int getnum(){ return num; }
}
public class BinMono implements Mono {
   int n, sum, level, cursum; int[] data, rsum; boolean[] set; PrintWriter f;
   public BinMono(int nn, int ss, int[] d, int[] rs, PrintWriter fout){
      n=nn; sum=ss; data=d; rsum=rs; f=fout; set=new boolean[nn]; }
   public boolean target(){ return level==n; }
   public int subnum(){ return level<n ? 2 : 0; }
   public boolean down(int i){
      boolean ans = i==0 && cursum+rsum[level+1]>=sum ||
                    i==1 && cursum+data[level]<=sum;
      if (ans){ set[level] = i!=0;
         if (i!=0) cursum+=data[level];
         ++level; }
      return ans;
   }
   public void up(){ --level; if (set[level]) cursum-=data[level];
   }
   public void output(){
      f.print("The sum of ");
      for (int i=0; i<n; ++i) if (set[i]) f.print(data[i]+", ");
      f.println("is "+sum);
   }
   public static void main(String[] args) {
      PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("BinMono30.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(); int m=30, n=1000, sum=m*n/4;
      int[] d=new int[m]; int[] rightsum=new int[m+1]; Random rm=new Random(m);
      for (int i=0; i<m; ++i) d[i]=rm.nextInt(n); Arrays.sort(d);
```

```
        for (int i=0, j=m-1; i<j; ++i, --j) { int t=d[i]; d[i]=d[j]; d[j]=t; }
        for (int i=m-1; i>=0; --i) rightsum[i]=d[i]+rightsum[i+1];
        fout.print("The elements of the set are: ");
        for (int i=0; i<m; ++i)  fout.print(d[i]+", ");
        fout.println();  BinMono bm=new BinMono(m, sum, d, rightsum, fout);
        BackSearch x=new BackSearch(bm, Integer.MAX_VALUE);
        x.depthfirst(); fout.println(x.getnum());
        long end=System.currentTimeMillis();
        fout.println(m+", "+sum+":  Elapsed time: "+(end-begin));
        fout.close();
    }
}
public class ComMono implements Mono {
    int n, sum, level, cursum; int[] data, rsum, set; PrintWriter f;
    public ComMono(int nn, int ss, int[] d, int[] rs, PrintWriter fout){
        n=nn; sum=ss; data=d; rsum=rs; f=fout; set=new int[nn+1]; set[0]=-1; }
    public boolean target(){ return cursum==sum; }
    public int subnum(){ return n-set[level]-1; }
    public boolean down(int i){ int t=set[level]+i+1;
        boolean ans = cursum+data[t]<=sum && cursum+rsum[t]>=sum;
        if (ans){ set[++level]=t; cursum+=data[t]; }
        return ans;
    }
    public void up(){ cursum-=data[set[level--]];
    }
    public void output(){
        f.print("The sum of ");
        for (int i=1; i<=level; ++i) f.print(data[set[i]]+", ");
        f.println("is "+sum);
    }
    public static void main(String[] args) {
        PrintWriter fout=null;
        try{ fout=new PrintWriter(new FileWriter("ComMono30.txt")); }
        catch(IOException e){ System.out.println("Error!"); }
        long begin=System.currentTimeMillis(); int m=30, n=1000, sum=m*(n/4);
        int[] d=new int[m]; int[] rightsum=new int[m+1]; Random rm=new Random(m);
        for (int i=0; i<m; ++i) d[i]=rm.nextInt(n); Arrays.sort(d);
        for (int i=0, j=m-1; i<j; ++i, --j) { int t=d[i]; d[i]=d[j]; d[j]=t; }
        for (int i=m-1; i>=0; --i) rightsum[i]=d[i]+rightsum[i+1];
        fout.print("The elements of the set are: ");
        for (int i=0; i<m; ++i)  fout.print(d[i]+", ");
        fout.println(); ComMono cm=new ComMono(m, sum, d, rightsum, fout);
        BackSearch x=new BackSearch(cm, Integer.MAX_VALUE);
        x.depthfirst(); fout.println(x.getnum());
        long end=System.currentTimeMillis();
        fout.println(m+", "+sum+":  Elapsed time: "+(end-begin));
```

```
        fout.close();
    }
}
BinMono:
The elements of the set are: 904, 868, 847, 831, 815, 771, 765, 680, 647, 606, 604, 560, 456,
446, 403, 394, 386, 344, 316, 298, 282, 275, 186, 169, 142, 136, 109, 101, 34, 3,
252910
30, 7500:  Elapsed time: 967
ComMono:
The elements of the set are: 904, 868, 847, 831, 815, 771, 765, 680, 647, 606, 604, 560, 456,
446, 403, 394, 386, 344, 316, 298, 282, 275, 186, 169, 142, 136, 109, 101, 34, 3,
252910
30, 7500:  Elapsed time: 968
```

思考题 12：用算法 1.4.8 求解跳马问题。

```
public class Knight2  implements Mono { static PrintWriter out;
   static final int[][] move=
      {{2,1},{1,2},{-1,2},{-2,1},{-2,-1},{-1,-2},{1,-2},{2,-1}};
   int m, n, x, y, level; int[][] board, parent;
   public Knight2(int mm, int nn, int x, int y){
      m=mm; n=nn; this.x=x; this.y=y; level=1;
      board=new int[m][n]; board[x][y]=1; parent=new int[m][n];
   }
   public boolean target(){ return level==m*n; }
   public int subnum(){ return level<m*n ? 8 : 0; }
   public boolean down(int i){ int x1=x+move[i][0], y1=y+move[i][1];
      if (x1>=0 && x1<m && y1>=0 && y1<n && board[x1][y1]==0){
         x=x1; y=y1; board[x][y]=++level; parent[x][y]=i;
         return true; }
      return false;
   }
   public void up(){ int i=parent[x][y]; board[x][y]=0;
      x-=move[i][0]; y-=move[i][1]; --level;
   }
   public void output(){
      for (int i=0; i<m; ++i){
         for (int j=0; j<n; ++j){
            if (board[i][j]>=0 && board[i][j]<10) out.print("0");
            out.print(board[i][j]+", "); }
         out.println(); }
      out.println();
   }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("out-2-8-8.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int m=8, n=8; Knight2.out=fout;
```

