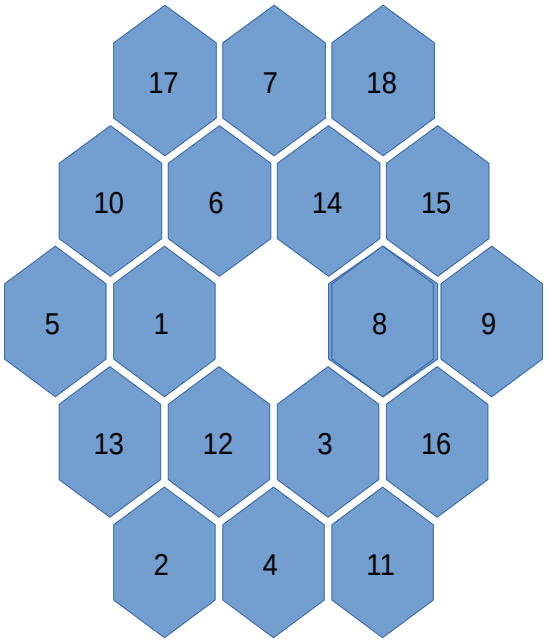
COSC 410: Artificial Intelligence

Assignment #2: Hexagonal Puzzle

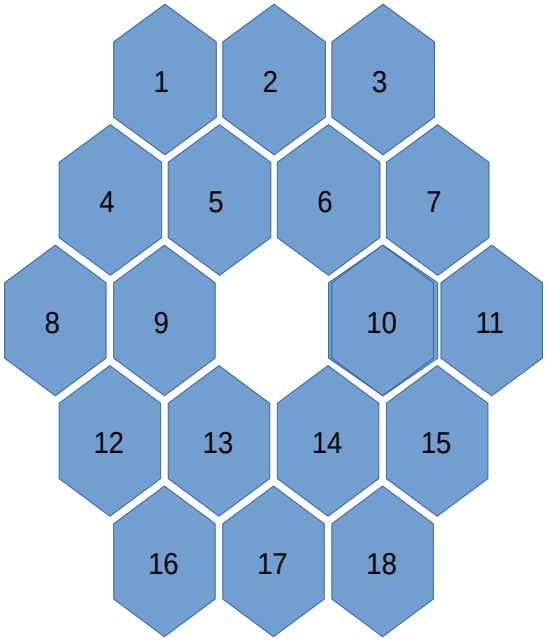
Due: 2/5/24

Points: 100

Consider a variant of the 8-puzzle where there are 18 hexagonal tiles plus a blank, arranged in the shape of a larger hexagon. An example would be:



The rules of the puzzle are the same as that of the 8-puzzle. A tile may be moved to the blank space in each move (or you can think of it as moving the blank). The goal is to get all the tiles in the following order using the fewest number of moves:



1) Write an program that uses the A* algorithm to solve this puzzle, and
2) write a program that uses iterative-deepening to solve this puzzle.

The program should use a consistent heuristic measure for h. One possible function is similar to the Manhattan distance: For any non-blank tile, find the difference in the x-direction from where the tile is to where it should be in the solution and add that to the difference in the y-direction from where the tile is to where it should be. Sum up these numbers for all the non-blank tiles to calculate h.

The program should read in the starting state from standard input and print out the states that lead to the goal to standard output. The input and output will consist of just the numbers, not the lines as shown above. The number 0 will indicate the blank in the input and the output.

Sample Run

```
Step 0:
       1     5    10
    0    16     2     3
 4    14    13    17    11
   12     8     6    18
```

```
       7    15     9

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Step 20:
       1     5    10
    4    14     2     3
 8    13     0     7    11
   12     6    17    15
      16     9    18

Step 21:
       1     5    10
    4     0     2     3
 8    13    14     7    11
   12     6    17    15
      16     9    18

Step 22:
       1     0    10
    4     5     2     3
 8    13    14     7    11
   12     6    17    15
      16     9    18

Step 23:
       1     2    10
    4     5     0     3
 8    13    14     7    11
   12     6    17    15
      16     9    18

Step 24:
       1     2     0
    4     5    10     3
 8    13    14     7    11
   12     6    17    15
      16     9    18

Step 25:
       1     2     3
    4     5    10     0
 8    13    14     7    11
   12     6    17    15
      16     9    18

Step 26:
       1     2     3
    4     5    10     7
 8    13    14     0    11
   12     6    17    15
      16     9    18

Step 27:
       1     2     3
    4     5    10     7
 8    13     0    14    11
   12     6    17    15
      16     9    18

Step 28:
```

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

Step 29:

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

Step 30:

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

Step 31:

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

Step 32:

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

Step 33:

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

Step 34:

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

Step 35:

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

Step 36:

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

Hint:
The function to find the distance between hexagons is different from
the Manhattan distance. I've implement the distance method as follows:

1. I consider moving a hex one square left or right as moving it 1
unit distance, and moving it up or down as moving it 1/2 unit
distance. However, when a hexagon is moved vertically, it is
simultaneously moved horizontally. To count the number of moves, the
horizontal distance can be ignored, if it is less than or equal to the
vertical distance (as it is already included,) but if the horizontal
distance is greater than the vertical distance, the overage must be
added.

2. I multiplied all distances by 2, to avoid using 1/2 unit distances.

3. I've assigned the rows (x) and columns (y) to the hexagons
according to the following chart. The chart shows the conversion
between the index of the array of tiles (0-19) to the Cartesian
coordinates of the tile.

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

3. Although the explanation is complicated, the code is simple. To
find the distance between two hexagons use:

```
int dx = Math.abs(x1-x2);
int dy = Math.abs(y1-y2);
distance = Math.max(dy-dx, 0) / 2 + dx;
```