MP307 Practical 2 Queueing Theory

Dara Corr, ID = 18483836

Python functions

The Python functions contained in this notebook are:

1. `Nearneigh(pup,pdown,r)` . This generates the transition matrix `P` for a simple nearest neighbour model with maximum size `r` and probability `pup` of one step up transition and `pdown` for one step down transition. The output is the transition matrix.
2. `Equilibrium(P)` . This computes the equilibrium probabilities for a given transition matrix `P` . If the system is not ergodic then an error message appears. The output is an array.
3. `Queue(P,n0,nit)` . This simulates a queue for a given transition matrix `P` with initial queue size `n0` for `nit` iterations. The output is a list of simulated queue data.
4. `Qplot(qdata,pi)` . This generates an animation of the queue for any input data list `qdata` . The second argument is optional and consists of the equilibrium probabilities `pi` (if they exist and have been calculated via `Equilibrium(P)` above or otherwise). The output is a Python animation where 50 plots are shown for the normalised frequency of events which can be compared to a plot of the equilibrium probabilities `pi` if provided.

In [1]:
```
%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import matplotlib.path as path
import matplotlib.animation as animation
from matplotlib.patches import Rectangle
import random as ran
```

In [2]:
```
def Nearneigh (pup, pdown, r):
    # This generates the transition matrix P for a nearest neighbour model wit
h
    # maximum size r and probability pup of one step up transition and pdown
    # for one step down transition.
    # The output is the transition matrix P.
    P = np.zeros((r + 1, r + 1))
    P[0, 0] = 1 - pup
    P[0, 1] = pup
    P[r , r ] = 1 - pdown
    P[r , r - 1] = pdown
    for i in range(1, r):
        P[i, i - 1] = pdown
        P[i , i] = 1 - pdown - pup
        P[i , i + 1] = pup
    return(P)
```

Example. Find the transition matrix for a queue of maximum size 5 with prob customer arrival=0.2 and prob customer servicing=0.3

In [3]:
```
P=Nearneigh(0.2,0.3,5)
print(P)
```

```
[[0.8 0.2 0.  0.  0.  0. ]
 [0.3 0.5 0.2 0.  0.  0. ]
 [0.  0.3 0.5 0.2 0.  0. ]
 [0.  0.  0.3 0.5 0.2 0. ]
 [0.  0.  0.  0.3 0.5 0.2]
 [0.  0.  0.  0.  0.3 0.7]]
```

In [5]:
```
type(P)
```

Out[5]: numpy.ndarray

In [4]:
```
def Equilibrium (P):
    # This computes the equilibrium probabilities pi for a given transition ma
trix P.
    # If the system is not ergodic then an error message appears.
    # The output is an array.
    degenflag = 0
    n = len(P)
    pi = np.zeros((n))
    for i in range(n):
        if float(sum(P[i,:]))!= 1.0:
            print("ERROR: row",i,"of P does not sum to 1 !")
    eigenvals, eigenvects=np.linalg.eig(P.transpose()) # eigendata for P trans
pose
    for k in range(n):
        if abs(eigenvals[k]-1)< 1.0e-14:
            degenflag = degenflag + 1
            k1=k # label for eigenvalue 1.
    if 1 < degenflag:
        print("ERROR: P is not Ergodic")
    else:
        v=eigenvects[:,k1]
        pi=v/sum(v)
    return(pi)
```

In [7]:
```
pi=Equilibrium(P)
print(pi)
```

```
[0.36541353 0.24360902 0.16240602 0.10827068 0.07218045 0.0481203 ]
```

In [ ]:
```

```

In [5]:
```
def Queue (P, n0, nit):
# This simulates a queue for a given transition matrix P with initial queue si
ze n0 for nit iterations.
# The output is a list of simulated queue data.
    nP = len(P)
# pp is a relabelled transition matrix with row elements in decreasing order.
# This is used to speed up the simulation process
# The relabelling is stored in the matrix loc.
    pp = np.zeros((nP,nP))
    for i in range(nP):
        for j in range(nP):
            pp[i, j] = P[i, j]
    loc = np.array([list(range(nP))]*nP)
    for i in range(nP):
        for j in range(nP-1):
            maxpp = pp[i, j]
            kmax = j
            for k in range(j, nP):
                if maxpp < pp[i, k]:
                    maxpp = pp[i, k]
                    kmax = k
                    pp[i, j], pp[i, k] = pp[i, k], pp[i, j]
                    loc[i, j], loc[i, k] = loc[i, k], loc[i, j]
    n = n0
    qseq = [n0]
    if n0 < 0 or nP-1 < n0:
        print("ERROR: inputted initial Q size : Qsize<0 or Qsize>",nP-1)
    for m in range(nit):
        x = ran.random() # generates uniform random no on [0,1]
        k = 0
        x1 = pp[n, k]
        while (x1 < x):
            k = k + 1
            x1 = x1 + pp[n, k]
        n = loc[n, k]
        qseq = qseq + [n]
    return(qseq)
```

