1 N 皇后问题的快速算法：

```
public class Queen0 {
   public static void main(String[] args){
      long begin=System.currentTimeMillis(), end;
      int n=14, m=n/2, r=m%3;
      if (r!=1){
         for (int i=0; i<m; ++i) System.out.print((2*i+1)+", ");
         for (int i=0; i<m; ++i) System.out.print(2*i+", ");
      }
      else {
         for (int i=0; i<m; ++i)
            System.out.print((m-1+2*i)%(2*m)+", ");
         for (int i=m-1; i>=0; --i)
            System.out.print((3*m-2*i)%(2*m)+", ");
      }
      if (n!=2*m) System.out.println(2*m);
      else System.out.println();
      end=System.currentTimeMillis();
      System.out.println(n+": Elapsed time: "+(end-begin));
   }
}
```

2 跳马问题的快速算法：

```
public class Knight  {  static PrintWriter out;
   static final int[][] move=
      {{2,1},{1,2},{-1,2},{-2,1},{-2,-1},{-1,-2},{1,-2},{2,-1}};
   static final int[] xychange={0, 2, 1, 8, 7, 6, 5, 4, 3};
   static final int[] xchange={0, 8, 7, 6, 5, 4, 3, 2, 1};
   static final int[] ychange={0, 4, 3, 2, 1, 8, 7, 6, 5};
   public static final int[][] threeten={
      {0, 27, 24, 17, 4, 19, 22, 13, 10, 7},
      {25, 16, 29, 2, 23, 14, 5, 8, 21, 12},
      {28, 1, 26, 15, 18, 3, 20, 11, 6, 9}};
   public static final int[][] dthreeten={
      {1, 8, 7, 1, 2, 1, 7, 7, 8, 7},
      {2, 3, 6, 2, 6, 7, 2, 2, 6, 6},
      {3, 3, 5, 6, 4, 5, 3, 3, 4, 5}};
   public static final int[][] threetwelve={
      {0, 33, 30, 21, 24, 27, 14, 19, 16, 7, 12, 9},
      {31, 22, 35, 2, 29, 20, 25, 4, 13, 10, 17, 6},
      {34, 1, 32, 23, 26, 3, 28, 15, 18, 5, 8, 11}};
   public static final int[][] dthreetwelve={
      {1, 8, 7, 7, 2, 1, 1, 7, 2, 1, 7, 7},
      {2, 2, 6, 2, 6, 6, 7, 2, 6, 2, 7, 6},
      {3, 3, 5, 4, 4, 3, 6, 4, 5, 3, 4, 5}};
   public static final int[][] sthreefour={
      {0, 3, 6, 9},
      {11, 8, 1, 4},
      {2, 5, 10, 7}};
   public static final int[][] dsthreefour={
      {2, 2, 1, 8},
      {0, 3, 7, 7},
      {4, 4, 6, 6}};
   public static final int[][] rdsthreefour={
      {0, 8, 8, 7},
      {2, 2, 6, 6},
      {3, 3, 4, 5}};
   public static final int[][] fivesix={
      {5, 18, 29, 10, 3, 20},
      {28, 9, 4, 19, 24, 11},
      {17, 6, 15, 0, 21, 2},
      {14, 27, 8, 23, 12, 25},
      {7, 16, 13, 26, 1, 22}};
   public static final int[][] dfivesix={
      {1, 2, 1, 2, 7, 8},
      {3, 3, 6, 3, 1, 8},
      {4, 8, 8, 1, 1, 5},
      {3, 5, 5, 4, 7, 7},
      {3, 5, 6, 6, 4, 6}};
   public static final int[][] fiveeight={
      {30, 25, 36, 23, 4, 17, 8, 13},
      {37, 22, 29, 18, 9, 14, 3, 16},
      {26, 31, 24, 35, 0, 5, 12, 7},
      {21, 38, 33, 28, 19, 10, 15, 2},
      {32, 27, 20, 39, 34, 1, 6, 11}};
   public static final int[][] dfiveeight={
      {1, 8, 7, 8, 1, 7, 7, 7},
      {1, 3, 6, 1, 1, 1, 6, 6},
      {1, 8, 5, 5, 1, 1, 4, 5},
      {4, 2, 2, 5, 7, 2, 4, 5},
      {3, 3, 6, 4, 5, 3, 4, 5}};
   public static final int[][] sfivefour={
      {0, 5, 18, 13},
      {19, 14, 9, 4},
      {6, 1, 12, 17},
      {15, 10, 3, 8},
      {2, 7, 16, 11}};
  public static final int[][] dsfivefour={
      {1, 8, 7, 7},
      {0, 8, 8, 6},
      {1, 8, 4, 5},
      {2, 2, 4, 5},
      {3, 3, 4, 5}};
  public static final int[][] sixsix={
      {18, 31, 4, 25, 16, 13},
      {5, 26, 17, 14, 3, 24},
      {32, 19, 30, 23, 12, 15},
      {29, 6, 27, 0, 9, 2},
      {20, 33, 8, 11, 22, 35},
      {7, 28, 21, 34, 1, 10}};
  public static final int[][] dsixsix={
      {1, 8, 7, 7, 7, 7},
      {1, 1, 6, 2, 6, 6},
      {1, 8, 5, 3, 4, 5},
      {3, 8, 8, 1, 1, 5},
      {2, 2, 3, 4, 5, 6},
      {3, 5, 3, 3, 4, 6}};
  public static final int[][] sixseven={
      {40, 9, 32, 19, 30, 7, 4},
      {33, 20, 41, 8, 5, 18, 29},
      {10, 39, 34, 31, 14, 3, 6},
      {21, 36, 23, 0, 17, 28, 15},
      {24, 11, 38, 35, 26, 13, 2},
      {37, 22, 25, 12, 1, 16, 27}};
  public static final int[][] dsixseven={
      {2, 8, 7, 7, 8, 7, 7},
      {2, 8, 1, 6, 2, 6, 6},
      {1, 5, 1, 5, 2, 4, 5},
      {1, 8, 7, 1, 4, 4, 8},
      {2, 2, 5, 6, 2, 5, 5},
      {3, 4, 3, 3, 3, 5, 5}};
  public static final int[][] sixeight={
      {31, 26, 15, 40, 37, 24, 13, 4},
      {16, 45, 30, 25, 14, 5, 38, 23},
      {27, 32, 41, 36, 39, 22, 3, 12},
      {44, 17, 46, 29, 0, 11, 6, 9},
      {33, 28, 19, 42, 35, 8, 21, 2},
      {18, 43, 34, 47, 20, 1, 10, 7}};
  public static final int[][] dsixeight={
      {1, 8, 7, 8, 2, 7, 7, 7},
      {1, 1, 6, 6, 6, 1, 7, 6},
      {1, 8, 1, 4, 5, 3, 4, 5},
      {4, 8, 1, 5, 1, 3, 1, 8},
      {2, 3, 2, 7, 5, 3, 5, 5},
      {3, 5, 3, 4, 3, 3, 5, 6}};
  public static final int[][] ssixfour={
      {0, 3, 22, 19},
      {23, 20, 11, 2},
      {4, 1, 18, 21},
      {15, 12, 7, 10},
      {8, 5, 14, 17},
      {13, 16, 9, 6}};
  public static final int[][] dssixfour={
      {1, 8, 7, 7},
      {0, 2, 8, 6},
      {1, 3, 4, 5},
      {1, 8, 7, 5},
      {2, 2, 6, 5},
      {3, 3, 4, 5}};
  public static final int[][] seveneight={
      {33, 30, 7, 18, 53, 44, 5, 16},
      {8, 19, 32, 55, 6, 17, 50, 43},
      {31, 34, 29, 52, 45, 54, 15, 4},
      {20, 9, 40, 35, 0, 51, 42, 49},
      {25, 28, 23, 46, 41, 36, 3, 14},
      {10, 21, 26, 39, 12, 1, 48, 37},
      {27, 24, 11, 22, 47, 38, 13, 2}};
  public static final int[][] dseveneight={
      {1, 8, 7, 7, 1, 8, 7, 7},
      {1, 8, 6, 1, 6, 6, 8, 6},
      {3, 2, 5, 4, 8, 6, 4, 5},
      {1, 8, 2, 2, 1, 6, 4, 5},
      {2, 4, 8, 1, 3, 2, 4, 5},
      {2, 2, 7, 5, 2, 2, 4, 7},
      {4, 5, 3, 5, 3, 6, 4, 5}};
  public static final int[][] ssevenfour={
      {0, 13, 26, 15},
      {27, 16, 3, 12},
      {4, 1, 14, 25},
      {17, 24, 11, 2},
      {8, 5, 20, 23},
      {21, 18, 7, 10},
      {6, 9, 22, 19}};
  public static final int[][] dssevenfour={
      {1, 1, 7, 7},
      {0, 8, 7, 6},
      {1, 2, 4, 5},
      {1, 3, 4, 5},
      {1, 8, 7, 6},
      {2, 2, 6, 5},
      {3, 3, 4, 5}};
  public static final int[][] seightthree={
      {0, 13, 22},
      {23, 16, 1},
      {14, 21, 12},
      {17, 2, 15},
      {20, 11, 18},
      {5, 8, 3},
      {10, 19, 6},
      {7, 4, 9}};
  public static final int[][] dseightthree={
      {2, 8, 7},
      {0, 8, 8},
      {2, 4, 5},
      {2, 1, 5},
      {4, 4, 8},
      {2, 1, 8},
      {4, 5, 7},
      {4, 5, 6}};
  public static final int[][] sninefour={
      {0, 17, 34, 19},
      {35, 20, 3, 16},
      {4, 1, 18, 33},
      {21, 24, 15, 2},
      {14, 5, 32, 23},
      {25, 22, 13, 6},
      {10, 7, 28, 31},
      {29, 26, 9, 12},
      {8, 11, 30, 27}};
  public static final int[][] dsninefour={
      {1, 1, 7, 7},
      {0, 8, 7, 6},
      {1, 2, 4, 5},
      {1, 8, 4, 5},
      {3, 2, 4, 6},
      {1, 3, 6, 7},
      {1, 8, 7, 5},
      {2, 2, 6, 5},
      {3, 3, 4, 5}};
  public static final int[][] ssixfive={
      {6, 27, 14, 21, 4, 9},
      {15, 22, 5, 8, 13, 20},
      {28, 7, 26, 19, 10, 3},
      {23, 16, 1, 12, 25, 18},
      {0, 29, 24, 17, 2, 11}};
  public static final int[][] dssixfive={
      {1, 8, 7, 7, 7, 8},
      {1, 8, 6, 3, 6, 6},
      {1, 3, 5, 3, 1, 5},
      {2, 2, 2, 4, 6, 6},
      {3, 0, 3, 3, 4, 6}};
  public static final int[][] seightfive={
      {36, 3, 14, 29, 34, 5, 12, 21},
      {15, 30, 35, 4, 13, 20, 25, 6},
      {2, 37, 18, 33, 28, 9, 22, 11},
      {31, 16, 1, 38, 19, 24, 7, 26},
      {0, 39, 32, 17, 8, 27, 10, 23}};
  public static final int[][] dseightfive={
      {1, 2, 7, 7, 7, 2, 7, 8},
      {1, 8, 6, 3, 6, 3, 1, 8},
      {4, 2, 2, 4, 5, 1, 1, 5},
      {2, 2, 6, 7, 4, 4, 7, 7},
      {3, 0, 4, 5, 4, 5, 4, 6}};
  public static int[][] ktour(int m, int n){ int[][] ans=new int[m][n];
      if (m%2==1 && n%2==1){
          System.out.println("no answer!"); return null;
      }
      if (m<=9 || n<=9){
          if (m>n){ System.out.println("please make m<=n"); return null; }
          if (m<3 || m==4 || (m==3 && n<=8)){
             System.out.println("no answer!"); return null;
          }
          if (m==3){
             if (n==10) ans=dthreeten;
             else if (n==12) ans=dthreetwelve;
             else if (n%4==2) union3(dthreeten, ans, 10, (n-10)/4);
             else  union3(dthreetwelve, ans, 12, (n-12)/4);
          }
          else if (m==5){
             if (n==6) ans=dfivesix;
             else if (n==8) ans=dfiveeight;
             else if (n%4==2) union1(dfivesix, ans, dsfivefour, 5, 6, 4, (n-6)/4);
             else  union1(dfiveeight, ans, dsfivefour, 5, 8, 4, (n-8)/4);
          }
          else if (m==6){
```

```java
            if (n==6) ans=dsixsix;
            else if (n==7) ans=dsixseven;
            else if (n==8) ans=dsixeight;
            else if (n%4==1){ int[][] dsixfive=xytr(dfivesix, 5, 6);
                union1(dsixfive, ans, dssixfour, 6, 5, 4, (n-5)/4); }
            else if (n%4==2) union1(dsixsix, ans, dssixfour, 6, 6, 4, (n-6)/4);
            else if (n%4==3) union1(dsixseven, ans, dssixfour, 6, 7, 4, (n-7)/4);
            else  union1(dsixeight, ans, dssixfour, 6, 8, 4, (n-8)/4);
        }
        else if (m==7){
            if (n==8) ans=dseveneight;
            else if (n%4==2){
                int[][] dsevensix=xtr(xytr(dsixseven, 6, 7), 7, 6);
                union1(dsevensix, ans, dssevenfour, 7, 6, 4, (n-6)/4); }
            else  union1(dseveneight, ans, dssevenfour, 7, 8, 4, (n-8)/4);
        }
        else if (m==8){
            if (n%3==2){ int[][] deightfive=xytr(dfiveeight, 5, 8);
                union1(deightfive, ans, dseightthree, 8, 5, 3, (n-5)/3); }
            else if (n%3==1){
                int[][] deightseven=xtr(xytr(dseveneight, 7, 8), 8, 7);
                union1(deightseven, ans, dseightthree, 8, 7, 3, (n-7)/3); }
            else { int[][] deightsix=xtr(xytr(dsixeight, 6, 8), 8, 6);
                union1(deightsix, ans, dseightthree, 8, 6, 3, (n-6)/3); }
        }
        else if (n%4==0){  int[][] dnineeight=xytr(ktour(8, 9), 8, 9);
            union1(dnineeight, ans, dsninefour, 9, 8, 4, (n-8)/4); }
        else {  int[][] dninesix=xytr(ktour(6, 9), 6, 9);
            union1( dninesix, ans, dsninefour, 9, 6, 4, (n-6)/4); }
    }
    else if (m%2==1){ System.out.println("please make m even!"); return null; }
    else { int[][] sblockh=new int[5][m], sblockv=null;;
        if (m%4==0) union1(dseightfive, sblockh, dsfivefour, 5, 8, 4, (m-8)/4);
        else  union1(dssixfive, sblockh, dsfivefour, 5, 6, 4, (m-6)/4);
        sblockv=xtr(xytr(sblockh, 5, m), m, 5);
        if (n%5==0)
            union1(xytr(ktour(5, m), 5, m), ans, sblockv, m, 5, 5, (n-5)/5);
        else if (n%5==1)
            union1(xtr(xytr(ktour(6, m), 6, m), m, 6),
                ans, sblockv, m, 6, 5, (n-6)/5);
        else if (n%5==2)
            union1(xtr(xytr(ktour(7, m), 7, m), m, 7),
                ans, sblockv, m, 7, 5, (n-7)/5);
        else if (n%5==3){ int[][] ini = xytr(ktour(8, m), 8, m);
            ini=xtr(ini, m, 8); ini=ytr(ini, m, 8);
            union1(ini, ans, sblockv, m, 8, 5, (n-8)/5); }
        else union1(xytr(ktour(9, m), 9, m),
                ans, sblockv, m, 9, 5, (n-9)/5);
    }
    return ans;
}
public static void union3(int[][] a, int[][] res, int len, int k){
    for (int i=0; i<3; ++i) for (int j=0; j<len; ++j) res[i][j]=a[i][j];
    for (int x=0; x<k; ++x)
        if (x%2==0){ res[2][len-1+4*x]=4;
            for (int i=0; i<3; ++i) for (int j=0; j<4; ++j)
                res[i][len+4*x+j]=dsthreefour[i][j];
            res[1][len+4*x]=6;
        }
        else { res[0][len-2+4*x]=2;
            for (int i=0; i<3; ++i) for (int j=0; j<4; ++j)
                res[i][len+4*x+j]=rdsthreefour[i][j];
            res[0][len+4*x]=8;
        }
}
public static void union1(int[][] a, int[][] res, int[][] b,
        int xlen, int ylen1, int ylen2, int k){
    for (int i=0; i<xlen; ++i) for (int j=0; j<ylen1; ++j) res[i][j]=a[i][j];
    for (int x=0; x<k; ++x){
        res[2][ylen1-1+ylen2*x]=4;
        for (int i=0; i<xlen; ++i) for (int j=0; j<ylen2; ++j)
            res[i][ylen1+ylen2*x+j]=b[i][j];
        res[1][ylen1+ylen2*x]=6;
    }
}
public static int[][] tr(int[][] board, int m, int n){
    int[][] ans=new int[m][n]; int x=0, y=0;
    for (int i=1; i<m*n; ++i){
        int x1=x+move[board[x][y]-1][0], y1=y+move[board[x][y]-1][1];
        ans[x1][y1]=i; x=x1; y=y1; }
    return ans;
}
public static int[][] xytr(int[][] board, int m, int n){
    int[][] ans=new int[n][m];
    for (int i=0; i<n; ++i) for (int j=0; j<m; ++j)
        ans[i][j]=xychange[board[j][i]];
    return ans;
}
public static int[][] xtr(int[][] board, int m, int n){
    int[][] ans=new int[m][n];
    for (int i=0; i<m; ++i) for (int j=0; j<n; ++j)
        ans[i][j]=xchange[board[i][n-1-j]];
    return ans;
}
public static int[][] ytr(int[][] board, int m, int n){
    int[][] ans=new int[m][n];
    for (int i=0; i<m; ++i) for (int j=0; j<n; ++j)
        ans[i][j]=ychange[board[m-1-i][j]];
    return ans;
}
public static void output(int[][] board, int m, int n, int k){
    if (k==4){
        for (int i=0; i<m; ++i){
            for (int j=0; j<n; ++j){
                if (board[i][j]<10) out.print("000");
                else if (board[i][j]<100) out.print("00");
                else if (board[i][j]<1000) out.print("0");
                    out.print(board[i][j]+", "); }
            out.println(); }
        out.println(); }
    else if (k==3){
        for (int i=0; i<m; ++i){
            for (int j=0; j<n; ++j){
                if (board[i][j]<10) out.print("00");
                else if (board[i][j]<100) out.print("0");
                    out.print(board[i][j]+", "); }
            out.println(); }
        out.println(); }
    else {
        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 static void main(String[] args){ PrintWriter fout=null;
    try{ fout=new PrintWriter(new FileWriter("out-100-100.txt")); }
    catch(IOException e){ System.out.println("Error!"); }
    int m=100, n=100; Knight.out=fout;
    long begin=System.currentTimeMillis();
    int[][] b=ktour(m, n);
    if (b!=null) output(tr(b, m, n), m, n, 4);
    long end=System.currentTimeMillis();
    fout.println(m+", "+n+": Elapsed time: "+(end-begin));
```

```
        fout.close();
    }
}
```

```
000, 143, 128, 139, 482, 479, 122, 149, 472, 477, 464, 235, 244, 457, 462, 449, 330, 339, 442, 447, 434, 425,
129, 140, 483, 120, 123, 138, 481, 478, 465, 236, 471, 476, 463, 450, 331, 456, 461, 448, 435, 426, 441, 446,
142, 001, 144, 127, 480, 121, 148, 473, 150, 475, 234, 243, 458, 245, 460, 329, 338, 443, 340, 445, 424, 433,
133, 130, 141, 004, 119, 124, 137, 466, 241, 468, 237, 470, 451, 336, 453, 332, 455, 436, 431, 438, 427, 440,
002, 145, 132, 135, 126, 147, 118, 151, 474, 239, 242, 233, 246, 459, 334, 337, 328, 341, 444, 429, 432, 423,
131, 134, 003, 146, 005, 136, 125, 240, 467, 152, 469, 238, 335, 452, 247, 454, 333, 430, 437, 342, 439, 428,
012, 027, 014, 023, 010, 117, 006, 155, 168, 159, 232, 153, 250, 263, 254, 327, 248, 345, 358, 349, 422, 343,
015, 024, 011, 030, 007, 022, 019, 160, 163, 154, 167, 158, 255, 258, 249, 262, 253, 350, 353, 344, 357, 348,
028, 013, 026, 017, 020, 009, 116, 169, 156, 165, 162, 231, 264, 251, 260, 257, 326, 359, 346, 355, 352, 421,
025, 016, 029, 008, 031, 018, 021, 164, 161, 170, 157, 166, 259, 256, 265, 252, 261, 354, 351, 360, 347, 356,
038, 053, 040, 049, 036, 115, 032, 173, 186, 177, 230, 171, 268, 281, 272, 325, 266, 363, 376, 367, 420, 361,
041, 050, 037, 056, 033, 048, 045, 178, 181, 172, 185, 176, 273, 276, 267, 280, 271, 368, 371, 362, 375, 366,
054, 039, 052, 043, 046, 035, 114, 187, 174, 183, 180, 229, 282, 269, 278, 275, 324, 377, 364, 373, 370, 419,
051, 042, 055, 034, 057, 044, 047, 182, 179, 188, 175, 184, 277, 274, 283, 270, 279, 372, 369, 378, 365, 374,
064, 079, 066, 075, 062, 113, 058, 191, 204, 195, 228, 189, 286, 299, 290, 323, 284, 381, 394, 385, 418, 379,
067, 076, 063, 082, 059, 074, 071, 196, 199, 190, 203, 194, 291, 294, 285, 298, 289, 386, 389, 380, 393, 384,
080, 065, 078, 069, 072, 061, 112, 205, 192, 201, 198, 227, 300, 287, 296, 293, 322, 395, 382, 391, 388, 417,
077, 068, 081, 060, 083, 070, 073, 200, 197, 206, 193, 202, 295, 292, 301, 288, 297, 390, 387, 396, 383, 392,
090, 105, 092, 101, 088, 111, 084, 209, 222, 213, 226, 207, 304, 317, 308, 321, 302, 399, 412, 403, 416, 397,
093, 102, 089, 108, 085, 100, 097, 214, 217, 208, 221, 212, 309, 312, 303, 316, 307, 404, 407, 398, 411, 402,
106, 091, 104, 095, 098, 087, 110, 223, 210, 219, 216, 225, 318, 305, 314, 311, 320, 413, 400, 409, 406, 415,
103, 094, 107, 086, 109, 096, 099, 218, 215, 224, 211, 220, 313, 310, 319, 306, 315, 408, 405, 414, 401, 410,

22, 22:    Elapsed time: 0
```

