**Šestka, verze Loydovy patnáctky**

1. Vygeneruji vsech 720 (6!) permutaci pro [0…5] cisel pro List[List[int]] data, tedy 2 x 3 matici.

def perm(a, k=0):

   if k == len(a):

      print(a)

   else:

      for i in range(k, len(a)):

         a[k], a[i] = a[i] ,a[k]

         perm(a, k+1)

         a[k], a[i] = a[i], a[k]

1. Vsechny tyto permutace vlozim iterativne jako argument do funkce nize

def slidingPuzzle(board):

    s = ''.join(str(c) for row in board for c in row) # z 2d matrix do 1d string

    q = collections.deque([(s, s.index('0'), 0)])

    m, n, seen = len(board), len(board[0]), {s}

    while q:

        t, i, cnt = q.popleft()

        if t == '123450': return cnt

        for d in (-1, 1, -n, n):

            j = i+d

            if j<0 or j>= m\*n or i\*j == 6: continue  # omezujici podminky

            s=t[:i]+t[j]+t[i+1:]

            s=s[:j]+'0'+s[j+1:]

            if s not in seen:

                seen.add(s)

                q.append((s, j, cnt+1))

    return -1

1. Setridim sestupne dle poctu kroku k dosazeni urcite konfigurace a vyfiltruji jen ty s nejvyssim poctem kroku (**nejvzdalenejsi konfigurace**), tedy rekl jsem si, ze to budou [19, 20, 21].

|  |  |
| --- | --- |
| Konfigurace | Pocet Kroku |
| [4, 5, 0], [1, 2, 3] | 21 |
| [3, 2, 1], [5, 4, 0] | 20 |
| [3, 2, 1], [0, 5, 4] | 20 |
| [4, 2, 5], [1, 3, 0] | 20 |
| [4, 3, 2], [1, 5, 0] | 20 |
| [4, 5, 3], [1, 2, 0] | 20 |
| [4, 0, 5], [1, 2, 3] | 20 |
| [2, 1, 0], [3, 5, 4] | 19 |
| [3, 2, 1], [5, 0, 4] | 19 |
| [3, 2, 0], [5, 4, 1] | 19 |
| [4, 2, 5], [1, 0, 3] | 19 |
| [4, 2, 0], [1, 3, 5] | 19 |
| [4, 3, 2], [1, 0, 5] | 19 |
| [4, 3, 0], [1, 5, 2] | 19 |
| [4, 5, 3], [1, 0, 2] | 19 |
| [0, 2, 1], [3, 5, 4] | 19 |
| [0, 3, 2], [5, 4, 1] | 19 |
| [0, 3, 1], [5, 2, 4] | 19 |
| [0, 4, 5], [1, 2, 3] | 19 |