```
      long begin=System.currentTimeMillis();
      BackSearch x=new BackSearch(new Knight2(m, n, 0, 0), 1);
      x.depthfirst(); if (x.getnum()==0) fout.println("No Solution!");
      long end=System.currentTimeMillis();
      fout.println(m+", "+n+":  Elapsed time: "+(end-begin));
      fout.close(); }
}
public class Knight2a extends Knight2 {
   static final int[][] cmove={{1,1},{-1,-1}, {-1,1}, {1,-1}};
   int[][] point;
   public Knight2a(int mm, int nn){
      super(mm, nn, mm/2, nn/2); level=1; point=new int[m][n];
      point[2][1]=3; point[1][2]=4; point[m-3][1]=1;
      point[m-2][2]=2; point[m-2][n-3]=3; point[m-3][n-2]=4;
      point[2][n-2]=2; point[1][n-3]=1;
   }
   public boolean target(){
      return level==m*n && edge(x, y, m/2, n/2); }
   public int subnum(){ return level<m*n ? 8 : 0; }
   public boolean down(int i){
      int x1=x+move[i][0], y1=y+move[i][1];
      if (x1>=0 && x1<m && y1>=0 && y1<n && board[x1][y1]==0){
         x=x1; y=y1; board[x][y]=++level; parent[x][y]=i;
         int k=point[x][y]-1;
         if (k>=0){ x+=cmove[k][0]; y+=cmove[k][1];
            level+=2; board[x][y]=level; }
         return true; }
      return false;
    }
   public void up(){ int k=point[x][y]-1;
      if (k>=0){ board[x][y]=0; x+=cmove[k][0];
         y+=cmove[k][1]; level-=2; }
      int i=parent[x][y]; board[x][y]=0;
      x-=move[i][0]; y-=move[i][1]; --level;
    }
   public void output(){ board[0][0]=(board[1][2]+board[2][1])/2;
      board[m-1][0]=(board[m-2][2]+board[m-3][1])/2;
      board[0][n-1]=(board[2][n-2]+board[1][n-3])/2;
      board[m-1][n-1]=(board[m-2][n-3]+board[m-3][n-2])/2;
      for (int i=0; i<m; ++i){
         for (int j=0; j<n; ++j){
            if (board[i][j]<10) out.print("0");
            out.print(board[i][j]+", "); }
         out.println(); }
      out.println();
   }
```

```
   public boolean edge(int x1, int y1, int x2, int y2){
      return Math.abs(x1-x2)==2 && Math.abs(y1-y2)==1 ||
         Math.abs(x1-x2)==1 && Math.abs(y1-y2)==2;
   }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("out-2a-8-8.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int m=8, n=8; Knight2.out=fout;
      long begin=System.currentTimeMillis();
      BackSearch x=new BackSearch(new Knight2a(m, n), 1);
      x.depthfirst(); if (x.getnum()==0) fout.println("No Solution!");
      long end=System.currentTimeMillis();
      fout.println(m+", "+n+":  Elapsed time: "+(end-begin));
      fout.close(); }
}
01, 60, 39, 34, 31, 18, 09, 64,
38, 35, 32, 61, 10, 63, 30, 17,
59, 02, 37, 40, 33, 28, 19, 08,
36, 49, 42, 27, 62, 11, 16, 29,
43, 58, 03, 50, 41, 24, 07, 20,
48, 51, 46, 55, 26, 21, 12, 15,
57, 44, 53, 04, 23, 14, 25, 06,
52, 47, 56, 45, 54, 05, 22, 13,
8, 8:  Elapsed time: 1279

53, 44, 63, 28, 07, 16, 21, 18,
62, 29, 54, 43, 22, 19, 06, 15,
45, 52, 39, 64, 27, 08, 17, 20,
40, 61, 30, 55, 42, 23, 14, 05,
51, 46, 41, 38, 01, 26, 09, 24,
60, 35, 48, 31, 56, 11, 04, 13,
47, 50, 33, 58, 37, 02, 25, 10,
34, 59, 36, 49, 32, 57, 12, 03,
8, 8:  Elapsed time: 31
```

思考题 13：用算法 1.4.8 求解 8 皇后问题(推广到 *N* 皇后问题)。

```
public class Queen1 implements Mono {
   static PrintWriter out; int n, level; int[] board;
   public Queen1(int k){ n=k; board=new int[n]; }
   public boolean target(){ return level==n; }
   public int subnum(){ return level<n ? n : 0; }
   public boolean down(int i){ boolean q=true;
      for (int j=0; q && j<level; ++j)
         q=board[j]!=i && Math.abs(board[j]-i)!=Math.abs(j-level);
      if (q) board[level++]=i;  return q;
   }
```

```
   public void up(){ --level; }
   public void output(){
      for (int i=0; i<n; ++i) out.print(board[i]+", "); out.println();  }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("1-1-28.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int m=28; Queen1.out=fout;
      long begin=System.currentTimeMillis();
      //BackSearch x=new BackSearch(new Queen1(m), Integer.MAX_VALUE);
      BackSearch x=new BackSearch(new Queen1(m), 1);
      x.depthfirst(); //fout.println(x.getnum());
      long end=System.currentTimeMillis();
      fout.println(m+":  Elapsed time: "+(end-begin));
      fout.close();
    }
}
public class Queen2 extends Queen1 { boolean[] column, d1, d2;
   public Queen2(int k){ super(k); column=new boolean[n];
      d1=new boolean[2*n-1]; d2=new boolean[2*n-1]; }
   public boolean down(int i){
      if (!column[i]&&!d1[level+i] && !d2[level-i+n-1]){
         column[i]=d1[level+i]=d2[level-i+n-1]=true;
         board[level++]=i; return true; }
      return false; }
   public void up(){ int i=board[--level];
      column[i]=d1[level+i]=d2[level-i+n-1]=false;  }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("2-1-28.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int m=28; Queen1.out=fout;
      long begin=System.currentTimeMillis();
      //BackSearch x=new BackSearch(new Queen2(m), Integer.MAX_VALUE);
      BackSearch x=new BackSearch(new Queen2(m), 1);
      x.depthfirst(); //fout.println(x.getnum());
      long end=System.currentTimeMillis();
      fout.println(m+":  Elapsed time: "+(end-begin));
      fout.close();
   }
}
365596
14:  Elapsed time: 19656

0, 2, 4, 1, 3, 8, 10, 12, 14, 16, 22, 24, 21, 27, 25, 23, 26, 6, 11, 15, 17,
7, 9, 13, 19, 5, 20, 18,
28:  Elapsed time: 5538
```

```
365596
14:  Elapsed time: 5990

0, 2, 4, 1, 3, 8, 10, 12, 14, 16, 22, 24, 21, 27, 25, 23, 26, 6, 11, 15, 17,
7, 9, 13, 19, 5, 20, 18,
28:  Elapsed time: 1186
```

思考题 14：用算法 1.4.8 求解 0-1 矩阵问题。

```
public class BitMatrix1 implements Mono {
   int m, n; int[] row, column; int[][] data; PrintWriter fout;
   int[] rowcount, colcount; int x, y;
   public BitMatrix1(int i, int j, int[] r, int[] c, PrintWriter f){
      m=i; n=j; row=r; column=c; fout=f;
      data=new int[m][n]; rowcount=new int[m]; colcount=new int[n]; }
   public boolean target(){ return x==m; }
   public int subnum(){ return x<m ? 2 : 0; }
   public boolean down(int i){
      boolean ans = i==0 ?
         rowcount[x]+n-1-y>=row[x] && colcount[y]+m-1-x>=column[y] :
         rowcount[x]<row[x] && colcount[y]<column[y];
      if (ans){
        if (i==0) data[x][y]=0;
        else{ data[x][y]=1;  ++rowcount[x]; ++colcount[y]; }
        ++y; if(y==n){ y=0; ++x; } } return ans; }
   public void up(){ --y; if(y<0){ y=n-1; --x; }
      if (data[x][y]==1){ --rowcount[x]; --colcount[y]; } }
   public void output(){ /*
      for (int i=0; i<m; ++i){
          for (int j=0; j<n; ++j)
             fout.print(data[i][j]+", ");  fout.println(); }
      fout.println(); */ }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("out1a-all-8-8.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(), end;
      //int m=6, n=8; int[] r={2, 3, 2, 3, 5, 1}, c={2, 2, 2, 2, 2, 2, 1, 3};
      int m=8, n=8;
      int[] r={3, 3, 5, 3, 7, 3, 1, 4}, c={2, 3, 5, 4, 3, 6, 4, 2};
      //int m=10, n=12;
      //int[] r={10, 10, 6, 4, 4, 5, 10, 12, 9, 6},
      //c={2, 4, 5, 5, 7, 6, 7, 10, 10, 10, 7, 3};
      BackSearch x=new
         BackSearch(new BitMatrix1(m, n, r, c, fout), Integer.MAX_VALUE);
      x.depthfirst(); fout.println(x.getnum());
      end=System.currentTimeMillis();
      fout.println(m+", "+n+":  Elapsed time: "+(end-begin));
```