搜索 5×N 棋盘结构化巡游的程序：

```java
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 class Knight2c extends Knight2 {
   public Knight2c(int nn){
      super(5, nn, 2, nn/2); level=1; board[2][1]=board[2][n-2]=-1;
      board[2][0]=board[2][n-1]=-1;
   }
```

```
public boolean target(){
   return level==m*n && edge(x, y, m/2, n/2); }
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;
      if (x==1 && (y==2 || y==n-3)){ x+=2;
         level+=4; board[x][y]=level; }
      else if (x==3 && (y==2 || y==n-3)){ x-=2;
         level+=4; board[x][y]=level; }
      else if (x==0 && (y==1 || y==n-2)){  x+=4;
         level+=2; board[x][y]=level; }
      else if (x==4 && (y==1 || y==n-2)){  x-=4;
         level+=2; board[x][y]=level; }
      return true; }
    return false;
}
public void up(){
   if (x==1 && (y==2 || y==n-3)){ board[x][y]=0; x+=2;
level-=4; }
   else if (x==3 && (y==2 || y==n-3)){
      board[x][y]=0; x-=2; level-=4;  }
   else if (x==0 && (y==1 || y==n-2)){
      board[x][y]=0; x+=4; level-=2;  }
   else if (x==4 && (y==1 || y==n-2)){
      board[x][y]=0; x-=4; level-=2; }
    int i=parent[x][y]; board[x][y]=0;
    x-=move[i][0]; y-=move[i][1]; --level;
}
public void output(){
   board[2][1]=(board[1][2]+board[3][2])/2;
   board[2][0]=(board[0][1]+board[4][1])/2;
   board[2][n-2]=(board[1][n-3]+board[3][n-3])/2;
   board[2][n-1]=(board[0][n-2]+board[4][n-2])/2;
   board[0][0]=(board[1][2]+board[2][1])/2;
   board[4][0]=(board[3][2]+board[2][1])/2;
   board[0][n-1]=(board[2][n-2]+board[1][n-3])/2;
   board[4][n-1]=(board[3][n-3]+board[2][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]-1)+", "); }
      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-2c-10.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int n=10; Knight2.out=fout;
      long begin=System.currentTimeMillis();
      BackSearch x=new BackSearch(new Knight2c(n), 1);
      x.depthfirst(); fout.println(x.getnum());
      long end=System.currentTimeMillis();
      fout.println(5+", "+n+": Elapsed time: "+(end-begin));
      fout.close();
   }
```

```
05, 18, 29, 10, 03, 20,
28, 09, 04, 19, 24, 11,
17, 06, 15, 00, 21, 02,
14, 27, 08, 23, 12, 25,
07, 16, 13, 26, 01, 22,
5, 6:    Elapsed time: 0

30, 25, 36, 23, 04, 17, 08, 13,
37, 22, 29, 18, 09, 14, 03, 16,
26, 31, 24, 35, 00, 05, 12, 07,
21, 38, 33, 28, 19, 10, 15, 02,
32, 27, 20, 39, 34, 01, 06, 11,
5, 8:    Elapsed time: 0
```