In [ ]:
```
```

In [9]:
```
print(P) # For the transition matrix in our example above
```

```
[[0.8 0.2 0.  0.  0.  0. ]
 [0.3 0.5 0.2 0.  0.  0. ]
 [0.  0.3 0.5 0.2 0.  0. ]
 [0.  0.  0.3 0.5 0.2 0. ]
 [0.  0.  0.  0.3 0.5 0.2]
 [0.  0.  0.  0.  0.3 0.7]]
```

In [10]:
```
qdata=Queue(P,2,1000) # simulate 1000 steps with intial state 2 for given P
```

In [11]:
```
len(qdata) # notice there are 1001 data pts for 1000 steps
```

Out[11]:
```
1001
```

In [ ]:
```

```

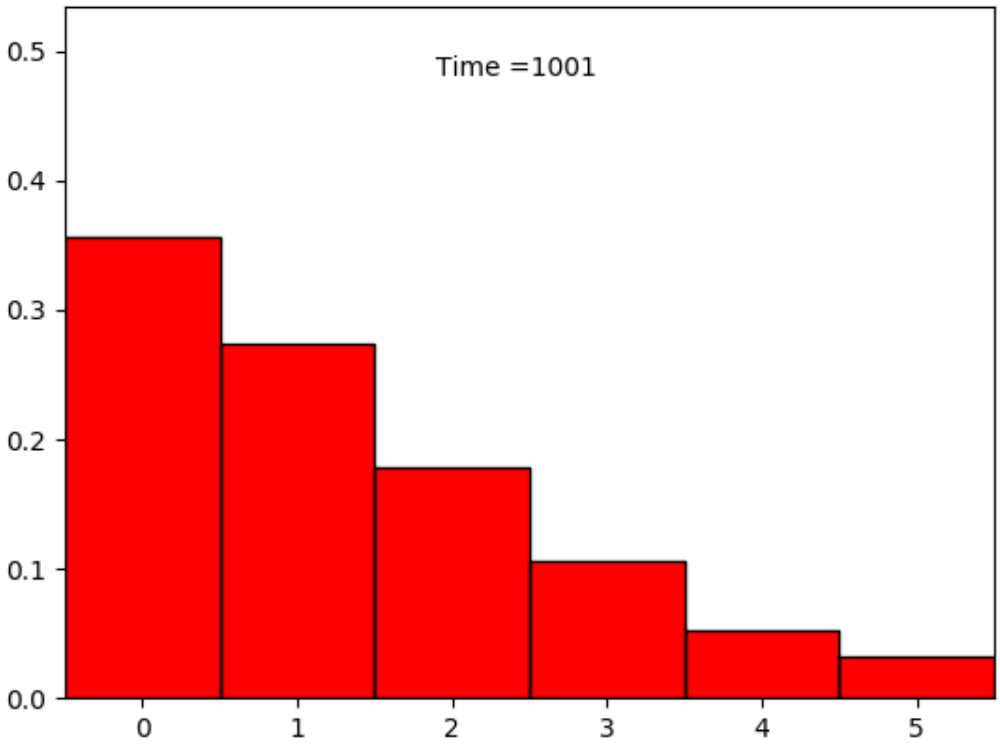
In [6]:
```
def Qplot(qdata,pi=0):
    # This generates up to 50 animations of the queue for any input data list
 qdata
    # The second argument is optional and consists of the equilibrium probabil
ities pi
    # (if they exist and have been calculated via Equilibrium(P) above).
    # The output is an animation of 50 plots of the normalised frequency of ev
ents.
    # This is compared to the equilibrium probabilities pi if provided.

    # Nsim animations
    Nsim=50

    Npoints=len(qdata)
    qmax=max(qdata)

    shift = lambda x: x-1/2
    inbins=list(map(shift,list(range(qmax+2)))) # bins -1/2,1/2,3/2,...
    n,bins  = np.histogram(qdata, bins=inbins)

    freq=np.array(n)/Npoints

    # get the corners of the rectangles for the histogram

    left = bins[:-1]
    right = bins[1:]
    bottom = np.zeros(len(left))
    top = bottom + freq
    nrects = len(left)

    # Here comes the tricky part -- we have to set up the vertex and path code
s
    # arrays using `.Path.MOVETO`, `.Path.LINETO` and `.Path.CLOSEPOLY` for ea
ch rect.

    #* We need 1 ``MOVETO`` per rectangle, which sets the initial point.
    #* We need 3 ``LINETO``'s, which tell Matplotlib to draw lines from
    #  vertex 1 to vertex 2, v2 to v3, and v3 to v4.
    #* We then need one ``CLOSEPOLY`` which tells Matplotlib to draw a line fr
om
    #  the v4 to our initial vertex (the ``MOVETO`` vertex), in order to close
the
    #  polygon.

    #<div class="alert alert-info"><h4>Note</h4><p>The vertex for ``CLOSEPOLY`
` is ignored, but we still need a placeholder
    #  in the ``verts`` array to keep the codes aligned with the vertices.</p>
</div>

    nverts = nrects * (1 + 3 + 1)
    verts = np.zeros((nverts, 2))
    codes = np.full(nverts, path.Path.LINETO)
    codes[0::5] = path.Path.MOVETO
    codes[4::5] = path.Path.CLOSEPOLY
    verts[0::5, 0] = left
    verts[0::5, 1] = bottom
```

```
    verts[1::5, 0] = left
    verts[1::5, 1] = top
    verts[2::5, 0] = right
    verts[2::5, 1] = top
    verts[3::5, 0] = right
    verts[3::5, 1] = bottom

    #To animate the histogram, we use the ``qanimate`` function, which updates
the locations of the vertices for the
    #histogram (in this case, only the heights of each rectangle).

    def qanimate(i):
        # simulate data
        Ni_points=int(Npoints*i/Nsim)
        datai=qdata[1:Ni_points] # first Ni_points
        ni, bin = np.histogram(datai, bins=inbins)
        freq=np.array(ni)/Ni_points
        top = bottom + freq
        verts[1::5, 1] = top
        verts[2::5, 1] = top
        time_text.set_text('Time =' +str(Ni_points))

    #``patch`` will eventually be a `.Patch` object.

    patch = None

    fig, ax = plt.subplots()
    barpath = path.Path(verts, codes)
    patch = patches.PathPatch(
        barpath, facecolor='red', edgecolor='black', alpha=1)
    ax.add_patch(patch)

    time_text = ax.text(0.4, 0.9, '', transform=ax.transAxes)

    # draw rectangle for equilibrium probs pi
    if not type(pi) is int:
        for m in range(len(pi)):
            heightpi=pi[m]
            widthpi=1
            rectpi = Rectangle((m-1/2, 0), widthpi, heightpi, fc='white', ec =
'black',lw = 2, alpha=0.5)
            ax.add_patch(rectpi)

    ax.set_xlim(-1/2, qmax+1/2)
    ymax=max(freq)*1.5
    ax.set_ylim(0, ymax)

    ani = animation.FuncAnimation(fig, qanimate, Nsim+1,repeat=False, blit=Tru
e)
    plt.show()
    return ani
```