```
        fout.close(); }
    }
657480
6, 8:  Elapsed time: 1202
34883366
8, 8:  Elapsed time: 53821
```

思考题 15: 用算法 1.4.9 求解 0-1 矩阵问题。

```
public interface ItrMono1 {
   boolean target(); ItrObj subnodes(); boolean down(ItrObj itr);
   void up(); void output();
}
public class ItrBackSearch1 { ItrMono1 prob; int num, bound;
   public ItrBackSearch1(ItrMono1 p, int b) { prob=p; bound=b; }
   public void depthfirst(){
      ItrObj nodes=prob.subnodes(); boolean hasnext=true;
      if (prob.target()){ prob.output(); ++num; }
      while (num<bound && nodes!=null && hasnext){
         if (prob.down(nodes)){ depthfirst(); prob.up(); }
         hasnext=nodes.next(); }
   }
   public int getnum(){ return num; }
}
public class LexIncSeq  extends ComSet implements ComObj {
   public LexIncSeq(int mm, int nn){ super(nn); m=mm;
      for (int i=0; i<n; ++i) set[i]=i;
   }
   public boolean next(){ int i=n-1;
      while (i>=0 && set[i]==m-n+i) --i;
      if (i>=0){ ++set[i];
         for (int j=i+1; j<n; ++j) set[j]=set[j-1]+1;
      }
      return i>=0;
   }
}
public class BitMatrix4 implements ItrMono1 {
   int m, n; int[] row, column; int[][] data; PrintWriter fout;
   int[]  count; int level;
   public BitMatrix4(int i, int j, int[] r, int[] c, PrintWriter f){
      m=i; n=j; row=r; column=c; fout=f;
      data=new int[m][n]; count=new int[n]; }
   public boolean target(){ return level==m; }
   public ItrObj subnodes(){
      return level<m ? new LexIncSeq(n, row[level]) : null;
   }
   public boolean down(ItrObj itr){
```

```
      boolean res=true;  LexIncSeq com=(LexIncSeq)itr;
         Arrays.fill(data[level], 0, n, 0);
         for (int j=0; j<row[level]; ++j) data[level][com.content(j)]=1;
         for (int i=0; res && i<n; ++i)

   res=data[level][i]==1?count[i]<column[i]:level-count[i]<m-column[i];
         if (res){ for (int i=0; i<n; ++i) count[i]+=data[level][i]; ++level; }
         return res;
    }
   public void up(){ --level;
      for (int i=0; i<n; ++i) count[i]-=data[level][i]; }
   public void output(){ /*
      for (int i=0; i<m; ++i){
          for (int j=0; j<n; ++j)
            fout.print(data[i][j]+", ");  fout.println(); }
      fout.println(); */ }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("out4-all-8-8.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(), end;
      //int m=6, n=8; int[] r={2, 3, 2, 3, 5, 1}, c={2, 2, 2, 2, 2, 2, 1, 3};
      int m=8, n=8; int[] r={3, 3, 5, 3, 7, 3, 1, 4},
         c={2, 3, 5, 4, 3, 6, 4, 2};
      ItrBackSearch1 x=new
      ItrBackSearch1(new BitMatrix4(m, n, r, c, fout), Integer.MAX_VALUE);
      x.depthfirst(); fout.println(x.getnum());
      end=System.currentTimeMillis();
      fout.println(m+", "+n+":  Elapsed time: "+(end-begin));
      fout.close();
    }
}
```

657480

6, 8: Elapsed time: 3947

34883366

8, 8: Elapsed time: 243844

思考题 16: 用算法 1.4.9 求解循环赛日程安排问题。

```
public class Schedule2 implements ItrMono1 { PrintWriter fout;
   int n, lev; int[][] data; boolean[][] set;
   public Schedule2(int k, PrintWriter f){ fout=f;
      n=k; lev=1; data=new int[2*n][2*n]; set=new boolean[2*n][2*n];
      for (int i=1; i<2*n; ++i){ set[i][0]=set[i][i]=true;
         data[0][i]=data[i][0]=i;
      }
   }
   public boolean target(){ return lev==2*n; }
```

```
   public ItrObj subnodes(){
      return lev<2*n ? new LexPerm(2*n-2) : null;
   }
   public boolean down(ItrObj itr){ boolean res=true;  LexPerm
perm=(LexPerm)itr;
      for (int i=0; res && i<2*n-2; ++i){
         int k=perm.content(i), pos=i+1<lev ? i+1 : i+2;
         data[lev][pos]=k+1<lev ? k+1 : k+2;
         res=!set[pos][data[lev][pos]];
      }
      res=res && check(lev);
      if (res){
         for (int i=1; i<2*n; ++i)
            if (i!=lev) set[i][data[lev][i]]=true;
         ++lev;
      }
      return res;
   }
   public void up(){ --lev;
      for (int i=1; i<2*n; ++i)
         if (i!=lev) set[i][data[lev][i]]=false;
}
   public void output(){
      for (int i=0; i<2*n; ++i){
         for (int j=0; j<2*n; ++j){
            if (data[i][j]<10) fout.print("0");
            fout.print(data[i][j]+", ");
         }
         fout.println();
      }
      fout.println();
   }
   private boolean check(int k){ boolean ans=true;
      for (int i=1; ans && i<2*n; ++i)
         if (i!=k) ans=data[k][data[k][i]]==i;
      return ans;
   }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("out-two-7.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(), end;
      int m=7;
      ItrBackSearch1 x=new ItrBackSearch1(new Schedule2(m, fout), 1);
      x.depthfirst();
      end=System.currentTimeMillis();
      fout.println(m+":  Elapsed time: "+(end-begin));
```

```
      fout.close();
   }
}
public class Schedule1 extends Schedule2 {
   public Schedule1(int k, PrintWriter f){ super(k, f);
   }
   public boolean target(){ boolean ans=false;
      if (lev==2*n-1){ ans=true;
         for (int i=1; ans && i<2*n-1; ++i) ans=set[i][2*n-1];
         if (ans){ boolean tset[]=new boolean[2*n];
            for (int i=1; ans && i<2*n-1; ++i){  int j=1;
               while (set[i][j]) ++j;
               data[2*n-1][i]=j; ans=!tset[j]; tset[j]=true;
            }
            ans=check(2*n-1);
         }
      }
      return ans;
   }
   public ItrObj subnodes(){
      return lev<2*n-1 ? new LexPerm(2*n-2) : null;
   }
}
public class Schedule4 implements Mono { PrintWriter fout;
   int n, x, y; int[][] data; boolean[][] setx, sety;
   public Schedule4(int k, PrintWriter f){ fout=f;
      n=k; x=1; y=0; data=new int[2*n][2*n];
      setx=new boolean[2*n][2*n];  sety=new boolean[2*n][2*n];
      for (int i=0; i<2*n; ++i){ sety[i][i]=true; data[0][i]=i;
      }
   }
   public boolean target(){ return x==2*n; }
   public int subnum(){ return x<2*n ? 2*n : 0; }
   public boolean down(int k){ boolean res=!setx[x][k] && !sety[y][k];
      if (res && k<y) res=data[x][k]==y;
      if (res){ data[x][y]=k; setx[x][k]=sety[y][k]=true; ++y;
         if (y==2*n){ ++x; y=0; }
      }
      return res;
   }
   public void up(){ --y; if (y<0){ --x; y=2*n-1; }
      setx[x][data[x][y]]=sety[y][data[x][y]]=false;
   }
public class Schedule3 implements Mono { PrintWriter fout;
   int n, x, y; int[][] data, lab; boolean[][] setx, sety;
   public Schedule3(int k, PrintWriter f){ fout=f;
```