搜索 M×N(M, N>=6)棋盘结构化巡游的程序：

```
public class Knight2b extends Knight2 {
   static final int[][] cmove={{1,1},{-1,-1}, {-1,1}, {1,-1}};
   int[][] point;
   public Knight2b(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;
      point[0][1]=-8; point[0][2]=-7; point[1][0]=-3;
      point[2][0]=-4; point[0][n-3]=-2; point[0][n-2]=-1;
      point[1][n-1]=-6; point[2][n-1]=-5; point[m-1][1]=-5;
      point[m-1][2]=-6; point[m-3][0]=-1; point[m-2][0]=-2;
      point[m-1][n-3]=-3; point[m-1][n-2]=-4;
      point[m-3][n-1]=-8; point[m-2][n-1]=-7;
   }
   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];
         if (k>0){ --k; x+=cmove[k][0]; y+=cmove[k][1];
            level+=2; board[x][y]=level; }
         else if (k<0){ k=-k-1; x+=move[k][0]; y+=move[k][1];
            ++level; board[x][y]=level;  }
         return true; }
       return false;
   }
   public void up(){ int k=point[x][y];
       if (k>0){ --k; board[x][y]=0; x+=cmove[k][0];
         y+=cmove[k][1]; level-=2; }
       else if (k<0){ k=-k-1; board[x][y]=0;
         x+=move[k][0]; y+=move[k][1]; --level; }
       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]-1)+", "); }
         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-2b-6-7.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      int m=6, n=7; Knight2.out=fout;
      long begin=System.currentTimeMillis();
```

```
        BackSearch x=new BackSearch(new Knight2b(m, n), 1);
        x.depthfirst();
        long end=System.currentTimeMillis();
        fout.println(m+", "+n+": Elapsed time: "+(end-begin));
        fout.close();
    }
}
```

```
18, 31, 04, 25, 16, 13,
05, 26, 17, 14, 03, 24,
32, 19, 30, 23, 12, 15,
29, 06, 27, 00, 09, 02,
20, 33, 08, 11, 22, 35,
07, 28, 21, 34, 01, 10,
6, 6:    Elapsed time: 0

24, 15, 32, 35, 22, 13, 04,
31, 36, 23, 14, 05, 34, 21,
16, 25, 30, 33, 20, 03, 12,
37, 28, 39, 00, 11, 06, 09,
40, 17, 26, 29, 08, 19, 02,
27, 38, 41, 18, 01, 10, 07,
6, 7:    Elapsed time: 0

31, 26, 15, 40, 37, 24, 13, 04,
16, 45, 30, 25, 14, 05, 38, 23,
27, 32, 41, 36, 39, 22, 03, 12,
44, 17, 46, 29, 00, 11, 06, 09,
33, 28, 19, 42, 35, 08, 21, 02,
18, 43, 34, 47, 20, 01, 10, 07,
6, 8:    Elapsed time: 0

45, 16, 33, 38, 11, 14, 05, 26,
34, 39, 46, 15, 06, 27, 10, 13,
17, 44, 55, 32, 37, 12, 25, 04,
54, 35, 40, 47, 00, 07, 28, 09,
43, 18, 51, 36, 31, 22, 03, 24,
50, 53, 20, 41, 48, 01, 08, 29,
19, 42, 49, 52, 21, 30, 23, 02,
7, 8:    Elapsed time: 94739
```

搜索可扩展巡游的程序：

```
public class Knight2d extends Knight2 {
    public Knight2d(int mm, int nn){ super(mm, nn, 0, 0);
board[1][0]=m*n; }
    public boolean target(){
        return level==m*n-1 && edge(x, y, 1, 0); }
```

```
00, 03, 06, 09,
11, 08, 01, 04,
02, 05, 10, 07,
3, 4:    Elapsed time: 16

00, 05, 18, 13,
19, 14, 09, 04,
06, 01, 12, 17,
15, 10, 03, 08,
02, 07, 16, 11,
5, 4:    Elapsed time: 3

00, 03, 22, 19,
23, 20, 11, 02,
04, 01, 18, 21,
15, 12, 07, 10,
08, 05, 14, 17,
13, 16, 09, 06,
6, 4:    Elapsed time: 10

00, 13, 26, 15,
27, 16, 03, 12,
04, 01, 14, 25,
17, 24, 11, 02,
08, 05, 20, 23,
21, 18, 07, 10,
06, 09, 22, 19,
7, 4:    Elapsed time: 47

00, 13, 22,
23, 16, 01,
14, 21, 12,
17, 02, 15,
20, 11, 18,
05, 08, 03,
10, 19, 06,
07, 04, 09,
8, 3:    Elapsed time: 0

00, 17, 34, 19,
35, 20, 03, 16,
04, 01, 18, 33,
21, 24, 15, 02,
14, 05, 32, 23,
25, 22, 13, 06,
10, 07, 28, 31,
29, 26, 09, 12,
08, 11, 30, 27,
9, 4:    Elapsed time: 18372
```

搜索 5×N 棋盘可扩展巡游的程序：

```
public class Knight3 implements Mono1 { 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 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 class Knight2e extends Knight3 {
   public Knight2e(int nn){
      super(5, nn, 4, 0); board[4][1]=5*nn; board[1][0]=-1;
      board[2][n-2]=-1; board[2][n-1]=-1;
   }
   public boolean target(){
      return level==m*n-1 && edge(x, y, 4, 1); }
   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;
      if (x==1 && y==n-3){ x+=2;
         level+=4; board[x][y]=level; }
      else if (x==3 && y==n-3){ x-=2;
         level+=4; board[x][y]=level; }
       else if (x==0 && y==n-2){  x+=4;
         level+=2; board[x][y]=level; }
       else if (x==4 && y==n-2){  x-=4;
         level+=2; board[x][y]=level; }
       else if (x==0 && y==2){ ++level;
         x=1; y=0; board[x][y]=level;
       }
```

```
    }
    public void up(){
       if (x==1 && y==n-3){ board[x][y]=0; x+=2;   level-=4; }
       else if (x==3 && y==n-3){ board[x][y]=0; x-=2; level-=4;  }
       else if (x==0 && y==n-2){ board[x][y]=0; x+=4; level-=2;  }
       else if (x==4 && y==n-2){ board[x][y]=0; x-=4; level-=2; }
       else if (x==1 && y==0){ board[x][y]=-1; --level; x=0; y=2;}
        int i=parent[x][y]; board[x][y]=0;
        x-=move[i][0]; y-=move[i][1]; --level;
    }
    public void output(){
       board[2][n-2]=(board[1][n-3]+board[3][n-3])/2;
       board[2][n-1]=(board[0][n-2]+board[4][n-2])/2;
       board[0][n-1]=(board[2][n-2]+board[1][n-3])/2;
       board[4][n-1]=(board[3][n-3]+board[2][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]-1)+", "); }
          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("outa-2e-8.txt")); }
       catch(IOException e){ System.out.println("Error!"); }
       int n=8; Knight3.out=fout;
       long begin=System.currentTimeMillis();
       BackSearch2 x=new BackSearch2(new Knight2e(n), 1, 8);
       x.reordering();
       long end=System.currentTimeMillis();
       fout.println(5+", "+n+": Elapsed time: "+(end-begin));
       fout.close();
    }
}
```

```
06, 27, 14, 21, 04, 09,
15, 22, 05, 08, 13, 20,
28, 07, 26, 19, 10, 03,
23, 16, 01, 12, 25, 18,
00, 29, 24, 17, 02, 11,
5, 6:    Elapsed time: 94
```

```
36, 03, 14, 29, 34, 05, 12, 21,
15, 30, 35, 04, 13, 20, 25, 06,
02, 37, 18, 33, 28, 09, 22, 11,
31, 16, 01, 38, 19, 24, 07, 26,
00, 39, 32, 17, 08, 27, 10, 23,
5, 8:    Elapsed time: 0
```

思考题 **3.1**：实现算法 **3.1** 并用来求解连续邮资问题。

```
public interface Valued { int val(); void setval(int k); }
public interface Check { boolean target();  void output();
public interface WNode extends Check, Valued {
   int subnum(); WNode down(int i); }
public class WLink implements Comparable {
   public WNode prob; public WLink next;
   public WLink(WNode p, WLink n){ prob=p; next=n; }
   public void output(){ if (next!=null) next.output();
      prob.output(); }
   public Object cloneself(){
      return next==null ? new WLink(prob, null) :
         new WLink(prob, (WLink)next.cloneself()); }
   public int compareTo(Object obj){
      return prob.val()-((WLink)obj).prob.val(); }
}
public class TreeOptBruteSearch {
   WLink head, ans; WNode node, lnode;
   public TreeOptBruteSearch(WNode p){
      head=new WLink(p, null); node=p; }
   public void depthfirst(){
      WNode p=head.prob, sub=null;
      if (p.target())
         if (lnode==null || p.val()<lnode.val()){
            ans=(WLink)head.cloneself(); lnode=p; }
      int n=p.subnum();
      for (int i=0; i<n; ++i)
         if ((sub=p.down(i))!=null){
            head=new WLink(sub, head);
            depthfirst(); head=head.next; }
   }
   public WLink outans(){ return ans; }
}
public class Stamp1  implements  WNode {
   static byte m, n; static int maxv=2000;
   static PrintWriter out;
   byte[] least=new byte[maxv+1];
   int name, level, curv;
   public Stamp1(int n, int lev){ name=n; level=lev; }
   public boolean target(){ return level==n; }
   public int subnum(){ return level<n ? curv-name+1 : 0; }
   public  WNode down(int i){
      Stamp1 ans=new Stamp1(name+i+1, level+1);
      if (level==0){ Arrays.fill(ans.least, (byte)(m+1));
         ans.least[0]=0; }
      else System.arraycopy(least, 0, ans.least, 0, maxv);
      for (int j=0; j<=Math.min(maxv-1, (m-1)*name); ++j)
         if (ans.least[j]<m)
            for (byte k=1; k<=m-ans.least[j]; ++k){
               int loc=Math.min(maxv-1, j+k*ans.name);
               ans.least[loc]=
                  ans.least[loc]<(byte)(ans.least[j]+k) ?
                     ans.least[loc] : (byte)(ans.least[j]+k);
            }
      int j=curv+1;
      while(j<=maxv && ans.least[j]<=m) ++j;
      ans.curv=j-1;
      return ans;
   }
   public  int val(){   return -curv; }
   public  void setval(int k){ curv=k; }
   public void output(){ out.println(name); }
   public static void main(String[] args) {
      PrintWriter fout=null;
       try{ fout=new PrintWriter(new FileWriter("out-7-6.txt")); }
       catch(IOException e){ System.out.println("Error!"); }
       long begin=System.currentTimeMillis();
       byte m=7, n=6; Stamp1.m=m; Stamp1.n=n; Stamp1.out=fout;
       TreeOptBruteSearch x=new TreeOptBruteSearch(new Stamp1(0, 0));
       x.depthfirst(); WLink a=x.outans(); a.output();
       fout.println("maxvalue: "+((Stamp1)a.prob).curv);
       long end=System.currentTimeMillis();
       fout.println(m+", "+n+": Elapsed time: "+(end-begin));
       fout.close();
   }
}
```

```
0, 1, 3, 11, 15, 32
maxvalue: 70
4, 5: Elapsed time: 16

0, 1, 4, 9, 16, 38, 49
```

```
maxvalue: 108
4, 6:  Elapsed time: 374

0, 1, 4, 9, 24, 35, 49, 51
maxvalue: 162
4, 7:  Elapsed time: 13323

0, 1, 4, 9, 31, 51
maxvalue: 126
5, 5:  Elapsed time: 90

0, 1, 4, 13, 24, 56, 61
maxvalue: 211
5, 6:  Elapsed time: 4496

0, 1, 7, 12, 43, 52
maxvalue: 216
6, 5:  Elapsed time: 374

0, 1, 7, 11, 48, 83, 115
maxvalue: 388
6, 6:  Elapsed time: 40189

0, 1, 8, 11, 64, 102
maxvalue: 345
7, 5:  Elapsed time: 1575

0, 1, 7, 12, 64, 113
193
maxvalue: 664
7, 6:  Elapsed time: 331179

0, 1, 9, 15, 78, 115
maxvalue: 512
8, 5:  Elapsed time: 5851

0, 1, 9, 23, 108, 181
maxvalue: 797
9, 5:  Elapsed time: 19859

0, 1, 8, 27, 119, 194
maxvalue: 1055
10, 5:  Elapsed time: 63274
```

思考题 3.2：用算法 3.1 来求解 0-1 背包问题。

```
public class KnapBestNodeCom implements WNode {
   public int curnode; int cval, cw;
   public KnapBestNodeCom(int cn, int cv, int w){
      curnode=cn; cval=cv; cw=w; }
   public boolean target(){ return cw<=Knap.wlimit; }
   public int subnum(){ return Knap.n-curnode-1; }
   public WNode down(int i){ KnapBestNodeCom ans=null;
      int cn=curnode+i+1, w=cw+Knap.weight[cn],
      v=cval+Knap.value[cn];
      if (w<=Knap.wlimit) ans=new KnapBestNodeCom(cn, v, w);
      return ans;
   }
   public void output(){ Knap.out.print(curnode+", ");  }
   public  int val(){   return -cval; }
   public  void setval(int k){ cval=k; }…….
```