In [33]:
```
print(qdata)
```

```
[2, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 0, 0, 1, 2, 2, 1, 2, 2, 1, 1,
2, 2, 2, 2, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 2, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 2, 2, 2, 3, 3, 3, 4, 4, 3, 2, 2, 3, 4, 5, 5, 5, 5, 5, 5, 4, 4, 3, 2, 3, 3,
3, 2, 1, 1, 2, 2, 2, 3, 3, 2, 2, 3, 3, 3, 2, 3, 2, 3, 3, 2, 1, 2, 2, 1, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0,
0, 1, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 3, 3, 2, 1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 2, 3, 2, 1, 2, 3, 2, 2, 1, 0, 0, 1, 2,
2, 1, 1, 0, 0, 1, 2, 1, 1, 2, 2, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 1, 2, 1, 0, 0, 0, 0, 0, 1, 2, 1, 1, 0, 1, 2, 1, 1, 1, 1, 1, 1, 1,
1, 2, 2, 2, 2, 3, 3, 2, 1, 0, 0, 0, 0, 0, 1, 2, 1, 1, 2, 2, 3, 2, 2, 2, 2, 1,
1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 2, 3, 2, 2, 2, 3, 3, 3, 4, 5, 4, 5, 4,
4, 4, 3, 3, 3, 4, 4, 3, 3, 2, 2, 2, 3, 3, 3, 4, 4, 4, 3, 2, 1, 2, 1, 0, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 3, 4, 4, 4, 3, 2, 3, 3, 3, 3, 2, 2, 2,
3, 3, 3, 3, 3, 2, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 2, 2, 2, 2,
1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 3, 4, 4,
4, 3, 2, 3, 3, 3, 3, 3, 4, 5, 5, 4, 4, 4, 3, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3,
3, 3, 3, 4, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 3, 3, 4, 4, 5, 4, 4, 3, 3, 2, 2, 2,
2, 1, 1, 1, 2, 3, 3, 3, 4, 5, 5, 4, 4, 4, 4, 4, 5, 5, 4, 5, 5, 5, 5, 5, 4, 5,
5, 4, 3, 3, 3, 3, 4, 3, 2, 2, 3, 4, 3, 3, 3, 2, 1, 2, 3, 3, 3, 2, 2, 3, 2, 2,
1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 0, 0,
0, 1, 1, 0, 0, 1, 1, 2, 1, 1, 2, 1, 2, 2, 2, 1, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0,
0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
4, 4, 3, 3, 3, 4, 4, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3, 2, 3, 3, 4, 3, 3, 2, 1, 2,
1, 1, 2, 2, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 5, 4, 3, 4, 5, 4,
5, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 3, 3, 2, 1, 2, 1, 1, 2, 2, 1, 1,
2, 2, 2, 3, 3, 3, 2, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 3, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1,
0, 1, 2, 3, 4, 4, 3, 4, 4, 4, 5, 5, 5, 4, 5, 4, 3, 2, 2, 3, 2, 3, 2, 2, 2, 2,
2, 2, 2, 1, 0, 1, 1, 2, 2, 1, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1, 1, 2, 2, 3, 3, 4,
3, 3, 2, 1, 1, 1, 1, 0, 0, 0, 0, 1, 2, 2, 2, 1, 0, 1, 2, 2, 2, 2, 3, 2, 2, 1,
1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0]
```