```
        n=k; x=1; y=0; data=new int[2*n][2*n]; lab=new int[2*n][2*n];
        for (int i=0; i<2*n; ++i) for (int j=0; j<2*n; ++j) lab[i][j]=-1;
        setx=new boolean[2*n][2*n];  sety=new boolean[2*n][2*n];
        for (int i=0; i<2*n; ++i){ sety[i][i]=true; data[0][i]=i;
        }
    }
    public boolean target(){ return x==2*n; }
    public int subnum(){ return x==2*n ? 0 : lab[x][y]>=0 ?  1 : 2*n-y-1; }
    public boolean down(int k){ boolean res=true; int k1=lab[x][y], k2=y+k+1;
        if (k1>=0) { data[x][y]=k1; sety[y][k1]=true; }
        else {   res = !setx[x][k2] && !sety[y][k2];
            if (res){ data[x][y]=k2;
            setx[x][k2]=sety[y][k2]=true; lab[x][k2]=y; }
        }
        if (res){ ++y; if (y==2*n){ ++x; y=0; }
        }
        return res;
    }
    public void up(){ --y; if (y<0){ --x; y=2*n-1; }
        setx[x][data[x][y]]=sety[y][data[x][y]]=false;
        if (y<data[x][y]) lab[x][data[x][y]]=-1;
    }
```

Schedule2:

00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13,
01, 00, 03, 02, 05, 04, 07, 06, 09, 08, 11, 10, 13, 12,
02, 03, 00, 01, 06, 07, 04, 05, 10, 12, 08, 13, 09, 11,
03, 02, 01, 00, 07, 06, 05, 04, 11, 13, 12, 08, 10, 09,
04, 05, 06, 07, 00, 01, 02, 03, 12, 11, 13, 09, 08, 10,
05, 04, 07, 06, 01, 00, 03, 02, 13, 10, 09, 12, 11, 08,
06, 08, 09, 10, 11, 12, 00, 13, 01, 02, 03, 04, 05, 07,
07, 09, 08, 11, 10, 13, 12, 00, 02, 01, 04, 03, 06, 05,
08, 06, 10, 09, 13, 11, 01, 12, 00, 03, 02, 05, 07, 04,
09, 07, 11, 08, 12, 10, 13, 01, 03, 00, 05, 02, 04, 06,
10, 12, 04, 13, 02, 08, 09, 11, 05, 06, 00, 07, 01, 03,
11, 13, 05, 12, 08, 02, 10, 09, 04, 07, 06, 00, 03, 01,
12, 10, 13, 04, 03, 09, 11, 08, 07, 05, 01, 06, 00, 02,
13, 11, 12, 05, 09, 03, 08, 10, 06, 04, 07, 01, 02, 00,

7: Elapsed time: 110651

Schedule1:

00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13,
01, 00, 03, 02, 05, 04, 07, 06, 09, 08, 11, 10, 13, 12,
02, 03, 00, 01, 06, 07, 04, 05, 10, 12, 08, 13, 09, 11,
03, 02, 01, 00, 07, 06, 05, 04, 11, 13, 12, 08, 10, 09,
04, 05, 06, 07, 00, 01, 02, 03, 12, 11, 13, 09, 08, 10,
05, 04, 07, 06, 01, 00, 03, 02, 13, 10, 09, 12, 11, 08,

06, 08, 09, 10, 11, 12, 00, 13, 01, 02, 03, 04, 05, 07,
07, 09, 08, 11, 10, 13, 12, 00, 02, 01, 04, 03, 06, 05,
08, 06, 10, 09, 13, 11, 01, 12, 00, 03, 02, 05, 07, 04,
09, 07, 11, 08, 12, 10, 13, 01, 03, 00, 05, 02, 04, 06,
10, 12, 04, 13, 02, 08, 09, 11, 05, 06, 00, 07, 01, 03,
11, 13, 05, 12, 08, 02, 10, 09, 04, 07, 06, 00, 03, 01,
12, 10, 13, 04, 03, 09, 11, 08, 07, 05, 01, 06, 00, 02,
13, 11, 12, 05, 09, 03, 08, 10, 06, 04, 07, 01, 02, 00,
7: Elapsed time: 94505

Schedule4:
00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13,
01, 00, 03, 02, 05, 04, 07, 06, 09, 08, 11, 10, 13, 12,
02, 03, 00, 01, 06, 07, 04, 05, 10, 12, 08, 13, 09, 11,
03, 02, 01, 00, 07, 06, 05, 04, 11, 13, 12, 08, 10, 09,
04, 05, 06, 07, 00, 01, 02, 03, 12, 11, 13, 09, 08, 10,
05, 04, 07, 06, 01, 00, 03, 02, 13, 10, 09, 12, 11, 08,
06, 08, 09, 10, 11, 12, 00, 13, 01, 02, 03, 04, 05, 07,
07, 09, 08, 11, 10, 13, 12, 00, 02, 01, 04, 03, 06, 05,
08, 06, 10, 09, 13, 11, 01, 12, 00, 03, 02, 05, 07, 04,
09, 07, 11, 08, 12, 10, 13, 01, 03, 00, 05, 02, 04, 06,
10, 12, 04, 13, 02, 08, 09, 11, 05, 06, 00, 07, 01, 03,
11, 13, 05, 12, 08, 02, 10, 09, 04, 07, 06, 00, 03, 01,
12, 10, 13, 04, 03, 09, 11, 08, 07, 05, 01, 06, 00, 02,
13, 11, 12, 05, 09, 03, 08, 10, 06, 04, 07, 01, 02, 00,
7: Elapsed time: 31

00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
01, 00, 03, 02, 05, 04, 07, 06, 09, 08, 11, 10, 13, 12, 15, 14, 17, 16, 19, 18, 21, 20,
02, 03, 00, 01, 06, 07, 04, 05, 10, 11, 08, 09, 14, 15, 12, 13, 18, 20, 16, 21, 17, 19,
03, 02, 01, 00, 07, 06, 05, 04, 11, 10, 09, 08, 15, 14, 13, 12, 19, 21, 20, 16, 18, 17,
04, 05, 06, 07, 00, 01, 02, 03, 12, 13, 14, 15, 08, 09, 10, 11, 20, 19, 21, 17, 16, 18,
05, 04, 07, 06, 01, 00, 03, 02, 13, 12, 15, 14, 09, 08, 11, 10, 21, 18, 17, 20, 19, 16,
06, 07, 04, 05, 02, 03, 00, 01, 14, 16, 17, 18, 19, 20, 08, 21, 09, 10, 11, 12, 13, 15,
07, 06, 05, 04, 03, 02, 01, 00, 15, 17, 16, 19, 18, 21, 20, 08, 10, 09, 12, 11, 14, 13,
08, 09, 10, 11, 12, 16, 17, 18, 00, 01, 02, 03, 04, 19, 21, 20, 05, 06, 07, 13, 15, 14,
09, 08, 11, 10, 13, 17, 16, 20, 01, 00, 03, 02, 21, 04, 18, 19, 06, 05, 14, 15, 07, 12,
10, 11, 08, 09, 14, 18, 19, 21, 02, 03, 00, 01, 20, 16, 04, 17, 13, 15, 05, 06, 12, 07,
11, 10, 09, 08, 15, 20, 21, 16, 03, 02, 01, 00, 17, 18, 19, 04, 07, 12, 13, 14, 05, 06,
12, 13, 14, 15, 08, 19, 18, 17, 04, 20, 21, 16, 00, 01, 02, 03, 11, 07, 06, 05, 09, 10,
13, 12, 15, 14, 09, 21, 20, 19, 16, 04, 18, 17, 01, 00, 03, 02, 08, 11, 10, 07, 06, 05,
14, 15, 16, 17, 18, 08, 12, 13, 05, 19, 20, 21, 06, 07, 00, 01, 02, 03, 04, 09, 10, 11,
15, 14, 17, 16, 19, 10, 13, 12, 18, 21, 05, 20, 07, 06, 01, 00, 03, 02, 08, 04, 11, 09,
16, 17, 12, 13, 20, 09, 11, 14, 21, 05, 19, 06, 02, 03, 07, 18, 00, 01, 15, 10, 04, 08,
17, 16, 13, 19, 21, 11, 14, 15, 20, 18, 12, 05, 10, 02, 06, 07, 01, 00, 09, 03, 08, 04,
18, 19, 20, 21, 10, 12, 08, 09, 06, 07, 04, 13, 05, 11, 17, 16, 15, 14, 00, 01, 02, 03,

19, 18, 21, 20, 11, 13, 15, 10, 17, 14, 07, 04, 16, 05, 09, 06, 12, 08, 01, 00, 03, 02,
20, 21, 18, 12, 16, 14, 10, 11, 19, 15, 06, 07, 03, 17, 05, 09, 04, 13, 02, 08, 00, 01,
21, 20, 19, 18, 17, 15, 09, 08, 07, 06, 13, 12, 11, 10, 16, 05, 14, 04, 03, 02, 01, 00,
11: Elapsed time: 2293

Schedule3:
00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
01, 00, 03, 02, 05, 04, 07, 06, 09, 08, 11, 10, 13, 12, 15, 14, 17, 16, 19, 18, 21, 20,
02, 03, 00, 01, 06, 07, 04, 05, 10, 11, 08, 09, 14, 15, 12, 13, 18, 20, 16, 21, 17, 19,
03, 02, 01, 00, 07, 06, 05, 04, 11, 10, 09, 08, 15, 14, 13, 12, 19, 21, 20, 16, 18, 17,
04, 05, 06, 07, 00, 01, 02, 03, 12, 13, 14, 15, 08, 09, 10, 11, 20, 19, 21, 17, 16, 18,
05, 04, 07, 06, 01, 00, 03, 02, 13, 12, 15, 14, 09, 08, 11, 10, 21, 18, 17, 20, 19, 16,
06, 07, 04, 05, 02, 03, 00, 01, 14, 16, 17, 18, 19, 20, 08, 21, 09, 10, 11, 12, 13, 15,
07, 06, 05, 04, 03, 02, 01, 00, 15, 17, 16, 19, 18, 21, 20, 08, 10, 09, 12, 11, 14, 13,
08, 09, 10, 11, 12, 16, 17, 18, 00, 01, 02, 03, 04, 19, 21, 20, 05, 06, 07, 13, 15, 14,
09, 08, 11, 10, 13, 17, 16, 20, 01, 00, 03, 02, 21, 04, 18, 19, 06, 05, 14, 15, 07, 12,
10, 11, 08, 09, 14, 18, 19, 21, 02, 03, 00, 01, 20, 16, 04, 17, 13, 15, 05, 06, 12, 07,
11, 10, 09, 08, 15, 20, 21, 16, 03, 02, 01, 00, 17, 18, 19, 04, 07, 12, 13, 14, 05, 06,
12, 13, 14, 15, 08, 19, 18, 17, 04, 20, 21, 16, 00, 01, 02, 03, 11, 07, 06, 05, 09, 10,
13, 12, 15, 14, 09, 21, 20, 19, 16, 04, 18, 17, 01, 00, 03, 02, 08, 11, 10, 07, 06, 05,
14, 15, 16, 17, 18, 08, 12, 13, 05, 19, 20, 21, 06, 07, 00, 01, 02, 03, 04, 09, 10, 11,
15, 14, 17, 16, 19, 10, 13, 12, 18, 21, 05, 20, 07, 06, 01, 00, 03, 02, 08, 04, 11, 09,
16, 17, 12, 13, 20, 09, 11, 14, 21, 05, 19, 06, 02, 03, 07, 18, 00, 01, 15, 10, 04, 08,
17, 16, 13, 19, 21, 11, 14, 15, 20, 18, 12, 05, 10, 02, 06, 07, 01, 00, 09, 03, 08, 04,
18, 19, 20, 21, 10, 12, 08, 09, 06, 07, 04, 13, 05, 11, 17, 16, 15, 14, 00, 01, 02, 03,
19, 18, 21, 20, 11, 13, 15, 10, 17, 14, 07, 04, 16, 05, 09, 06, 12, 08, 01, 00, 03, 02,
20, 21, 18, 12, 16, 14, 10, 11, 19, 15, 06, 07, 03, 17, 05, 09, 04, 13, 02, 08, 00, 01,
21, 20, 19, 18, 17, 15, 09, 08, 07, 06, 13, 12, 11, 10, 16, 05, 14, 04, 03, 02, 01, 00,
11: Elapsed time: 16

00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
01, 00, 03, 02, 05, 04, 07, 06, 09, 08, 11, 10, 13, 12, 15, 14, 17, 16, 19, 18, 21, 20, 23, 22, 25, 24,
02, 03, 00, 01, 06, 07, 04, 05, 10, 11, 08, 09, 14, 15, 12, 13, 18, 19, 16, 17, 22, 24, 20, 25, 21, 23,
03, 02, 01, 00, 07, 06, 05, 04, 11, 10, 09, 08, 15, 14, 13, 12, 19, 18, 17, 16, 23, 25, 24, 20, 22, 21,
04, 05, 06, 07, 00, 01, 02, 03, 12, 13, 14, 15, 08, 09, 10, 11, 20, 21, 22, 25, 16, 17, 18, 24, 23, 19,
05, 04, 07, 06, 01, 00, 03, 02, 13, 12, 15, 14, 09, 08, 11, 10, 21, 20, 23, 24, 17, 16, 25, 18, 19, 22,
06, 07, 04, 05, 02, 03, 00, 01, 14, 15, 12, 13, 10, 11, 08, 09, 22, 23, 24, 21, 25, 19, 16, 17, 18, 20,
07, 06, 05, 04, 03, 02, 01, 00, 15, 14, 13, 12, 11, 10, 09, 08, 23, 24, 25, 20, 19, 22, 21, 16, 17, 18,
08, 09, 10, 11, 12, 13, 14, 15, 00, 01, 02, 03, 04, 05, 06, 07, 24, 25, 20, 22, 18, 23, 19, 21, 16, 17,
09, 08, 11, 10, 13, 12, 15, 14, 01, 00, 03, 02, 05, 04, 07, 06, 25, 22, 21, 23, 24, 18, 17, 19, 20, 16,
10, 11, 08, 16, 17, 18, 19, 20, 02, 21, 00, 01, 22, 23, 24, 25, 03, 04, 05, 06, 07, 09, 12, 13, 14, 15,
11, 10, 09, 17, 16, 19, 18, 21, 20, 02, 01, 00, 23, 22, 25, 24, 04, 03, 06, 05, 08, 07, 13, 12, 15, 14,
12, 13, 14, 18, 19, 16, 17, 22, 21, 20, 24, 25, 00, 01, 02, 23, 05, 06, 03, 04, 09, 08, 07, 15, 10, 11,
13, 12, 15, 19, 18, 17, 16, 23, 22, 24, 25, 20, 01, 00, 21, 02, 06, 05, 04, 03, 11, 14, 08, 07, 09, 10,
14, 15, 12, 20, 21, 22, 23, 16, 17, 18, 19, 24, 02, 25, 00, 01, 07, 08, 09, 10, 03, 04, 05, 06, 11, 13,
15, 14, 13, 21, 20, 23, 22, 17, 16, 25, 18, 19, 24, 02, 01, 00, 08, 07, 10, 11, 04, 03, 06, 05, 12, 09,

16, 17, 18, 08, 09, 10, 20, 19, 03, 04, 05, 21, 25, 24, 23, 22, 00, 01, 02, 07, 06, 11, 15, 14, 13, 12,
17, 16, 19, 09, 08, 11, 24, 25, 04, 03, 23, 05, 18, 20, 22, 21, 01, 00, 12, 02, 13, 15, 14, 10, 06, 07,
18, 19, 16, 12, 10, 08, 25, 24, 05, 22, 04, 23, 03, 21, 17, 20, 02, 14, 00, 01, 15, 13, 09, 11, 07, 06,
19, 18, 17, 13, 11, 24, 21, 12, 25, 23, 22, 04, 07, 03, 20, 16, 15, 02, 01, 00, 14, 06, 10, 09, 05, 08,
20, 21, 22, 14, 23, 25, 09, 10, 24, 06, 07, 16, 17, 18, 03, 19, 11, 12, 13, 15, 00, 01, 02, 04, 08, 05,
21, 20, 23, 24, 25, 09, 08, 13, 06, 05, 16, 22, 19, 07, 18, 17, 10, 15, 14, 12, 01, 00, 11, 02, 03, 04,
22, 23, 20, 25, 24, 14, 10, 08, 07, 16, 06, 17, 21, 19, 05, 18, 09, 11, 15, 13, 02, 12, 00, 01, 04, 03,
23, 24, 25, 15, 22, 20, 12, 09, 19, 07, 21, 18, 06, 17, 16, 03, 14, 13, 11, 08, 05, 10, 04, 00, 01, 02,
24, 25, 21, 22, 14, 15, 11, 18, 23, 19, 17, 06, 20, 16, 04, 05, 13, 10, 07, 09, 12, 02, 03, 08, 00, 01,
25, 22, 24, 23, 15, 21, 13, 11, 18, 17, 20, 07, 16, 06, 19, 04, 12, 09, 08, 14, 10, 05, 01, 03, 02, 00,
13: Elapsed time: 14482

思考题 17: 用算法 1.4.10 求解 N 皇后问题。