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

-1, 0, 1, 2, 3, 5, 8, 9, 10, 12, 13, 14, 16, 17, 19, 21, 24, 27, 29,

maxvalue: 2436

weight: 1194

30:   Elapsed time: 17269

思考题 3.3：实现算法 3.2 并用来求解 0-1 背包问题。

```
public interface WMono extends Mono, Valued { Object cloneself(); }
public class BackOptBruteSearch  {
   private WMono prob, ans;
   public BackOptBruteSearch(WMono p)  { prob=p; }
   public void depthfirst(){
      if (prob.target())
         if (ans==null || prob.val()<ans.val())
            ans=(WMono)prob.cloneself();
      int n=prob.subnum();
      for (int i=0; i<n; ++i)
         if (prob.down(i)){
            depthfirst(); prob.up();
         }
   }
   public WMono outans(){ return ans; }
}
public class KnapBestMonoBin implements WMono {
   int level, cval, cw; boolean[] set;
   public KnapBestMonoBin(){ set=new boolean[Knap.n]; }
   public KnapBestMonoBin(boolean[] d, int w, int v){
      set=d; cw=w; cval=v; }
   public boolean target(){
      return level==Knap.n && cw<=Knap.wlimit; }
   public int subnum(){ return level<Knap.n ? 2 : 0; }
   public boolean down(int i){
      boolean ans = i==0 || i==1 &&
                  cw+Knap.weight[level]<=Knap.wlimit;
      if (ans){ set[level] = i!=0;
         if (i!=0){
            cw+=Knap.weight[level];
            cval+=Knap.value[level]; }
         ++level; }
      return ans;
   }
   public void up(){ --level;
      if (set[level]){ cw-=Knap.weight[level];
      cval-=Knap.value[level]; }
   }
   public void output(){
      for (int i=0; i<Knap.n; ++i)
         if (set[i]) Knap.out.print(i+", ");
      Knap.out.println();
      Knap.out.println("Value:  "+cval+"  Weight:  "+cw); }
   public  int val(){  return -cval; }
   public  void setval(int k){ cval=k; }
   public  Object cloneself(){ boolean[] s=new boolean[Knap.n];
      System.arraycopy(set, 0, s, 0, Knap.n);
      return new  KnapBestMonoBin(s, cw, cval);
   }............
```

```
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
0, 1, 2, 3, 5, 8, 9, 10, 12, 13, 14, 16, 17, 19, 21, 24, 27, 29,
Value:   2436    Weight:   1194
30:   Elapsed time: 25896
```

思考题 3.4：实现算法 3.3 并用来求解 0-1 背包问题。

```
public class TreeOptSearch {
   WLink head, ans; WNode node, lnode;
   public TreeOptSearch(WNode p){
      head=new WLink(p, null); node=p; }
   public void depthfirst(){
      WNode p=head.prob, sub=null;
      if (lnode==null || p.val()<lnode.val())
         if (p.target()){
            ans=(WLink)head.cloneself(); lnode=p; }
         else { int n=p.subnum();
            for (int i=0; i<n; ++i)
               if ((sub=p.down(i))!=null){
                  head=new WLink(sub, head);
                  depthfirst(); head=head.next; }
         }
   }
   public WLink outans(){ return ans; }………
public class KnapLeastNodeCom implements WNode {
   public int curnode; int cval, cw;
   public KnapLeastNodeCom(int cn, int cv, int w){
      curnode=cn; cval=cv; cw=w; }
   public boolean target(){
      return cw>=Knap.wall-Knap.wlimit; }
   public int subnum(){ return Knap.n-curnode-1; }
   public WNode down(int i){
      int cn=curnode+i+1, w=cw+Knap.weight[cn],
      v=cval+Knap.value[cn];
      return new KnapLeastNodeCom(cn, v, w);
   }
   public void output(){ Knap.out.print(curnode+", "); }
   public  int val(){   return cval; }
   public  void setval(int k){ cval=k; }……….
```

```
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
-1, 4, 6, 7, 11, 15, 18, 20, 22, 23, 25, 26, 28,
maxvalue: 2436
weight: 1194
30:    Elapsed time: 3760
```

思考题3.5：实现算法3.4并用来求解0-1背包问题。

```
public class TreeOptSearch {
   WLink head, ans; WNode node, lnode;
   public TreeOptSearch(WNode p){
      head=new WLink(p, null); node=p; }
   public void optfirst1(){
      PriorityQueue<WLink> queue=new PriorityQueue<WLink>();
      queue.add(new WLink(node, null));
      while (queue.size()>0){
         WLink cur=queue.poll(); WNode prob=cur.prob;
         if (prob.target()) { ans=cur; return; }
         else { int n=prob.subnum(); WNode sub=null;
            for (int i=0; i<n; ++i)
               if ((sub=prob.down(i))!=null)
                  queue.add(new WLink(sub, cur));
         }
      }
   }
   public WLink outans(){ return ans; }…….
```

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
-1, 1, 3, 5, 6, 7, 8, 10, 13, 14, 15, 17, 18, 19,
maxvalue: 474
weight: 398
20: Elapsed time: 671

30: out of memory!

思考题 **3.6**：实现算法 **3.5** 并用来求解 **0-1** 背包问题。

```
public class BackOptSearch { private WMono prob, ans;
   public BackOptSearch(WMono p) { prob=p;  }
   public void depthfirst(){
      if (ans==null || prob.val()<ans.val())
         if (prob.target()) ans=(WMono)prob.cloneself();
         else { int n=prob.subnum();
            for (int i=0; i<n; ++i)
               if (prob.down(i)){
                  depthfirst(); prob.up(); }
         }
   }
   public WMono outans(){ return ans; }
}
public class KnapLeastMonoBin implements WMono {
   int level, cval, cw; boolean[] set;
   public KnapLeastMonoBin(){
      set=new boolean[Knap.n]; }
   public KnapLeastMonoBin(boolean[] d, int w, int v){
      set=d; cw=w; cval=v; }
   public boolean target(){
      return level==Knap.n && cw>=Knap.wall-Knap.wlimit; }
   public int subnum(){
      return level<Knap.n ? 2 : 0; }
   public boolean down(int i){ set[level] = i!=0;
      if (i!=0){
         cw+=Knap.weight[level];
         cval+=Knap.value[level]; }
      ++level; return true;
   }
   public void up(){ --level;
      if (set[level]){
         cw-=Knap.weight[level];
         cval-=Knap.value[level]; }
   }
   public void output(){
      Knap.out.println("Value:  "+(Knap.vall-cval)+
            "  Weight:  "+(Knap.wall-cw)); }
   public  int val(){   return cval; }
   public  void setval(int k){ cval=k; }
   public  Object cloneself(){
      boolean[] s=new boolean[Knap.n];
      System.arraycopy(set, 0, s, 0, Knap.n);
      return new  KnapLeastMonoBin(s, cw, cval);
   }.........
```

```
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
Value:   2436     Weight:   1194
30:   Elapsed time: 4025
```

思考题 3.7：用动态规划来求解 0-1 背包问题。

```
public class KnapDynamic {
   int[][] a=new int[Knap.n+1][Knap.wlimit+1];
   int[] perm=new int[Knap.n];
   public void compval(){
      for (int i=Knap.n-1; i>=0; --i){
         int k=Math.min(Knap.weight[i]-1, Knap.wlimit);
         for (int j=0; j<=k; ++j) a[i][j]=a[i+1][j];
         for (int j=Knap.weight[i]; j<=Knap.wlimit; ++j)
            a[i][j]=Math.max(a[i+1][j],
                  a[i+1][j-Knap.weight[i]]+Knap.value[i]);
      }
   }
   public void compperm(){ int b=Knap.wlimit;
      for (int i=0; i<Knap.n; ++i)
         if (a[i][b]==a[i+1][b]) perm[i]=0;
         else{ perm[i]=1; b-=Knap.weight[i]; }
   }......... .
```

```
wall: 762413     vall: 754879     wlimit: 400000
1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1,
1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0,
1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0,
1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
399973,    629864
500:    Elapsed time: 3011
```

```
public class KnapDynamic1 {
   int[][] a=new int[Knap.n][Knap.wlimit+1];
   int[] perm=new int[Knap.n];
   public int compval1(int k, int lim){
      if (a[k][lim]>1) return a[k][lim];
      if (k==Knap.n-1){
         if (lim<Knap.weight[k]){
            a[k][lim]=1; return 1; }
         else { a[k][lim]=Knap.value[k]+1;
            return Knap.value[k]+1; }
      } else if (lim<Knap.weight[k]){
          a[k][lim]=compval1(k+1, lim);
          return a[k][lim];
      } else { a[k][lim]=Math.max(compval1(k+1, lim),
                compval1(k+1, lim-Knap.weight[k])+Knap.value[k]);
          return a[k][lim];   }
   }
   public void compperm(){ int b=Knap.wlimit;
       for (int i=0; i<Knap.n-1; ++i)
          if (a[i][b]==a[i+1][b]) perm[i]=0;
          else{ perm[i]=1; b-=Knap.weight[i]; }
       int k=Knap.n-1;
       perm[k]= a[k][b]==1 ? 0 : 1;
   }……….
wall: 762413  vall: 754879  wlimit: 400000
1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0,
1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0,
1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1,
1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1,
1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
399973,  629864
500:  Elapsed time: 6755
```

思考题 3.8：设计并实现单调赋值排列树的优化搜索算法，用它来求解旅行商问题。

```
public interface WPerNode extends Check, Valued {
   WPerNode down(int i); }
public class PerOptSearch {
   private WPerNode[] probs, ans;
   private int[] perm; private WPerNode lnode;
   private int dep, level;
   public PerOptSearch(WPerNode p, int k){
      dep=k; ans=new WPerNode[dep+1];
      probs=new WPerNode[dep+1];
      probs[0]=p; perm=new int[dep];
      for (int i=0; i<dep; ++i) perm[i]=i; }
   public void depthfirst(){
      WPerNode p=probs[level], sub=null;
      if (lnode==null || p.val()<lnode.val())
         if (p.target()){
            System.arraycopy(probs, 0, ans, 0, dep+1); lnode=p; }
         else for (int i=level; i<dep; ++i)
            if ((sub=p.down(perm[i]))!=null){
               int t=perm[level]; perm[level]=perm[i]; perm[i]=t;
```

```
                probs[++level]=sub; depthfirst(); --level;
                t=perm[level]; perm[level]=perm[i]; perm[i]=t; } }
   public WPerNode[] outans(){ return ans; }
}
public class Tsp1 implements WPerNode {
   static int m; static int[][] g; static PrintWriter out;
   private int node, level, curv;
   public Tsp1(int n, int lev, int c) { node=n; level=lev; curv=c; }
   public int depth(){ return m-1; }
   public boolean target(){ return level==m-1; }
   public WPerNode down(int i){ int k=curv+g[node][i+1];
      if (level==m-2) k+=g[i+1][0]; return new Tsp1(i+1, level+1, k); }
   public void output(){ out.print(node+", ");  }
   public int val(){ return curv; }
   public void setval(int k){ curv=k; }
   public static void main(String[] args) {
      PrintWriter fout=null;
       try{ fout=new PrintWriter(new FileWriter("out1-20.txt")); }
       catch(IOException e){ System.out.println("Error!"); }
       long begin=System.currentTimeMillis(); int mm=20;
       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;
       fout.println(mm+": ");
      for (int i=0; i<mm; ++i){
         for (int j=0; j<mm; ++j)
            fout.print(gg[i][j]+",  "); fout.println();
      }
       Tsp1.m=mm;
       Tsp1.out=fout; Tsp1.g=gg;
       PerOptSearch search=new PerOptSearch(new Tsp1(0, 0, 0), mm);
       search.depthfirst(); WPerNode[] a=search.outans();
       for (int i=0; i<a.length; ++i) fout.print(((Tsp1)a[i]).node+",
");
       fout.println(); fout.println(((Tsp1)a[a.length-1]).curv);
       long end=System.currentTimeMillis();
       fout.println(mm+", jt: 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
12, jt:    Elapsed time: 46

0, 3, 8, 16, 6, 11, 10, 7, 17, 19, 1, 14, 9, 12, 2, 13, 5, 15, 18, 4,
208
20, jt:    Elapsed time: 267025
```

思考题 3.9：设计并实现单调赋值回溯排列树的深度优先搜索算法，用它来求解旅行商问题。

```
public interface WPerMono extends PerMono, Valued { Object cloneself(); }
public class PerBackOptSearch {
   private WPerMono prob, ans;
   private int dep, level; private int[] perm;
   public PerBackOptSearch(WPerMono p, int k){
      prob=p; dep=k; perm=new int[dep];
      for (int i=0; i<dep; ++i) perm[i]=i; }
   public void depthfirst(){
      if (ans==null || prob.val()<ans.val())
         if (prob.target()) ans=(WPerMono)prob.cloneself();
         else for (int i=level; 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 WPerMono outans(){ return ans; }
}
public class Tsp2 implements WPerMono {
   static int m; static int[][] g; static PrintWriter out;
   private int level=1, curv; private int[] perm;
   public Tsp2(){ perm=new int[m]; }
   public Tsp2(int[] p, int v){ perm=new int[m];
      System.arraycopy(p, 0, perm, 0, m); curv=v; }
   public int depth(){ return m-1; }
   public boolean target(){ return level==m; }
   public boolean down(int i){
      curv+=g[perm[level-1]][i+1]; perm[level++]=i+1;
      if (level==m) curv+=g[i+1][0]; return true;  }
   public void up(){ int i=perm[--level];
      curv-=g[perm[level-1]][i];
      if (level==m-1) curv-=g[i][0]; }
   public void output(){
      for (int i=0; i<m; ++i)
         out.print(perm[i]+", ");
      out.println();  out.println(curv); }
   public Object cloneself(){ return new Tsp2(perm, curv); }
   public int val(){ return curv; }
   public void setval(int k){ curv=k; }.........
```