In [15]:
```
Qplot(qdata) # animation of given qdata without pi given
```

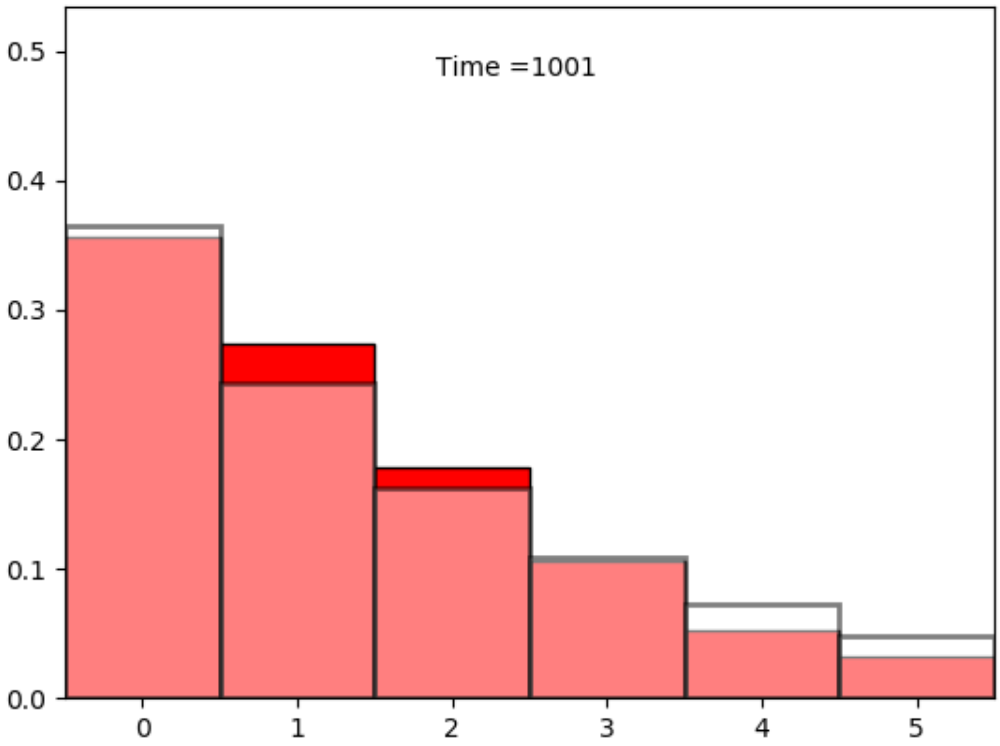


Out[15]: <matplotlib.animation.FuncAnimation at 0xecc7fd9b38>

In [15]:
```
print(pi)
```

```
[0.36541353 0.24360902 0.16240602 0.10827068 0.07218045 0.0481203 ]
```

In [16]:
```
Qplot(qdata,pi) # animation of given qdata with pi given
```



Out[16]: <matplotlib.animation.FuncAnimation at 0xecc82e5e48>

In [ ]:
```

```

Q.1

Simulate the Markov process with transition matrix:

$$\begin{bmatrix} 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 1/4 & 0 & 1/2 & 0 & 1/4 \\ 1/4 & 0 & 1/4 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \end{bmatrix}$$

This is the same example as in Q.3 of Practical 1. Observe the long-time behaviour for initial queue size 0 and 1.

Observe what happens if the first row is changed to $[1/4, 1/4, 1/2, 0, 0, 0]$

In [35]:
```
#Matrix P1
P=np.array([
    [1/2,0,1/2,0,0,0],
    [0,1/2,0,1/2,0,0],
    [0,0,1/2,0,1/2,0],
    [0,1/4,0,1/2,0,1/4],
    [1/4,0,1/4,0,1/2,0],
    [0,0,0,1/2,0,1/2]
    ])
print(P)
```