```
public interface PerMono extends Mono { int depth(); }
public class PerBackSearch {
   Mono prob; int dep, level; int num, bound; int[] perm;
   public PerBackSearch(Mono p, int b, int k) {
      prob=p; bound=b; dep=k;
      perm=new int[dep]; for (int i=0; i<dep; ++i) perm[i]=i; }
   public void depthfirst(){
      if (prob.target()){ prob.output(); ++num; }
      else for (int i=level; num<bound && i<dep; ++i)
         if (prob.down(perm[i])){
            int t=perm[level]; perm[level]=perm[i]; perm[i]=t;
            ++level; depthfirst(); prob.up(); --level;
            t=perm[level]; perm[level]=perm[i]; perm[i]=t; }
   }
   public long getnum(){ return num; }
}
public class Queen3 extends Queen2 {
   public Queen3(int k) { super(k); }
   public boolean down(int i){ if (!d1[level+i] && !d2[level-i+n-1]){
      d1[level+i]=d2[level-i+n-1]=true; board[level++]=i; return true; }
      return false;
   }
   public void up(){ int i=board[--level];
      d1[level+i]=d2[level-i+n-1]=false; }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("3-all-14.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int m=14; Queen1.out=fout;
      long begin=System.currentTimeMillis();
      PerBackSearch x=new PerBackSearch(new Queen3(m), Integer.MAX_VALUE, m);
      //PerBackSearch x=new PerBackSearch(new Queen3(m), 1, m);
      x.depthfirst(); fout.println(x.getnum());
      long end=System.currentTimeMillis();
```