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

208

20, jt: Elapsed time: 254549

思考题 **3.10**：用动态规划来求解旅行商问题。

```
public class DTsp {
   public static class Pair { int p, v;
      public Pair(int pp, int vv){ p=pp; v=vv; }
   }
   public static Pair[][] map; public static int n;
   public static boolean[] bin;
   public static int power(int k){ int ans=1;
      for (int i=0; i<k; ++i) ans*=2; return ans; }
   public static int bool2n(){ int ans=0;
      for (int i=n-2; i>=0; --i)
         if (bin[i]) ans=2*ans+1; else ans*=2; return ans;
   }
   public static void com2bin(LexIncSeq c, int k){
      for (int i=0; i<n-1; ++i) bin[i]=false;
      for (int i=0; i<k; ++i) bin[c.content(i)]=true;
   }
   public static int com2n(LexIncSeq c, int k){
      com2bin(c, k); return bool2n();
   }
   public static void dp(int[][] g){
      map=new Pair[power(n-1)][n-1];  bin=new boolean[n-1];
      for (int i=0; i<n-1; ++i)
         map[power(i)][i]=new Pair(n-1, g[n-1][i]);
      for (int k=2; k<n; ++k){
         LexIncSeq com=new LexIncSeq(n-1, k);
         do { int set1=com2n(com, k);
            for (int i=0; i<k; ++i){
               int node=com.content(i); bin[node]=false;
```

```
                int lst=Integer.MAX_VALUE,
                   pt=n-1, set2=bool2n();
                for (int j=0; j<k; ++j)
                   if (i!=j){
                      int cn=com.content(j),
                      cur=map[set2][cn].v+g[cn][node];
                      if (cur<lst){ lst=cur; pt=cn; }
                   }
                map[set1][node]=new Pair(pt, lst);
                bin[node]=true;
             }
          } while (com.next());
       }
       int set=power(n-1)-1, least=Integer.MAX_VALUE, last=n-1;
       for (int i=0; i<n-1; ++i){
          int cur=map[set][i].v+g[i][n-1];
          if (cur<least){ least=cur; last=i; }
       }
       int[] ans=new int[n];
       ans[n-1]=n-1; ans[n-2]=last;
       for (int i=0; i<n-1; ++i) bin[i]=true;
       for (int i=n-3; i>=0; --i){
          ans[i]=map[bool2n()][last].p;
          bin[last]=false; last=ans[i];
       }
       for (int i=0; i<n; ++i) System.out.print(ans[i]+",  ");
       System.out.println();
       System.out.println(least);
    }
```

```
208
20:   Elapsed time: 3323

367
22:   Elapsed time: 16614

24: OutOfMemoryError
```

思考题 3.11：为旅行商问题设计限界函数，用来求解旅行商问题，通过比较检查限界函数的效果。

```
public class Tsp3 implements WPerMono { static int m;
   static int[][] g; static PrintWriter out;
   static final int maxval=100000000;
   int level=1, curv, h, n1, n2;
   int[] perm=new int[m], remain=new int[m-1];
   public Tsp3(){ for (int i=0; i<m-1; ++i) remain[i]=i+1; }
   public Tsp3(int[] p, int v, int hh, int nn1, int nn2){
      System.arraycopy(p, 0, perm, 0, m);
      curv=v; h=hh; n1=nn1; n2=nn2; }
   int heu(int[] a, int len, int head, int tail){
      int ans1=0, ans2=0, least=maxval;
      if (len==2){
         int k1=g[tail][remain[0]]+g[remain[0]][remain[1]]+
               g[remain[1]][head],
         k2=g[tail][remain[1]]+g[remain[1]][remain[0]]+
               g[remain[0]][head];
         if (k1<=k2){ n1=remain[0]; n2=remain[1]; return k1; }
         else { n1=remain[1]; n2=remain[0]; return k2; }
      }
      for (int i=0; i<len; ++i)
         least=Math.min(least, g[tail][remain[i]]);
      ans1=least;
      for (int i=0; i<len; ++i){ least=g[remain[i]][head];
         for (int j=0; j<len; ++j)
            if (j!=i)
               least=Math.min(least, g[remain[i]][remain[j]]);
         ans1+=least; }
      least=maxval;
      for (int i=0; i<len; ++i)
         least=Math.min(least, g[remain[i]][head]); ans2=least;
      for (int i=0; i<len; ++i){ least=g[tail][remain[i]];
         for (int j=0; j<len; ++j)
            if (j!=i)
               least=Math.min(least, g[remain[j]][remain[i]]);
         ans2+=least; }
      return Math.max(ans1, ans2); }
   public boolean target(){ return level==m-2; }
   public boolean down(int i){ curv+=g[perm[level-1]][i+1];
      perm[level++]=i+1;
      int k=0; while (remain[k]!=i+1) ++k;
      for (int j=k; j<m-level; ++j) remain[j]=remain[j+1];
      if (level>2) h=heu(remain, m-level, 0, i+1); return true;  }
   public void up(){ int i=perm[--level];
      curv-=g[perm[level-1]][i]; h=0; remain[m-level-1]=i; }
   public void output(){
      for (int i=0; i<m-2; ++i) out.print(perm[i]+", ");
      out.println(n1+", "+n2); out.println(curv+h); }
   public Object cloneself(){
      return new Tsp3(perm, curv, h, n1, n2); }
   public int val(){ return curv+h; }
   public void setval(int k){ curv=k; }
```

```
    public static void main(String[] args) {
        PrintWriter fout=null;
        try{ fout=new PrintWriter(new FileWriter("out3-26.txt")); }
        catch(IOException e){ System.out.println("Error!"); }
        long begin=System.currentTimeMillis(); int mm=26;
        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(200)+1;
        fout.println(mm+": ");
        for (int i=0; i<mm; ++i){
            for (int j=0; j<mm; ++j)
                fout.print(gg[i][j]+",  "); fout.println();
        }
        Tsp3.m=mm;
        Tsp3.out=fout; Tsp3.g=gg;
        PerBackOptSearch search=new PerBackOptSearch(new Tsp3(), mm);
        search.depthfirst(); WPerMono a=search.outans();
        ((Tsp3)a).output();
        long end=System.currentTimeMillis();
        fout.println(mm+", jt:  Elapsed time:  "+(end-begin));
        fout.close();
    }
}
```

```
0, 3, 8, 16, 6, 11, 10, 7, 17, 19, 1, 14, 9, 12, 2, 13, 5, 15, 18, 4
208
20, jt:    Elapsed time: 2060

0, 23, 2, 13, 21, 24, 18, 3, 1, 20, 17, 25, 5, 4, 11, 7, 15, 14, 9, 22, 19, 10, 6, 8, 16, 12
383
26, jt:    Elapsed time: 45817
```

```
public class Tsp4 implements WPerMono {
    static int m; static int[][] g; static PrintWriter out;
    static final int maxval=100000000;
    private int level=1, curv, h, n1, n2;
    private int[] perm=new int[m],
        remain=new int[m-1], leasta=new int[m];
    public Tsp4(){ for (int i=0; i<m-1; ++i) remain[i]=i+1;  }
    public Tsp4(int[] p, int v, int hh, int nn1, int nn2){
        System.arraycopy(p, 0, perm, 0, m);
        curv=v; h=hh; n1=nn1; n2=nn2; }
    int heu(int[] a, int len, int head, int tail){
        int leastt=0, ans=0, least=maxval;
        if (len==2){
            int k1=g[tail][remain[0]]+g[remain[0]][remain[1]]+
               g[remain[1]][head],
             k2=g[tail][remain[1]]+g[remain[1]][remain[0]]+
               g[remain[0]][head];
          if (k1<=k2){ n1=remain[0]; n2=remain[1]; return k1; }
          else { n1=remain[1]; n2=remain[0]; return k2; }
       }
       for (int i=0; i<len; ++i)
          least=Math.min(least, g[tail][remain[i]]);
       ans=leastt=least;
       for (int i=0; i<len; ++i){ least=g[remain[i]][head];
          for (int j=0; j<len; ++j)
             if (j!=i)
                least=Math.min(least, g[remain[i]][remain[j]]);
             ans+=least; leasta[i]=least; }
       least=maxval;
       for (int i=0; i<len; ++i)
          least=Math.min(least, g[remain[i]][head]-leasta[i]);
       ans+=least;
       for (int i=0; i<len; ++i){ least=g[tail][remain[i]]-leastt;
          for (int j=0; j<len; ++j)
             if (j!=i)
                least=Math.min(least,
                     g[remain[j]][remain[i]]-leasta[j]);
          ans+=least; }
       return ans;
    }
    public boolean target(){ return level==m-2; }
    public boolean down(int i){
       curv+=g[perm[level-1]][i+1]; perm[level++]=i+1;
       int k=0; while (remain[k]!=i+1) ++k;
       for (int j=k; j<m-level; ++j) remain[j]=remain[j+1];
       if (level>2) h=heu(remain, m-level, 0, i+1);
       return true;
    }
    public void up(){ int i=perm[--level];
    curv-=g[perm[level-1]][i]; h=0; remain[m-level-1]=i; }
    public void output(){
       for (int i=0; i<m-2; ++i) out.print(perm[i]+", ");
       out.println(n1+", "+n2); out.println(curv+h); }
    public Object cloneself(){
       return new Tsp4(perm, curv, h, n1, n2); }
    public int val(){ return curv+h; }.........
```

```
0, 23, 2, 13, 21, 24, 18, 3, 1, 20, 17, 25, 5, 4, 11, 7, 15, 14, 9, 22, 19, 10, 6, 8, 16, 12
383
```

```
26, jt:    Elapsed time: 3401

0, 22, 12, 27, 20, 10, 1, 5, 3, 9, 21, 2, 13, 11, 14, 8, 18, 19, 24, 28, 25, 29, 7, 23, 4, 26, 15, 17, 6, 16
421
30, jt:    Elapsed time: 52057
```