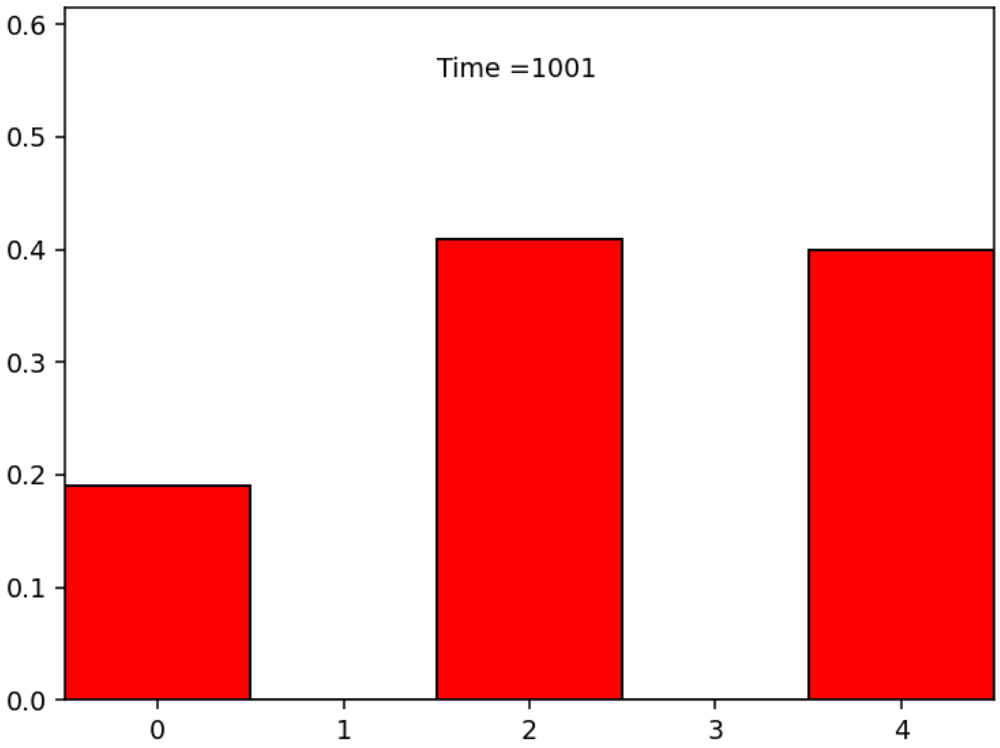
```
[[0.5  0.   0.5  0.   0.   0.  ]
 [0.   0.5  0.   0.5  0.   0.  ]
 [0.   0.   0.5  0.   0.5  0.  ]
 [0.   0.25 0.   0.5  0.   0.25]
 [0.25 0.   0.25 0.   0.5  0.  ]
 [0.   0.   0.   0.5  0.   0.5 ]]
```