```
      fout.println(m+":  Elapsed time: "+(end-begin));
      fout.close();
   }
}
```

```
0, 2, 4, 1, 3, 8, 10, 12, 14, 16, 22, 24, 21, 27, 25, 23, 26, 7, 13, 15, 6, 9, 5, 20, 17, 19, 11, 18,
28:  Elapsed time: 484
365596
14:  Elapsed time: 2543
```

思考题 18: 经验表明，如果在树形状态空间的深度优先搜索时，如果先遍历根结点分叉少的子树，就会更快地搜索到第一个解。编写和实现算法验证这一结论。选择 *N* 皇后问题和跳马问题作为搜索对象。

```
public interface Mono1 {
   boolean target(); int subnum(); boolean defined(int i);
   void down(int i); void up(); void output();
}
public class BackSearch1 { Mono1 prob; int num, bound, wid;
   static Subnode[] subs;
   public BackSearch1(Mono1 p, int b, int w) { prob=p; bound=b;
      wid=w; subs=new Subnode[wid]; }
   public void depthfirst(){ int n=prob.subnum();
      if (prob.target()){ prob.output(); ++num; }
      for (int i=0; num<bound && i<n; ++i)
         if (prob.defined(i)){
            prob.down(i); depthfirst(); prob.up(); }
   }
   public int getnum(){ return num; }
   public class Subnode implements Comparator {
      public byte n, k; public Subnode[] subnodes;
      public Subnode(byte kk){ k=kk; }
      public int compare(Object obj1, Object obj2){
         return ((Subnode)obj1).n-((Subnode)obj2).n; }
   }
   public void reordering(){ int nn=prob.subnum(); byte cnum=0;
       for (byte i=0; i<nn; ++i)
         if (prob.defined(i)) subs[cnum++]=new Subnode(i);
       Subnode[] root=new Subnode[cnum];
      System.arraycopy(subs, 0, root, 0, cnum);
      dfirst(root);
   }
   public void dfirst(Subnode[] a){ byte newn=0;
      for (int i=0; num<bound && i<a.length; ++i){ prob.down(a[i].k);
         if (prob.target()){ prob.output(); ++num; }
         else { byte nn=(byte)prob.subnum(), cnum=0;
            for (byte j=0; j<nn; ++j)
               if (prob.defined(j)) subs[cnum++]=new Subnode(j);
               if (cnum>0){ a[newn].n=cnum; a[newn].k=a[i].k;
```