算法欣赏 4：旅行商问题的 LMSK 算法。

版本1 (状态空间)：

```
public class Tsp5 implements WNode {
   static final int maxval=100000000;
   static int m; static PrintWriter out;
   static int[] rp, cp; private int[][] data;
   private int n, curv, lr, lc, lrv, lcv;
   private int[] next, prev, row, column;
   public Tsp5(int[][] g){
      m=n=g.length; data=new int[m][m];
      rp=new int[m]; cp=new int[m];
      next=new int[m]; prev=new int[m];
      for (int i=0; i<m; ++i)
         System.arraycopy(g[i], 0, data[i], 0, m);
      for (int i=0; i<m; ++i) data[i][i]=maxval;
      row=new int[m]; column=new int[m];
      for (int i=0; i<m; ++i)
         next[i]=prev[i]=row[i]=column[i]=i;
      proceed(); find(); }
   public Tsp5(int[][] g, int nn, int[] nex,
         int[] pre, int[] r, int[] c, int cv){
      data=g; n=nn; next=nex; prev=pre; row=r;
      column=c; curv=cv; }
   private void proceed(){
      for (int i=0; i<n; ++i){ int min=maxval;
         for (int j=0; j<n; ++j)
            min=Math.min(data[i][j], min);
         curv+=min;
         for (int j=0; j<n; ++j) data[i][j]-=min; }
      for (int i=0; i<n; ++i){ int min=maxval;
         for (int j=0; j<n; ++j)
            min=Math.min(data[j][i], min);
         curv+=min;
         for (int j=0; j<n; ++j) data[j][i]-=min; }
   }
   private void find(){ int sum=0;
      for (int i=0; i<n; ++i) for (int j=0; j<n; ++j)
         if (data[i][j]==0){
            int a=maxval, b=maxval; data[i][j]=maxval;
            for (int k=0; k<n; ++k){
               a=Math.min(data[i][k], a);
               b=Math.min(data[k][j], b); }
            if (a+b>=sum){
               lr=i; lc=j; lrv=a; lcv=b; sum=a+b; }
            data[i][j]=0; }
   }
   public boolean target(){ return n==2; }
   public int subnum(){ return n>2 ? 2 : 0; }
   public WNode down(int i){
      Tsp5 ans=null; int[] n1=new int[m], p1=new int[m];
      if (i==0){ int[][] d=new int[n-1][n-1];
         for (int j=0; j<n-1; ++j) for (int k=0; k<n-1; ++k){
            d[j][k]=data[j<lr?j:j+1][k<lc?k:k+1]; }
         System.arraycopy(next, 0, n1, 0, m);
         System.arraycopy(prev, 0, p1, 0, m);
         n1[row[lr]]=column[lc]; p1[column[lc]]=row[lr];
         int[] r1=new int[n-1], c1=new int[n-1];
         for (int j=0; j<n-1; ++j){
            if (j<lr){ r1[j]=row[j]; rp[row[j]]=j; }
            else { r1[j]=row[j+1]; rp[row[j+1]]=j; }
            if (j<lc){ c1[j]=column[j]; cp[column[j]]=j; }
            else { c1[j]=column[j+1]; cp[column[j+1]]=j; } }
         int j, k;
         for (j=row[lr]; prev[j]!=j;
         j=prev[j]);
         for (k=column[lc]; next[k]!=k; k=next[k]);
         d[rp[k]][cp[j]]=maxval;
         if (n>3){ ans=new Tsp5(d, n-1, n1, p1, r1, c1, curv);
            ans.proceed(); ans.find(); }
         else if (d[0][0]+d[1][1]<d[0][1]+d[1][0]){
            n1[r1[0]]=c1[0]; p1[c1[0]]=r1[0];
            n1[r1[1]]=c1[1]; p1[c1[1]]=r1[1];
            ans=new Tsp5(d, n-1, n1, p1, r1, c1,
                  curv+d[0][0]+d[1][1]); }
         else { n1[r1[1]]=c1[0]; p1[c1[0]]=r1[1];
         n1[r1[0]]=c1[1]; p1[c1[1]]=r1[0];
            ans=new Tsp5(d, n-1, n1, p1, r1, c1,
                  curv+d[0][1]+d[1][0]); } }
      else { int[][] d=new int[n][n];
         for (int j=0; j<n; ++j)
            System.arraycopy(data[j], 0, d[j], 0, n);
         for (int j=0; j<n; ++j){
            d[lr][j]-=lrv; d[j][lc]-=lcv; }
         d[lr][lc]=maxval;
         int[] r1=new int[n], c1=new int[n];
         System.arraycopy(next, 0, n1, 0, m);
         System.arraycopy(prev, 0, p1, 0, m);
         System.arraycopy(column, 0, c1, 0, n);
         System.arraycopy(row, 0, r1, 0, n);
         ans=new Tsp5(d, n, n1, p1, r1, c1, curv+lrv+lcv);
         ans.find(); }
      return ans;
   }
   public void output(){
      for (int i=0, j=0; i<m; ++i, j=next[j])
         out.print(j+", ");
      out.println(); out.println(curv); }
   public int val(){ return curv; }
   public void setval(int k){ curv=k; }
   public static void main(String[] args) {
      PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("out5-6.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(); int mm=6;
      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(50)+1;
      fout.println(mm+": ");
      for (int i=0; i<mm; ++i){
         for (int j=0; j<mm; ++j) fout.print(gg[i][j]+",  ");
         fout.println();
      }
      Tsp5.m=mm;  Tsp5.out=fout;
      TreeOptSearch search=new TreeOptSearch(new Tsp5(gg));
      //search.depthfirst(); WLink a=search.outans();
      //search.optfirst(); WLink a=search.outans();
      search.optfirst1(); WLink a=search.outans();
      ((Tsp5)a.prob).output();
      long end=System.currentTimeMillis();
      fout.println(mm+", jt:  Elapsed time: "+(end-begin));
      fout.close();
   }
}
```
深度优先：

```
0, 22, 12, 27, 20, 10, 1, 5, 3, 9, 21, 2, 13, 11, 14, 8, 18, 19, 24, 28, 25, 29, 7, 23, 4, 26, 15, 17, 6, 16,
421
30, jt:    Elapsed time: 47
```

```
0, 41, 29, 16, 42, 24, 38, 26, 31, 15, 40, 11, 36, 22, 33, 9, 39, 7, 13, 28, 35, 8, 21, 3, 27, 34, 19, 17,
32, 2, 37, 23, 30, 10, 14, 6, 12, 5, 20, 1, 43, 18, 25, 4,
682
44, jt:    Elapsed time: 78

0, 27, 20, 10, 35, 15, 43, 8, 16, 47, 29, 50, 30, 38, 51, 24, 41, 11, 32, 25, 28, 42, 53, 7, 52, 3, 33, 14,
1, 13, 5, 17, 4, 39, 2, 36, 46, 22, 26, 44, 49, 23, 19, 31, 21, 48, 9, 37, 12, 6, 40, 34, 18, 45,
838
54, jt:    Elapsed time: 796

0, 15, 44, 61, 10, 48, 2, 51, 42, 27, 19, 35, 5, 58, 9, 53, 57, 13, 11, 63, 50, 41, 47, 54, 43, 59, 1, 18,
12, 56, 28, 20, 25, 39, 62, 33, 26, 38, 14, 32, 40, 60, 29, 36, 45, 7, 30, 23, 37, 4, 34, 22, 21, 6, 55, 52,
3, 46, 24, 8, 49, 17, 16, 31,
1153
64, jt:    Elapsed time: 118107

0, 48, 28, 50, 60, 47, 43, 1, 9, 3, 17, 49, 51, 5, 27, 44, 57, 41, 65, 20, 26, 6, 46, 29, 55, 7, 13, 11, 2, 18, 24, 34, 12,
37, 30, 54, 25, 52, 21, 31, 40, 64, 42, 35, 16, 23, 10, 62, 14, 8, 22, 32, 33, 45, 39, 36, 38, 61, 4, 19, 56, 15, 58, 63,
53, 59,
956
66, jt:    Elapsed time: 27425

0, 17, 42, 30, 6, 49, 52, 4, 21, 31, 11, 44, 34, 50, 19, 38, 65, 5, 56, 37, 48, 1, 8, 24, 35, 27, 18, 9, 3,
41, 47, 15, 2, 58, 12, 28, 51, 61, 40, 54, 60, 36, 29, 20, 64, 63, 57, 45, 62, 13, 32, 16, 59, 53, 14, 46,
33, 43, 25, 23, 22, 26, 7, 55, 66, 39, 67, 10,
1337
68, jt:    Elapsed time: 13588

0, 42, 12, 7, 10, 28, 45, 6, 29, 48, 32, 68, 53, 3, 25, 56, 54, 5, 18, 4, 33, 38, 60, 22, 51, 23, 8, 36, 21,
64, 1, 52, 20, 61, 35, 59, 40, 57, 34, 17, 14, 15, 24, 2, 50, 13, 16, 46, 69, 11, 41, 58, 65, 63, 55, 37,
67, 43, 9, 62, 66, 26, 44, 47, 19, 39, 49, 30, 31, 27,
1074
70, jt:    Elapsed time: 171
```

最优优先：

```
0, 41, 29, 16, 42, 24, 38, 26, 31, 15, 40, 11, 36, 22, 33, 9, 39, 7, 13, 28, 35, 8, 21, 3, 27, 34, 19, 17,
32, 2, 37, 23, 30, 10, 14, 6, 12, 5, 20, 1, 43, 18, 25, 4,
682
```

44, jt: Elapsed time: 125

46：内存不足！

版本2(优化的状态空间)：

```
public class Tsp7 implements WNode {
   static final int maxval=100000000;
   static int m; static PrintWriter out;
   static int[] rp, cp; private int[][] data;
   boolean lchild;
   private int n, curv, lr, lc, lrv, lcv;
   private int[] next, prev, row, column;
   public Tsp7(int[][] g){
      m=n=g.length; data=new int[m][m];
      rp=new int[m]; cp=new int[m];
      next=new int[m]; prev=new int[m];
      for (int i=0; i<m; ++i)
         System.arraycopy(g[i], 0, data[i], 0, m);
      for (int i=0; i<m; ++i) data[i][i]=maxval;
      row=new int[m]; column=new int[m];
      for (int i=0; i<m; ++i)
         next[i]=prev[i]=row[i]=column[i]=i;
      proceed(); find(); }
   public Tsp7(int[][] g, int nn, int[] nex, int[] pre,
         int[] r, int[] c, int cv, boolean b){
      data=g; n=nn; next=nex; prev=pre; row=r;
      column=c; curv=cv; lchild=b; }
   private void proceed(){
      for (int i=0; i<n; ++i){ int min=maxval;
         for (int j=0; j<n; ++j) min=Math.min(data[i][j], min);
         curv+=min;
         for (int j=0; j<n; ++j) data[i][j]-=min; }
      for (int i=0; i<n; ++i){ int min=maxval;
         for (int j=0; j<n; ++j) min=Math.min(data[j][i], min);
         curv+=min;
         for (int j=0; j<n; ++j) data[j][i]-=min; }
   }
   private void find(){ int sum=0;
      for (int i=0; i<n; ++i) for (int j=0; j<n; ++j)
         if (data[i][j]==0){
            int a=maxval, b=maxval; data[i][j]=maxval;
            for (int k=0; k<n; ++k){
               a=Math.min(data[i][k], a);
               b=Math.min(data[k][j], b); }
            if (a+b>=sum){ lr=i; lc=j; lrv=a; lcv=b; sum=a+b; }
            data[i][j]=0; }
   }
   public boolean target(){ return n==2; }
   public int subnum(){ return n>2 ? 2 : 0; }
   public WNode down(int i){ Tsp7 ans=null;
      if (i==0){ int[][] d=new int[n-1][n-1];
      int[] n1=new int[n-1], p1=new int[n-1];
         for (int j=0; j<n-1; ++j) for (int k=0; k<n-1; ++k){
            d[j][k]=data[j<lr?j:j+1][k<lc?k:k+1]; }
         int[] r1=new int[n-1], c1=new int[n-1];
         for (int j=0; j<n-1; ++j){
            if (j<lr){
               r1[j]=row[j]; p1[j]=prev[j]; rp[row[j]]=j; }
            else { r1[j]=row[j+1];
               p1[j]=prev[j+1]; rp[row[j+1]]=j; }
            if (j<lc){ c1[j]=column[j];
               n1[j]=next[j]; cp[column[j]]=j; }
            else { c1[j]=column[j+1]; n1[j]=next[j+1];
               cp[column[j+1]]=j; } }
         p1[rp[next[lc]]]=prev[lr]; n1[cp[prev[lr]]]=next[lc];
         d[rp[next[lc]]][cp[prev[lr]]]=maxval;
         if (n>3){ ans=new Tsp7(d, n-1, n1, p1, r1, c1, curv, true);
         ans.proceed(); ans.find(); }
         else if (d[0][0]+d[1][1]<d[0][1]+d[1][0]){
            n1[0]=0; n1[1]=0; p1[0]=1; p1[1]=1;
            ans=new Tsp7(d, n-1, n1, p1, r1, c1,
                  curv+d[0][0]+d[1][1], true); }
         else { n1[0]=0; n1[1]=1; p1[0]=1; p1[1]=0;
            ans=new Tsp7(d, n-1, n1, p1, r1, c1,
                  curv+d[0][1]+d[1][0], true); } }
      else { int[][] d=new int[n][n];
         int[] n1=new int[n], p1=new int[n];
         for (int j=0; j<n; ++j)
            System.arraycopy(data[j], 0, d[j], 0, n);
         for (int j=0; j<n; ++j){
            d[lr][j]-=lrv; d[j][lc]-=lcv; }
         d[lr][lc]=maxval;
         int[] r1=new int[n], c1=new int[n];
         System.arraycopy(next, 0, n1, 0, n);
         System.arraycopy(prev, 0, p1, 0, n);
         System.arraycopy(column, 0, c1, 0, n);
         System.arraycopy(row, 0, r1, 0, n);
         ans=new Tsp7(d, n, n1, p1, r1, c1, curv+lrv+lcv, false);
         ans.find(); }
      return ans;
   }
   public void output(){}
   public int val(){ return curv; }
   public void setval(int k){ curv=k; }
   public static void main(String[] args) {
      PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("out7-64.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(); int mm=64;
      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(600)+1;
      fout.println(mm+": ");
      for (int i=0; i<mm; ++i){
         for (int j=0; j<mm; ++j) fout.print(gg[i][j]+",  ");
         fout.println();
      }
      Tsp7.m=mm;  Tsp7.out=fout;
      TreeOptSearch search=new TreeOptSearch(new Tsp7(gg));
      search.depthfirst(); WLink a=search.outans();
      //search.optfirst(); WLink a=search.outans();
      //search.optfirst1(); WLink a=search.outans();
      int[] ans=new int[m]; int val=a.prob.val();
      Tsp7 leaf=(Tsp7)a.prob;
      ans[leaf.row[leaf.next[0]]]=leaf.column[leaf.next[1]];
      ans[leaf.row[leaf.prev[0]]]=leaf.column[leaf.prev[1]];
      a=a.next; boolean isleft=true;
      while (a!=null){ Tsp7 cur=(Tsp7)a.prob;
         if (isleft) ans[cur.row[cur.lr]]=cur.column[cur.lc];
         isleft=cur.lchild; a=a.next;
      }
      for (int i=0, j=0; i<m; ++i, j=ans[j]) out.print(j+", ");
      fout.println(); fout.println(val);
      long end=System.currentTimeMillis();
      fout.println(mm+", jt:  Elapsed time: "+(end-begin));
      fout.close();
   }
}
```
深度优先：

```
0, 41, 29, 16, 42, 24, 38, 26, 31, 15, 40, 11, 36, 22, 33, 9, 39, 7, 13, 28, 35, 8, 21, 3, 27, 34, 19, 17, 32, 2, 37, 23,
30, 10, 14, 6, 12, 5, 20, 1, 43, 18, 25, 4,
682
44, jt:    Elapsed time: 78
```

```
0, 27, 20, 10, 35, 15, 43, 8, 16, 47, 29, 50, 30, 38, 51, 24, 41, 11, 32, 25, 28, 42, 53, 7, 52, 3, 33, 14,
1, 13, 5, 17, 4, 39, 2, 36, 46, 22, 26, 44, 49, 23, 19, 31, 21, 48, 9, 37, 12, 6, 40, 34, 18, 45,
838
54, jt:    Elapsed time: 765
```

```
0, 15, 44, 61, 10, 48, 2, 51, 42, 27, 19, 35, 5, 58, 9, 53, 57, 13, 11, 63, 50, 41, 47, 54, 43, 59, 1, 18, 12, 56, 28, 20,
25, 39, 62, 33, 26, 38, 14, 32, 40, 60, 29, 36, 45, 7, 30, 23, 37, 4, 34, 22, 21, 6, 55, 52, 3, 46, 24, 8, 49, 17, 16, 31,
1153
64, jt:    Elapsed time: 117484
```

```
0, 48, 28, 50, 60, 47, 43, 1, 9, 3, 17, 49, 51, 5, 27, 44, 57, 41, 65, 20, 26, 6, 46, 29, 55, 7, 13, 11, 2, 18, 24, 34, 12,
37, 30, 54, 25, 52, 21, 31, 40, 64, 42, 35, 16, 23, 10, 62, 14, 8, 22, 32, 33, 45, 39, 36, 38, 61, 4, 19, 56, 15, 58, 63,
53, 59,
956
66, jt:    Elapsed time: 27082
```

```
0, 17, 42, 30, 6, 49, 52, 4, 21, 31, 11, 44, 34, 50, 19, 38, 65, 5, 56, 37, 48, 1, 8, 24, 35, 27, 18, 9, 3, 41, 47, 15, 2,
58, 12, 28, 51, 61, 40, 54, 60, 36, 29, 20, 64, 63, 57, 45, 62, 13, 32, 16, 59, 53, 14, 46, 33, 43, 25, 23, 22, 26, 7, 55,
66, 39, 67, 10,
1337
68, jt:    Elapsed time: 13525
```

```
0, 42, 12, 7, 10, 28, 45, 6, 29, 48, 32, 68, 53, 3, 25, 56, 54, 5, 18, 4, 33, 38, 60, 22, 51, 23, 8, 36, 21, 64, 1, 52, 20,
61, 35, 59, 40, 57, 34, 17, 14, 15, 24, 2, 50, 13, 16, 46, 69, 11, 41, 58, 65, 63, 55, 37, 67, 43, 9, 62, 66, 26, 44, 47,
19, 39, 49, 30, 31, 27,
1074
70, jt:    Elapsed time: 187
```