In [28]:
```
qdata=Queue(P,0,1000)  # 1000 iterations with initial state 0
```

In [29]:
```
print(qdata)
```

```
[0, 2, 2, 4, 4, 0, 2, 2, 4, 4, 0, 0, 2, 4, 4, 0, 0, 0, 0, 0, 2, 2, 4, 4, 2,
4, 4, 0, 2, 2, 2, 4, 2, 4, 0, 2, 2, 4, 2, 4, 0, 2, 2, 2, 4, 4, 4, 0, 2, 4, 4,
4, 0, 0, 2, 2, 4, 4, 2, 2, 4, 2, 4, 2, 4, 2, 2, 4, 0, 2, 4, 2, 2, 4, 4, 4, 4,
2, 4, 4, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 2, 2, 4, 4, 0, 0, 2, 2, 4,
0, 0, 2, 2, 2, 2, 2, 4, 4, 4, 0, 0, 2, 4, 4, 4, 2, 4, 2, 4, 4, 0, 2, 4, 2, 2,
4, 2, 2, 4, 2, 4, 4, 4, 4, 2, 2, 2, 4, 0, 2, 2, 4, 4, 4, 0, 0, 0, 2, 2, 4, 4,
4, 4, 0, 0, 0, 2, 2, 2, 4, 4, 0, 2, 4, 0, 2, 4, 2, 4, 0, 0, 0, 2, 4, 4, 0, 2,
4, 4, 4, 4, 4, 4, 0, 2, 4, 2, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 4, 2, 2, 2,
2, 2, 2, 4, 2, 2, 2, 2, 4, 2, 2, 4, 0, 2, 2, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 2,
2, 2, 2, 2, 2, 4, 4, 0, 2, 2, 4, 0, 0, 2, 2, 4, 4, 4, 2, 2, 2, 4, 0, 2, 2, 2,
2, 4, 4, 0, 2, 2, 2, 2, 4, 2, 2, 2, 4, 4, 4, 4, 2, 2, 2, 4, 2, 4, 4, 4, 2, 4,
4, 4, 4, 4, 0, 2, 2, 4, 0, 2, 4, 4, 4, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 2, 2, 2,
4, 4, 4, 2, 4, 2, 4, 0, 2, 2, 2, 4, 4, 4, 4, 4, 0, 2, 2, 2, 4, 4, 4, 2, 4, 2,
2, 4, 4, 4, 2, 2, 4, 4, 0, 2, 4, 2, 4, 0, 2, 2, 4, 0, 0, 2, 2, 4, 2, 4, 4, 4,
4, 4, 4, 0, 0, 0, 0, 0, 2, 4, 4, 0, 2, 4, 0, 0, 2, 4, 4, 0, 0, 0, 0, 2, 2, 4,
2, 2, 4, 4, 4, 0, 2, 2, 4, 0, 2, 4, 0, 0, 0, 0, 0, 2, 2, 4, 2, 4, 2, 2, 2, 4,
4, 0, 2, 2, 2, 4, 2, 4, 0, 0, 2, 4, 0, 0, 2, 4, 0, 2, 2, 2, 4, 4, 4, 4, 0, 0,
0, 0, 0, 2, 4, 2, 4, 0, 2, 4, 4, 0, 2, 4, 4, 0, 2, 4, 0, 2, 4, 2, 2, 4, 4, 4,
4, 2, 4, 4, 4, 0, 2, 4, 4, 4, 4, 4, 2, 4, 2, 2, 2, 4, 0, 2, 4, 0, 0, 2, 2, 2,
2, 4, 0, 0, 0, 0, 2, 4, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0, 2, 4, 2, 2, 2, 4, 4, 2,
2, 4, 2, 4, 2, 4, 4, 4, 2, 4, 4, 4, 4, 2, 2, 2, 4, 2, 2, 2, 2, 2, 4, 0, 2, 2,
2, 2, 2, 2, 4, 2, 2, 4, 0, 0, 2, 2, 4, 4, 2, 4, 4, 2, 4, 4, 2, 4, 4, 2, 2, 4,
0, 2, 4, 2, 2, 4, 4, 4, 2, 4, 4, 2, 2, 2, 2, 4, 4, 4, 4, 4, 2, 4, 4, 0, 0, 2,
2, 2, 2, 4, 4, 4, 2, 2, 2, 2, 2, 4, 0, 2, 4, 2, 2, 4, 4, 4, 0, 0, 0, 2, 4, 0,
2, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 4, 0, 0, 2, 4, 2, 4, 2, 4, 2, 2, 4, 4, 2, 4,
4, 0, 0, 0, 2, 4, 4, 0, 0, 0, 2, 2, 4, 0, 0, 2, 4, 4, 4, 4, 4, 0, 2, 4, 2, 4,
4, 4, 2, 2, 2, 2, 2, 4, 4, 2, 2, 4, 0, 0, 0, 2, 2, 2, 2, 4, 4, 0, 2, 2, 4, 0,
0, 0, 0, 2, 4, 0, 2, 2, 4, 0, 2, 2, 4, 4, 4, 4, 2, 2, 4, 0, 0, 2, 4, 2, 2, 2,
4, 0, 0, 2, 2, 2, 4, 4, 4, 0, 2, 4, 2, 2, 4, 2, 2, 2, 4, 4, 4, 4, 4, 2, 2, 2,
2, 4, 2, 2, 4, 4, 2, 4, 0, 0, 0, 0, 2, 4, 4, 4, 2, 4, 4, 0, 0, 0, 0, 0, 0, 2,
4, 4, 4, 0, 0, 0, 2, 2, 2, 2, 2, 4, 0, 0, 2, 4, 0, 2, 4, 4, 4, 2, 2, 4, 0, 0,
2, 4, 0, 2, 2, 2, 2, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 4, 2, 2, 4, 4, 2, 2, 2, 4,
2, 2, 2, 4, 0, 2, 4, 4, 0, 0, 2, 2, 2, 4, 4, 4, 0, 0, 0, 0, 0, 2, 4, 0, 2, 2,
2, 4, 0, 2, 2, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 4, 2, 4, 4, 0, 0, 2,
4, 4, 4, 0, 0, 0, 0, 2, 4, 2, 4, 2, 2, 2, 4, 4, 0, 0, 0, 0, 0, 0, 2, 4, 2, 4,
4, 4, 4, 0, 2, 2, 4, 2, 2, 2, 4, 0, 0, 2, 4, 0, 0, 2, 2, 2, 2, 4, 4, 2, 4, 0,
0, 0, 2, 4, 4, 2, 4, 2, 4, 4, 0, 2, 4, 0, 2, 2, 4, 0, 0, 0, 2, 2, 4, 4, 4, 4,
0, 2, 4, 4, 0, 0, 0, 2, 4, 4, 2, 2, 2, 2, 4, 0, 2, 4, 0, 2, 4, 4, 2, 4, 4, 4,
4, 4, 0, 2, 4, 2, 2, 2, 4, 2, 4, 4, 4, 4]
```