```
                    a[newn].subnodes=new Subnode[cnum];
                    System.arraycopy(subs, 0, a[newn].subnodes, 0, cnum);
                    ++newn; }
            }
            prob.up();
        }
        if (num<bound){
            if (newn>1) Arrays.sort(a, 0, newn, a[0]);
            for (int i=0; num<bound && i<newn; ++i){
                prob.down(a[i].k); dfirst(a[i].subnodes); prob.up();
            }
        }
    }
}
public class BackSearch2 { Mono1 prob; int num, bound, wid;
    static Subnode1[] subs;
    public BackSearch2(Mono1 p, int b, int w) { prob=p; bound=b;
        wid=w; subs=new Subnode1[wid]; }
    public void depthfirst(){ int n=prob.subnum();
        if (prob.target()){ prob.output(); ++num; }
        for (int i=0; num<bound && i<n; ++i)
            if (prob.defined(i)){
                prob.down(i); depthfirst(); prob.up(); }
    }
    public int getnum(){ return num; }
    public class Subnode1 implements Comparator {
        public byte n, k, p;
        public Subnode1(byte kk){ k=kk; }
        public int compare(Object obj1, Object obj2){
            return ((Subnode1)obj1).n-((Subnode1)obj2).n; }
    }
    public void reordering(){ int nn=prob.subnum(); byte cnum=0;
        for (byte i=0; i<nn; ++i)
            if (prob.defined(i)) subs[cnum++]=new Subnode1(i);
        Subnode1[] root=new Subnode1[cnum];
        System.arraycopy(subs, 0, root, 0, cnum);
        dfirst(root);
    }
    public void dfirst(Subnode1[] a){ byte newn=0;
        Subnode1[][] as=new Subnode1[a.length][];
        for (int i=0; num<bound && i<a.length; ++i){ prob.down(a[i].k);
            if (prob.target()){ prob.output(); ++num; }
            else { byte nn=(byte)prob.subnum(), cnum=0;
                for (byte j=0; j<nn; ++j)
                    if (prob.defined(j)) subs[cnum++]=new Subnode1(j);
                    if (cnum>0){
```

```
                    a[newn].n=cnum; a[newn].k=a[i].k; a[newn].p=newn;
                    as[newn]=new Subnode1[cnum];
                    System.arraycopy(subs, 0, as[newn], 0, cnum); ++newn; }
            }
            prob.up();
        }
        if (num<bound){
           if (newn>1) Arrays.sort(a, 0, newn, a[0]);
           for (int i=0; num<bound && i<newn; ++i){
              prob.down(a[i].k); dfirst(as[a[i].p]); prob.up();
           }
        }
    }
}
public class Queen4 implements Mono1 {
    int n, level; int[] board; boolean[] column, d1, d2;
    public Queen4(int k){ n=k; board=new int[n]; column=new boolean[n];
       d1=new boolean[2*n-1]; d2=new boolean[2*n-1]; }
    public boolean target(){ return level==n; }
    public int subnum(){ return level<n ? n : 0; }
    public boolean defined(int i){
       return !column[i] && !d1[level+i] && !d2[level-i+n-1]; }
    public void down(int i){
       column[i]=d1[level+i]=d2[level-i+n-1]=true; board[level++]=i; }
    public void up(){ int i=board[--level];
       column[i]=d1[level+i]=d2[level-i+n-1]=false; }
    public void output(){
       for (int i=0; i<n; ++i) System.out.print(board[i]+", ");
       System.out.println();  }
    public static void main(String[] args){
       int m=87; long begin=System.currentTimeMillis();
       //BackSearch1 x=new BackSearch1(new Queen4(m), 1, m);
       BackSearch2 x=new BackSearch2(new Queen4(m), 1, m);
       //x.depthfirst(); //fout.println(x.getnum());
       x.reordering();
       long end=System.currentTimeMillis();
       System.out.println(m+":  Elapsed time: "+(end-begin));
    }
}
public class Queen5 implements Mono1 { static PrintWriter out;
    int n, level; int[] board, index; boolean[] d1, d2;
    public Queen5(int k){ n=k; board=new int[n]; index=new int[n];
       d1=new boolean[2*n-1]; d2=new boolean[2*n-1];
       for (int i=0; i<n; ++i) board[i]=i; }
    public boolean target(){ return level==n; }
    public int subnum(){ return n-level; }
```

```
   public boolean defined(int i){ int j=level+i, k=board[j];
      return !d1[level+k] && !d2[level-k+n-1]; }
   public void down(int i){ int j=level+i, k=board[j];
      board[j]=board[level]; board[level]=k; index[level]=j;
      d1[level+k]=d2[level-k+n-1]=true; ++level; }
   public void up(){ int i=board[--level], j=index[level];
      d1[level+i]=d2[level-i+n-1]=false;
      board[level]=board[j]; board[j]=i; }
   public void output(){
      for (int i=0; i<n; ++i) out.print(board[i]+", ");
      out.println();  }
   public static void main(String[] args){ PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("5*-1-26.txt")); }
      catch(IOException e){ System.out.println("Error!!"); }
      int m=26; Queen5.out=fout;
      long begin=System.currentTimeMillis();
      //BackSearch1 x=new BackSearch1(new Queen5(m), 1, m);
      BackSearch2 x=new BackSearch2(new Queen5(m), 1, m);
      x.depthfirst(); //fout.println(x.getnum());
      //x.reordering();
      long end=System.currentTimeMillis();
      fout.println(m+":  Elapsed time: "+(end-begin));
      fout.close();
   }
}
public class Knight3 implements Mono1 {
   static final int[][] move=
      {{2,1},{1,2},{-1,2},{-2,1},{-2,-1},{-1,-2},{1,-2},{2,-1}};
      //{{1,2},{2,1},{-1,-2},{-2,-1},{2,-1},{1,-2},{-2,1},{-1,2}};
   int m, n, x, y, level; int[][] board, parent;
   public Knight3(int mm, int nn, int x, int y){
      m=mm; n=nn; this.x=x; this.y=y; level=1;
      board=new int[m][n]; board[x][y]=1; parent=new int[m][n];
   }
   public boolean target(){ return level==m*n; }
   public int subnum(){ return level<m*n ? 8 : 0; }
   public boolean defined(int i){
      int x1=x+move[i][0], y1=y+move[i][1];
      return x1>=0 && x1<m && y1>=0 && y1<n && board[x1][y1]==0;
   }
   public void down(int i){ int x1=x+move[i][0], y1=y+move[i][1];
      x=x1; y=y1; board[x][y]=++level; parent[x][y]=i;
   }
   public void up(){ int i=parent[x][y]; board[x][y]=0;
      x-=move[i][0]; y-=move[i][1]; --level;
   }
```