最优优先：

```
0, 41, 29, 16, 42, 24, 38, 26, 31, 15, 40, 11, 36, 22, 33, 9, 39, 7, 13, 28, 35, 8, 21, 3, 27, 34, 19, 17,
32, 2, 37, 23, 30, 10, 14, 6, 12, 5, 20, 1, 43, 18, 25, 4,
682
44, jt:    Elapsed time: 125
```

```
0, 27, 20, 10, 35, 15, 43, 8, 16, 47, 29, 50, 30, 38, 51, 24, 41, 11, 32, 25, 28, 42, 53, 7, 52, 3, 33, 14,
1, 13, 5, 17, 4, 39, 2, 36, 46, 22, 26, 44, 49, 23, 19, 31, 21, 48, 9, 37, 12, 6, 40, 34, 18, 45,
838
54, jt:    Elapsed time: 1373
```

56：内存不足！

版本3(回溯空间)：

```
public class Tsp6 implements WMono {
   static final int maxval=100000000, maxlev=500;
   static PrintWriter out;
   private int[][][] datas;
   private int[][] nexts, prevs, rows, cols;
   private int[] his, next, curvs, lrs, lcs, lrvs, lcvs, rp, cp;
   private int m, lev, n, curv;
   public Tsp6(int[][] g){
      m=n=g.length; datas=new int[m+1][][];
      for (int i=1; i<=m; ++i) datas[i]=new int[i][i];
      nexts=new int[m+1][m]; prevs=new int[m+1][m];
      rows=new int[m+1][]; cols=new int[m+1][];
      for (int i=1; i<=m; ++i){
         rows[i]=new int[i]; cols[i]=new int[i]; }
      his=new int[maxlev]; curvs=new int[maxlev];
      lrs=new int[maxlev]; lcs=new int[maxlev];
      lrvs=new int[maxlev]; lcvs=new int[maxlev];
      rp=new int[m]; cp=new int[m];
      for (int i=0; i<m; ++i)
         System.arraycopy(g[i], 0, datas[m][i], 0, m);
      for (int i=0; i<m; ++i) datas[m][i][i]=maxval;
      for (int i=0; i<m; ++i)
         nexts[m][i]=prevs[m][i]=rows[m][i]=cols[m][i]=i;
      proceed(datas[m]); find(datas[m]);
   }
   public Tsp6(int[] ne, int cv){
      next=ne; curv=cv; m=ne.length; }
   private void proceed(int[][] d){
      for (int i=0; i<n; ++i){ int min=maxval;
         for (int j=0; j<n; ++j) min=Math.min(d[i][j], min);
         curv+=min;
         for (int j=0; j<n; ++j) d[i][j]-=min; }
      for (int i=0; i<n; ++i){ int min=maxval;
         for (int j=0; j<n; ++j) min=Math.min(d[j][i], min);
         curv+=min;
         for (int j=0; j<n; ++j) d[j][i]-=min; }
      curvs[lev]=curv;
   }
   private void find(int[][] d){ int sum=0;
      for (int i=0; i<n; ++i) for (int j=0; j<n; ++j)
         if (d[i][j]==0){
            int a=maxval, b=maxval; d[i][j]=maxval;
            for (int k=0; k<n; ++k){
               a=Math.min(d[i][k], a);
               b=Math.min(d[k][j], b); }
            if (a+b>=sum){ lrs[lev]=i; lcs[lev]=j;
               lrvs[lev]=a; lcvs[lev]=b; sum=a+b; }
            d[i][j]=0; }
   }
   public boolean target(){ return n==2; }
   public int subnum(){ return 2; }
   public boolean down(int i){ his[lev++]=i;
      if (i==0){ --n;
         for (int j=0; j<n; ++j) for (int k=0; k<n; ++k)
            datas[n][j][k]=
               datas[n+1][j<lrs[lev-1]?j:j+1][k<lcs[lev-1]?k:k+1];
         System.arraycopy(nexts[n+1], 0, nexts[n], 0, m);
         System.arraycopy(prevs[n+1], 0, prevs[n], 0, m);
         nexts[n][rows[n+1][lrs[lev-1]]]=cols[n+1][lcs[lev-1]];
         prevs[n][cols[n+1][lcs[lev-1]]]=rows[n+1][lrs[lev-1]];
         for (int j=0; j<n; ++j){
            if (j<lrs[lev-1]){
               rows[n][j]=rows[n+1][j]; rp[rows[n+1][j]]=j; }
            else { rows[n][j]=rows[n+1][j+1]; rp[rows[n+1][j+1]]=j; }
            if (j<lcs[lev-1]){
               cols[n][j]=cols[n+1][j]; cp[cols[n+1][j]]=j; }
            else { cols[n][j]=cols[n+1][j+1]; cp[cols[n+1][j+1]]=j; }
         }
         int j, k;
         for (j=rows[n+1][lrs[lev-1]];
         prevs[n+1][j]!=j; j=prevs[n+1][j]);
         for (k=cols[n+1][lcs[lev-1]];
         nexts[n+1][k]!=k; k=nexts[n+1][k]);
         datas[n][rp[k]][cp[j]]=maxval;
         if (n>2){ proceed(datas[n]); find(datas[n]); }
         else if (datas[n][0][0]+datas[n][1][1]<
               datas[n][0][1]+datas[n][1][0]){
            nexts[n][rows[n][0]]=cols[n][0];
            prevs[n][cols[n][0]]=rows[n][0];
            nexts[n][rows[n][1]]=cols[n][1];
            prevs[n][cols[n][1]]=rows[n][1];
            curv+=datas[n][0][0]+datas[n][1][1]; }
         else { nexts[n][rows[n][1]]=cols[n][0];
            prevs[n][cols[n][0]]=rows[n][1];
            nexts[n][rows[n][0]]=cols[n][1];
            prevs[n][cols[n][1]]=rows[n][0];
            curv+=datas[n][0][1]+datas[n][1][0]; }
      }
      else {
         for (int j=0; j<n; ++j){
            datas[n][lrs[lev-1]][j]-=lrvs[lev-1];
            datas[n][j][lcs[lev-1]]-=lcvs[lev-1]; }
         datas[n][lrs[lev-1]][lcs[lev-1]]=maxval;
         curv+=lrvs[lev-1]+lcvs[lev-1];
         curvs[lev]=curv; find(datas[n]);
      }
      return true;
   }
   public void up(){ int k=his[--lev];
      curv=curvs[lev]; if (k==0) ++n; }
   public void output(){
      for (int i=0, j=0; i<m; ++i, j=next[j])
         out.print(j+", ");
      out.println(); out.println(curv); }
   public Object cloneself(){ return new Tsp6(nexts[n], curv); }
   public int val(){ return curv; }
   public void setval(int k){ curv=k; }
   public static void main(String[] args) {
      PrintWriter fout=null;
      try{ fout=new PrintWriter(new FileWriter("out6-64.txt")); }
      catch(IOException e){ System.out.println("Error!"); }
      long begin=System.currentTimeMillis(); int mm=64;
      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(600)+1;
      fout.println(mm+": ");
      for (int i=0; i<mm; ++i){
         for (int j=0; j<mm; ++j) fout.print(gg[i][j]+",  ");
         fout.println();
      }
      Tsp6.out=fout;
      BackOptSearch search=new BackOptSearch(new Tsp6(gg));
      search.depthfirst(); WMono a=search.outans();
      ((Tsp6)a).output();
      long end=System.currentTimeMillis();
      fout.println(mm+", jt: Elapsed time: "+(end-begin));
      fout.close();
   }
}
```

```
0, 27, 20, 10, 35, 15, 43, 8, 16, 47, 29, 50, 30, 38, 51, 24, 41, 11, 32, 25, 28, 42, 53, 7, 52, 3, 33, 14, 1, 13, 5, 17, 4,
39, 2, 36, 46, 22, 26, 44, 49, 23, 19, 31, 21, 48, 9, 37, 12, 6, 40, 34, 18, 45,
838
54, jt:    Elapsed time: 640

0, 15, 44, 61, 10, 48, 2, 51, 42, 27, 19, 35, 5, 58, 9, 53, 57, 6, 55, 52, 3, 46, 24, 8, 13, 11, 63, 50, 41, 47, 54, 43, 59,
1, 18, 4, 34, 22, 21, 12, 56, 28, 20, 25, 39, 62, 33, 26, 38, 14, 32, 40, 60, 29, 36, 45, 7, 30, 23, 37, 49, 17, 16, 31,
1153
64, jt:    Elapsed time: 102664

0, 48, 28, 50, 60, 47, 43, 1, 9, 3, 17, 49, 51, 5, 27, 44, 57, 41, 65, 20, 26, 6, 46, 29, 55, 7, 13, 11, 2, 18, 24, 34, 12,
37, 30, 54, 25, 52, 21, 31, 40, 64, 42, 35, 16, 23, 10, 62, 14, 8, 22, 32, 33, 45, 39, 36, 38, 61, 4, 19, 56, 15, 58, 63,
53, 59,
956
66, jt:    Elapsed time: 23868

0, 17, 42, 30, 6, 49, 52, 4, 21, 31, 11, 44, 34, 50, 19, 38, 65, 5, 56, 37, 48, 1, 8, 24, 35, 27, 18, 9, 3, 41, 47, 15, 2,
58, 12, 28, 51, 61, 40, 54, 60, 36, 29, 20, 64, 63, 57, 45, 62, 13, 32, 16, 59, 53, 14, 46, 33, 43, 25, 23, 22, 26, 7, 55,
66, 39, 67, 10,
1337
68, jt:    Elapsed time: 11576

0, 42, 12, 7, 10, 28, 45, 6, 29, 48, 32, 68, 53, 3, 25, 56, 54, 5, 18, 4, 33, 38, 60, 22, 51, 23, 8, 36, 21, 64, 1, 52, 20,
61, 35, 59, 40, 57, 34, 17, 14, 15, 24, 2, 50, 13, 16, 46, 69, 11, 41, 58, 65, 63, 55, 37, 67, 43, 9, 62, 66, 26, 44, 47,
19, 39, 49, 30, 31, 27,
1074
70, jt:    Elapsed time: 156

0, 11, 34, 10, 41, 60, 8, 15, 44, 48, 35, 39, 19, 66, 12, 51, 5, 59, 20, 68, 69, 7, 18, 54, 46, 4, 65, 36, 70, 43, 37, 22,
58, 27, 26, 61, 23, 50, 67, 25, 45, 28, 31, 21, 17, 71, 62, 55, 64, 24, 52, 42, 30, 53, 32, 13, 63, 9, 33, 49, 56, 14, 29,
3, 40, 57, 16, 6, 2, 47, 1, 38,
1291
72, jt:    Elapsed time: 33743

0, 9, 31, 61, 19, 1, 72, 46, 60, 7, 59, 30, 26, 51, 54, 2, 12, 67, 66, 65, 73, 71, 39, 6, 48, 58, 45, 13, 32, 63, 28, 15,
52, 57, 27, 34, 35, 33, 37, 29, 47, 24, 64, 16, 17, 42, 20, 36, 3, 18, 41, 25, 4, 56, 62, 49, 22, 44, 38, 53, 70, 68, 55,
50, 10, 14, 23, 8, 40, 5, 21, 43, 11, 69,
1517
74, jt:    Elapsed time: 23774

0, 21, 65, 52, 53, 22, 55, 58, 70, 15, 63, 18, 31, 16, 6, 8, 68, 46, 61, 38, 28, 69, 56, 24, 40, 3, 20, 17, 45, 19, 48, 43,
11, 7, 13, 2, 49, 44, 73, 71, 30, 51, 5, 42, 75, 12, 57, 50, 10, 35, 47, 37, 62, 72, 26, 34, 1, 60, 66, 32, 74, 64, 36, 67,
9, 54, 4, 41, 33, 25, 59, 39, 29, 27, 23, 14,
1151
76, jt:    Elapsed time: 8049
```