In [20]:
```
Qplot(qdata)
```



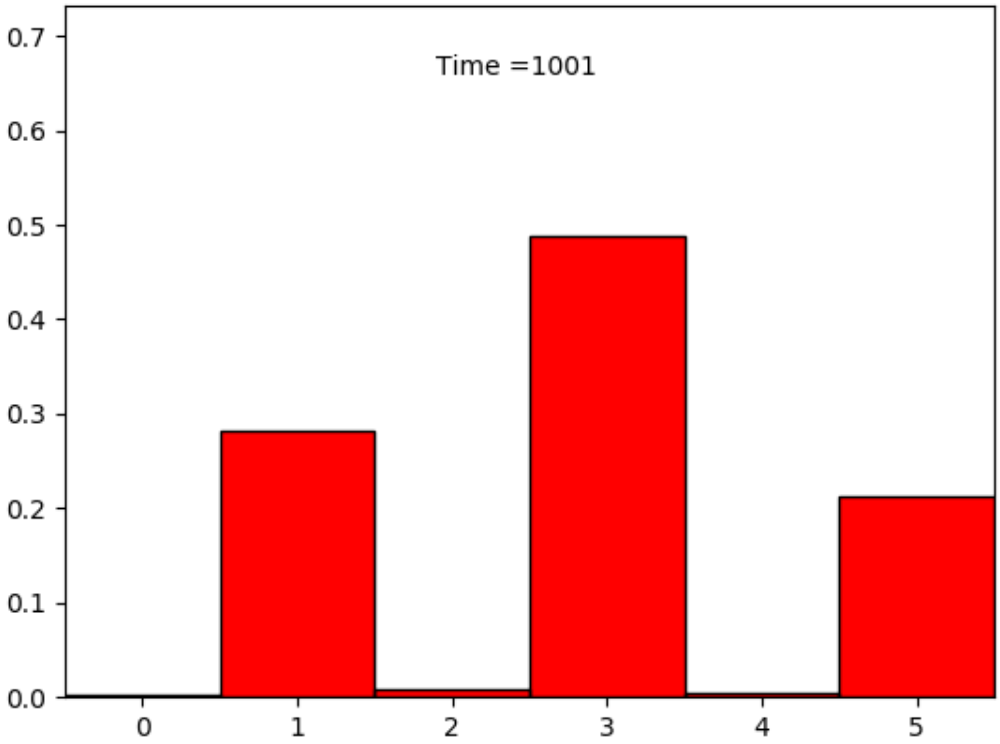
Out[20]: <matplotlib.animation.FuncAnimation at 0x981236cdf0>

In [36]:
```
#P2
P=np.array([
    [1/4,1/4,1/2,0,0,0],
    [0,1/2,0,1/2,0,0],
    [0,0,1/2,0,1/2,0],
    [0,1/4,0,1/2,0,1/4],
    [1/4,0,1/4,0,1/2,0],
    [0,0,0,1/2,0,1/2]
    ])
print(P)
```

```
[[0.25 0.25 0.5  0.   0.   0.  ]
 [0.   0.5  0.   0.5  0.   0.  ]
 [0.   0.   0.5  0.   0.5  0.  ]
 [0.   0.25 0.   0.5  0.   0.25]
 [0.25 0.   0.25 0.   0.5  0.  ]
 [0.   0.   0.   0.5  0.   0.5 ]]
```