```
   public void output(){
      for (int i=0; i<m; ++i){
         for (int j=0; j<n; ++j){
            if (board[i][j]<10) System.out.print("000");
            else if (board[i][j]<100) System.out.print("00");
            else if (board[i][j]<1000) System.out.print("0");
            System.out.print(board[i][j]+", "); }
         System.out.println(); }
      System.out.println();
   }
   public static void main(String[] args){
      int m=9, n=9;
      long begin=System.currentTimeMillis();
      BackSearch2 x=new BackSearch2(new Knight3(m, n, 0, 0), 1, 8);
      x.depthfirst(); //fout.println(x.getnum());
      //x.reordering();
      long end=System.currentTimeMillis();
      System.out.println(m+", "+n+":  Elapsed time: "+(end-begin));
   }
}
public class Knight4 extends Knight3{
   static final int[][] cmove={{1,1},{-1,-1}, {-1,1}, {1,-1}};
   int[][] point;
   public Knight4(int mm, int nn){
      super(mm, nn, mm/2, nn/2); level=1; point=new int[m][n];
      point[2][1]=3; point[1][2]=4; point[m-3][1]=1;
      point[m-2][2]=2; point[m-2][n-3]=3; point[m-3][n-2]=4;
      point[2][n-2]=2; point[1][n-3]=1;
   }
   public boolean target(){
      return level==m*n && edge(x, y, m/2, n/2); }
   public int subnum(){
      return level<m*n ? 8 : 0; }
   public void down(int i){
      int x1=x+move[i][0], y1=y+move[i][1];
      x=x1; y=y1; board[x][y]=++level; parent[x][y]=i;
      int k=point[x][y]-1;
      if (k>=0){ x+=cmove[k][0]; y+=cmove[k][1];
         level+=2; board[x][y]=level; }
   }
   public void up(){ int k=point[x][y]-1;
      if (k>=0){ board[x][y]=0; x+=cmove[k][0];
         y+=cmove[k][1]; level-=2; }
      int i=parent[x][y]; board[x][y]=0;
      x-=move[i][0]; y-=move[i][1]; --level;
}
```

```
    public void output(){
        board[0][0]=(board[1][2]+board[2][1])/2;
        board[m-1][0]=(board[m-2][2]+board[m-3][1])/2;
        board[0][n-1]=(board[2][n-2]+board[1][n-3])/2;
        board[m-1][n-1]=(board[m-2][n-3]+board[m-3][n-2])/2;
        for (int i=0; i<m; ++i){
            for (int j=0; j<n; ++j){
                if (board[i][j]<10) System.out.print("00");
                else if (board[i][j]<100) System.out.print("0");
                System.out.print(board[i][j]+", "); }
            System.out.println(); }
        System.out.println();
    }
    public boolean edge(int x1, int y1, int x2, int y2){
        return Math.abs(x1-x2)==2 && Math.abs(y1-y2)==1 ||
            Math.abs(x1-x2)==1 && Math.abs(y1-y2)==2;
    }
    public static void main(String[] args){ int m=20, n=20;
        long begin=System.currentTimeMillis();
        BackSearch2 x=new BackSearch2(new Knight4(m, n), 1, 8);
        //x.depthfirst(); //fout.println(x.getnum());
        x.reordering();
        long end=System.currentTimeMillis();
        System.out.println(m+", "+n+":  Elapsed time: "+(end-begin));
    }
}
```

N 皇后问题未经重排结点的运行时间：

N	4-27	28	29	30	31	32	33	34	35	36
Q4(s)	<1	1.2	<1	22.4	5.5	36.3	62.8	>120	未检查	未检查
Q5(s)	<1	<1	<1	3.0	<1	7.4	5.4	46.5	51.2	>120

N 皇后问题重排结点的运行时间：(先用重排版本 1，Q4 运行至 N=75，Q5 运行至 N=77 出现内存不够错误，改用重排版本 2)

N	4-62	63	64	65-71	72	73-74	75	76	77	78	79
Q4	<1	<1	<1	<1	<1	<1	1.9	1.1	2.4	2.0	40.1
Q5	<1	2.9	3.8	<1	4.0	<1	<1	<1	49.8	261.7	<1

N	80	81	82	83	84	85	86	87	88	89	90
Q4	155.1	5.5	142.6	<1	<1	<1	<1	>600	>600	未检查	未检查
Q5	<1	<1	<1	341.2	>600	16.7	3.3	63.6	50.1	66.9	231.4

跳马问题未经重排结点的运行时间：

N	6×6	6×8	8×8	8×10	10×10
K3(路)	<1	26.6	1.3	>600	未检查
K4(回路)	<1	11.8	<1	5.7	>600

跳马问题重排结点的运行时间：K3 在 N=6-52 时运行时间都在 1 秒以内，N=53 时出现内存不够错误，

N	8×10	10×10	12×12	14×14	16×16	18×18	18×20	20×20
K4(回路)	<1	<1	<1	<1	<1	90.6	>600	>600

算法欣赏 1：循环赛日程安排问题。

```
public class Schedule {
   public static void main(String[] args){
      PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("50.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(), end;
      int n=50;
      for (int i=0; i<2*n; ++i){
         if (i<10) fout.print("0");
         fout.print(i+", "); }
      fout.println();
      for (int i=0; i<2*n-1; ++i){
         for (int j=0; j<2*n-1; ++j){
            int ans = j==i ? 2*n-1 : 2*i-j>=2*n-1 ?
               2*i-j-2*n+1 : 2*i-j<0 ? 2*i-j+2*n-1 : 2*i-j;
            if (ans<10) fout.print("0");
            fout.print(ans+", "); }
         if (i<10) fout.print("0");
         fout.println(i+", ");
      }
      end=System.currentTimeMillis();
      fout.println(n+":  Elapsed time: "+(end-begin));
      fout.close();
   }
}
```

```
00, 01, 02, 03, 04, 05, 06, 07, 08, 09,
09, 08, 07, 06, 05, 04, 03, 02, 01, 00,
02, 09, 00, 08, 07, 06, 05, 04, 03, 01,
04, 03, 09, 01, 00, 08, 07, 06, 05, 02,
06, 05, 04, 09, 02, 01, 00, 08, 07, 03,
08, 07, 06, 05, 09, 03, 02, 01, 00, 04,
01, 00, 08, 07, 06, 09, 04, 03, 02, 05,
03, 02, 01, 00, 08, 07, 09, 05, 04, 06,
```

05, 04, 03, 02, 01, 00, 08, 09, 06, 07,
07, 06, 05, 04, 03, 02, 01, 00, 09, 08,
5: Elapsed time: 0

00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00,
02, 25, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 01,
04, 03, 25, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 02,
06, 05, 04, 25, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 09, 08, 07, 03,
08, 07, 06, 05, 25, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 09, 04,
10, 09, 08, 07, 06, 25, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 05,
12, 11, 10, 09, 08, 07, 25, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 06,
14, 13, 12, 11, 10, 09, 08, 25, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 07,
16, 15, 14, 13, 12, 11, 10, 09, 25, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 08,
18, 17, 16, 15, 14, 13, 12, 11, 10, 25, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 09,
20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 25, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 10,
22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 25, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 11,
24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 25, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 12,
01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 25, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 13,
03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 25, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 14,
05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 16, 25, 14, 13, 12, 11, 10, 09, 08, 07, 06, 15,
07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 17, 25, 15, 14, 13, 12, 11, 10, 09, 08, 16,
09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 18, 25, 16, 15, 14, 13, 12, 11, 10, 17,
11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 19, 25, 17, 16, 15, 14, 13, 12, 18,
13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 20, 25, 18, 17, 16, 15, 14, 19,
15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 21, 25, 19, 18, 17, 16, 20,
17, 16, 15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 22, 25, 20, 19, 18, 21,
19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 23, 25, 21, 20, 22,
21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 24, 25, 22, 23,
23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00, 25, 24,
13: Elapsed time: 0