思考题 3.12：为作业调度问题设计限界函数，用来求解作业调度问题，通过比较检查限界函数的效果。

```
public class Workshop2 implements WPerNode {
   static PrintWriter out; static int m; static int[][] data;
   int node, level, t1, t2;
   public Workshop2(int n, int lev, int tt1, int tt2){
      node=n; level=lev; t1=tt1; t2=tt2;
   }
   public boolean target(){ return level==m; }
   public WPerNode down(int i){
      int time1=t1+data[i][0],
         time2=Math.max(time1, t2)+data[i][1];
      return new Workshop2(i, level+1, time1, time2); }
   public void output(){ out.print(node+", ");  }
   public int val(){ return t2; }
   public void setval(int k){ t2=k; }…… .
```

```
17, 13;   7, 34;   25, 12;   1, 16;   47, 22;   19, 23;   12, 8;   46, 15;   37, 32;   6, 4;   41, 33;   17, 18;
-1, 0, 1, 3, 4, 5, 8, 10, 7, 11, 2, 6, 9,
281
12, jt:   Elapsed time: 41930
```

```
public class Workshop1 implements WPerNode {
   static PrintWriter out; static int m; static int[][] data;
   int node, level, t1, t2, curv;
   public Workshop1(int n, int lev, int tt1, int tt2, int v){
      node=n; level=lev; t1=tt1; t2=tt2; curv=v;
   }
   public boolean target(){ return level==m; }
   public WPerNode down(int i){
      int time1=t1+data[i][0],
         time2=Math.max(time1, t2)+data[i][1];
      return new Workshop1(i, level+1, time1, time2, curv+time2); }
   public void output(){ out.print(node+", ");  }
   public int val(){ return curv; }………
```

```
-1, 3, 9, 6, 1, 2, 0, 11, 5, 8, 10, 7, 4,
1366
12, jt:   Elapsed time: 20358
```

算法欣赏 5：车间调度问题的 Johnson 算法。

```
public class Workshop3 {
   public static class Pair implements Comparator {
      public int k, p;
      public Pair(int kk, int pp){ k=kk; p=pp; }
      public int compare(Object obj1, Object obj2){
         return ((Pair)obj1).k-((Pair)obj2).k; }
   }
   public static int[] perm;
   public static int wshop(){ int ans=0, k1=0, k2=0;
      perm=new int[Workshop1.m];
      Pair[] p1=new Pair[Workshop1.m],
         p2=new Pair[Workshop1.m];
      for (int i=0; i<Workshop1.m; ++i)
         if (Workshop1.data[i][0]<Workshop1.data[i][1])
            p1[k1++]=new Pair(Workshop1.data[i][0], i);
         else p2[k2++]=new Pair(Workshop1.data[i][1], i);
      Arrays.sort(p1, 0, k1, new Pair(0, 0));
      Arrays.sort(p2, 0, k2, new Pair(0, 0));
      for (int i=0; i<k1; ++i) perm[i]=p1[i].p;
      for (int i=0; i<k2; ++i) perm[k1+i]=p2[k2-1-i].p;
      int t1=0;
      for (int i=0; i<Workshop1.m; ++i){
         t1+=Workshop1.data[perm[i]][0];
         int t=Workshop1.data[perm[i]][1];
         ans = t1<ans ? ans+t : t1+t;
      }
      return ans;
   }
   public static void main(String[] args) {
      PrintWriter fout=null;
       try{ fout=new PrintWriter(new FileWriter("3a-12.txt")); }
       catch(IOException e){ System.out.println("Error!"); }
       long begin=System.currentTimeMillis();
       int mm=12; int[][] d=new int[mm][2];
       Random rm=new Random(mm);
       for (int i=0; i<mm; ++i) for (int j=0; j<2; ++j)
         d[i][j]=rm.nextInt(50)+1;
       for (int i=0; i<mm; ++i)
          fout.print(d[i][0]+", "+d[i][1]+";  ");
       fout.println();
       Workshop1.m=mm; Workshop1.data=d; Workshop1.out=fout;
       int a=wshop();
       for (int i=0; i<mm; ++i) fout.print(perm[i]+", ");
       fout.println(); fout.println(a);
       long end=System.currentTimeMillis();
       fout.println(mm+", jt:  Elapsed time: "+(end-begin));
       fout.close();
   }
}
```

```
17, 13;    7, 34;    25, 12;    1, 16;    47, 22;    19, 23;    12, 8;    46, 15;    37, 32;    6, 4;    41, 33;    17, 18;
3, 1, 11, 5, 10, 8, 4, 7, 0, 2, 6, 9,
281
12, jt:    Elapsed time: 0

2551785
1000, jt:    Elapsed time: 16
```

思考题 3.13：为稳定婚姻问题设计限界函数，用来求解稳定婚姻问题，通过比较检查限界函数的效果。

```
public class Stablem0 implements WMono {
   static PrintWriter out;
   static int m; static int[][] rmw, rwm, wmr,  mwr;
   static int[] temp=new int[100];
   int[] perm=new int[m]; int level, rm, rw;
   public boolean target(){ return level==m; }
   public int subnum(){ return level<m ? m : 0; }
   public boolean down(int i){
      int w=wmr[level][i]; boolean q=true;
      for (int k=0; q && k<level; ++k)
         q=w!=perm[k]&&
            (rmw[k][perm[k]]<rmw[k][w]||
                  rwm[w][level]<rwm[w][k])
            &&(rmw[level][w]<rmw[level][perm[k]]||
                  rwm[perm[k]][k]<rwm[perm[k]][level]);
      if (q){  perm[level]=w; rm+=i;
      rw+=rwm[w][level]; ++level; }
      return q; }
   public void up(){ --level; rm-=rmw[level][perm[level]];
      rw-=rwm[perm[level]][level];  }
   public Object cloneself(){ Stablem0 ans=new Stablem0();
      System.arraycopy(perm, 0, ans.perm, 0, m);
      ans.rm=rm; ans.rw=rw; return ans; }
   public void output(){
      for (int i=0; i<m; ++i) out.print(perm[i]+", ");
      out.println(); out.println(rm); }
   public int val(){ return rm; }
   public void setval(int k){ rm=k; }
   public static void shuffle(int[] a, int n){
      System.arraycopy(a, 0, temp, 0, 2*n);
      for (int i=0; i<2*n; ++i)
         a[i]= i%2==0 ? temp[n+i/2] : temp[i/2];
   }
   public static void mshuffle(int[] a, int n, int k){
      for (int i=0; i<k; ++i) shuffle(a, n);
```

```
    }
    public static void main(String[] args) {
        PrintWriter fout=null;
        try{ fout=new PrintWriter(new FileWriter("m0-24.txt")); }
        catch(IOException e){ System.out.println("Error!"); }
        long begin=System.currentTimeMillis();
        Stablem0.out=fout; int m=12;
        int[][] rmw=new int[2*m][2*m], rwm=new int[2*m][2*m],
            mwr=new int[2*m][2*m], wmr=new int[2*m][2*m];
        int[] lex0=new int[2*m];
        for (int i=0; i<2*m; ++i) lex0[i]=i;
        Random rm=new Random(m); int k=rm.nextInt(50)+1;
        mshuffle(lex0, m, k);
        JohnsonTrotter itr0=new JohnsonTrotter(2*m);
        for (int i=0; i<2*m; ++i){ k=rm.nextInt(1000)+1;
            for (int j=0; j<k; ++j) itr0.next();
            for (int j=0; j<2*m; ++j)
                wmr[lex0[i]][j]=itr0.content(j);
            k=rm.nextInt(50)+1; mshuffle(wmr[lex0[i]], m, k);
        }
        int[] lex1=new int[2*m];
        for (int i=0; i<2*m; ++i) lex1[i]=i;
        k=rm.nextInt(50)+1;
        mshuffle(lex1, m, k);
        JohnsonTrotter itr1=new JohnsonTrotter(2*m);
        for (int i=0; i<2*m; ++i){ k=rm.nextInt(1000)+1;
            for (int j=0; j<k; ++j) itr1.next();
            for (int j=0; j<2*m; ++j)
                mwr[lex1[i]][j]=itr1.content(j);
            k=rm.nextInt(50)+1; mshuffle(mwr[lex1[i]], m, k);
        }
        for (int i=0; i<2*m; ++i){
            for (int j=0; j<2*m; ++j)
                fout.print(wmr[i][j]+",  ");
            fout.println();
        }
        fout.println();
        for (int i=0; i<2*m; ++i){
            for (int j=0; j<2*m; ++j)
                fout.print(mwr[i][j]+",  ");
            fout.println();
        }
        fout.println();
        for (int i=0; i<2*m; ++i)
            for (int j=0; j<2*m; ++j){
```

```
            rmw[i][wmr[i][j]]=j; rwm[i][mwr[i][j]]=j; }
        Stablem0.m=2*m; Stablem0.mwr=mwr; Stablem0.rmw=rmw;
        Stablem0.rwm=rwm; Stablem0.wmr=wmr;
        BackOptSearch search=new BackOptSearch(new Stablem0());
        search.depthfirst(); WMono b=search.outans();
        ((Stablem0)b).output();
        long end=System.currentTimeMillis();
        fout.println(2*m+", jt: Elapsed time: "+(end-begin));
        fout.close();
    }
}
```

```
15, 18, 13, 0, 2, 7, 9, 8, 16, 23, 17, 11, 5, 20, 12, 3, 22, 19, 14, 10, 6, 21, 4, 1,
68
24, jt:    Elapsed time: 43540
```

算法欣赏 6：稳定婚姻问题的快速算法。

```
class Node { public int name, rank;
   public Node(int n, int r){ name=n; rank=r; }
}
public class Stable1 { static PrintWriter out;
   static int m; static int[][] rmw, rwm, wmr,  mwr;
   static int[] temp=new int[100];
   int[] perm=new int[m]; int rm, rw;
   Node[] perm1=new Node[m];
   public void compperm(){ LinkedList q=new LinkedList();
      for (int i=0; i<m; ++i) q.addLast(new Node(i, 0));
      while (!q.isEmpty()){
        Node cur=(Node)q.removeFirst();
        int w=wmr[cur.name][cur.rank];
        if (perm1[w]==null){
           perm1[w]=cur; perm[cur.name]=w; }
        else if (rwm[w][perm1[w].name]<rwm[w][cur.name]){
          ++cur.rank; q.addLast(cur); }
        else{ ++perm1[w].rank; q.addLast(perm1[w]);
          perm1[w]=cur; perm[cur.name]=w; } } }
   public void compr(){ rm=rw=0;
      for (int i=0; i<m; ++i){
         rm+=rmw[i][perm[i]];
         rw+=rwm[i][perm1[i].name]; } }
   public void output(){
      for (int i=0; i<m; ++i)
         out.print(perm[i]+", ");
      out.println(); out.println(rm); }
   public static void shuffle(int[] a, int n){
      System.arraycopy(a, 0, temp, 0, 2*n);
      for (int i=0; i<2*n; ++i)
         a[i]= i%2==0 ? temp[n+i/2] : temp[i/2];
   }
   public static void mshuffle(int[] a, int n, int k){
      for (int i=0; i<k; ++i) shuffle(a, n);
   }
```

```
15, 18, 13, 0, 2, 7, 9, 8, 16, 23, 17, 11, 5, 20, 12, 3, 22, 19, 14, 10, 6, 21, 4, 1,
68
24, jt:    Elapsed time: 32

1556
200, jt:    Elapsed time: 187
```