In [37]:
```
qdata=Queue(P,0,1000)  # 1000 iterations with initial state 0

print(qdata)

Qplot(qdata)
```

```
[0, 2, 2, 2, 2, 4, 0, 2, 2, 4, 4, 2, 2, 2, 4, 0, 1, 1, 3, 1, 1, 1, 3, 3, 1,
1, 3, 1, 1, 1, 1, 1, 3, 5, 3, 3, 3, 3, 5, 3, 1, 1, 3, 3, 5, 5, 3, 3, 3, 3, 3,
1, 1, 3, 1, 1, 3, 3, 3, 3, 5, 5, 3, 3, 1, 1, 3, 5, 5, 3, 1, 1, 3, 5, 5, 5, 5,
3, 3, 1, 1, 3, 5, 3, 3, 3, 3, 1, 1, 1, 1, 3, 3, 3, 1, 1, 3, 1, 1, 3, 5, 3, 1,
3, 5, 5, 5, 5, 5, 3, 3, 5, 5, 5, 3, 5, 3, 3, 1, 1, 1, 1, 3, 3, 5, 3, 1, 1, 3,
3, 1, 3, 3, 3, 3, 3, 1, 3, 1, 1, 3, 3, 3, 1, 1, 1, 3, 3, 5, 5, 3, 1, 3, 3, 5,
5, 3, 3, 3, 3, 1, 1, 1, 1, 1, 3, 3, 3, 1, 3, 3, 1, 3, 3, 5, 3, 5, 3, 1, 3, 1,
1, 1, 1, 3, 3, 5, 3, 3, 1, 3, 3, 5, 3, 3, 3, 3, 1, 1, 1, 1, 3, 5, 5, 3, 1, 3,
5, 5, 3, 3, 3, 1, 3, 3, 1, 1, 1, 3, 3, 3, 5, 3, 3, 5, 3, 3, 1, 1, 3, 3, 3, 3,
1, 1, 1, 1, 3, 1, 3, 5, 3, 5, 5, 5, 3, 3, 3, 5, 3, 5, 5, 3, 3, 3, 5, 3, 3, 3,
1, 3, 3, 3, 5, 5, 3, 3, 5, 3, 5, 5, 3, 5, 3, 5, 5, 3, 3, 5, 3, 3, 5, 5, 3, 1,
1, 3, 1, 1, 3, 3, 1, 3, 3, 1, 3, 3, 3, 1, 3, 3, 1, 3, 1, 3, 3, 1, 1, 1, 3, 1,
3, 1, 1, 3, 1, 3, 5, 3, 1, 1, 3, 5, 5, 3, 1, 1, 3, 1, 3, 3, 3, 5, 5, 5, 3, 5,
3, 1, 1, 1, 3, 1, 1, 3, 5, 5, 5, 3, 3, 3, 5, 5, 3, 1, 3, 1, 1, 3, 5, 3, 3, 5,
3, 1, 3, 3, 5, 3, 1, 3, 3, 1, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 5, 3, 5, 5, 5,
3, 5, 5, 5, 5, 3, 5, 3, 3, 1, 3, 5, 5, 3, 1, 3, 5, 5, 5, 3, 3, 1, 1, 1, 3, 3,
1, 3, 3, 3, 1, 3, 1, 3, 3, 1, 1, 3, 5, 3, 3, 3, 3, 5, 5, 5, 5, 3, 1, 1, 1, 1,
3, 5, 3, 1, 3, 3, 3, 3, 5, 3, 5, 3, 3, 1, 3, 1, 1, 3, 3, 5, 3, 3, 5, 5, 5, 3,
5, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 1, 1, 1, 1, 3, 3, 3, 3, 1, 3, 5, 5, 3, 1, 3,
3, 1, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 1, 3, 3, 1, 1, 1, 1, 1, 3, 1, 1, 3, 1, 1,
3, 3, 5, 3, 5, 3, 1, 1, 3, 3, 5, 3, 3, 5, 3, 5, 3, 3, 3, 1, 1, 1, 1, 3, 1, 3,
3, 1, 1, 1, 1, 3, 3, 3, 1, 3, 5, 3, 3, 3, 3, 5, 5, 5, 3, 1, 1, 3, 5, 5, 5, 5,
3, 3, 1, 1, 3, 3, 5, 5, 3, 3, 3, 1, 1, 3, 3, 1, 1, 1, 3, 3, 1, 3, 1, 3, 5, 3,
3, 1, 1, 3, 1, 3, 5, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 3, 3, 3,
1, 3, 3, 3, 5, 5, 5, 5, 3, 5, 3, 3, 3, 3, 3, 5, 3, 5, 3, 1, 3, 5, 3, 3, 1, 1,
3, 5, 3, 3, 5, 3, 3, 3, 5, 5, 3, 5, 3, 3, 5, 5, 5, 5, 5, 5, 3, 3, 3, 1, 3, 1,
1, 3, 1, 1, 3, 1, 1, 3, 3, 3, 3, 3, 1, 1, 1, 3, 1, 1, 1, 3, 3, 3, 1, 1, 3, 3,
1, 3, 3, 1, 3, 1, 3, 1, 1, 1, 3, 1, 3, 5, 3, 3, 5, 5, 5, 3, 5, 5, 3, 5, 5, 3,
1, 3, 5, 5, 5, 5, 3, 5, 5, 5, 3, 1, 1, 3, 3, 3, 3, 1, 3, 3, 1, 3, 3, 3, 5, 3,
5, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 3, 1, 1, 3, 5, 5, 3, 3, 3, 3, 3, 3, 1, 1,
1, 3, 1, 1, 1, 1, 3, 5, 5, 3, 3, 1, 3, 5, 3, 1, 3, 3, 3, 5, 3, 3, 3, 3, 5, 3,
1, 1, 1, 1, 3, 3, 3, 3, 3, 5, 3, 5, 3, 3, 3, 5, 5, 3, 5, 5, 3, 3, 3, 5, 5, 3,
5, 3, 3, 1, 1, 1, 3, 5, 3, 5, 3, 1, 3, 3, 3, 1, 3, 3, 5, 3, 3, 5, 5, 5, 3, 3,
1, 1, 3, 1, 3, 3, 1, 3, 1, 3, 3, 3, 1, 3, 5, 5, 3, 5, 5, 5, 5, 5, 3, 1, 3, 1,
3, 5, 5, 5, 5, 5, 5, 3, 3, 3, 3, 1, 1, 1, 1, 3, 3, 1, 3, 3, 3, 1, 3, 5, 3, 3,
5, 5, 3, 5, 5, 5, 5, 3, 1, 1, 1, 1, 1, 3, 3, 5, 3, 5, 3, 5, 5, 3, 3, 1, 1, 3,
5, 3, 1, 3, 3, 3, 3, 5, 3, 3, 1, 3, 5, 3, 5, 3, 3, 3, 3, 3, 1, 3, 5, 3, 1, 1,
1, 1, 1, 1, 3, 5, 3, 3, 1, 1, 1, 3, 5, 3, 3, 3, 3, 1, 3, 1, 3, 1, 1, 1, 1, 3,
3, 1, 3, 1, 3, 5, 3, 3, 3, 1, 1, 1, 1, 3]
```



Out[37]: <matplotlib.animation.FuncAnimation at 0xecc9186400>

Transition Matrix 1 is not Ergodic because for initial queue size 1 it only has queue sizes 1,3 or 5 and for initial size 0, it only has queue sizes 0, 2 and 4. To be Ergodic, the system needs to visit every state in the system so it would need to visit every queue size in the system up to the maximum queue size.

By changing the first row of the transition matrix to $[1/4, 1/4, 1/2, 0, 0, 0]$, this makes the system Ergodic since it now visits every queue size (up to the maximum queue size) at least once in the simulation above.

Q.2

A queue is observed 1000 time intervals with data as given in the array qdata below. This is the same as Question 4 of Practical 1.

Compare the actual frequencies of events with a simulated one. Use the probabilities found in Practical 1.

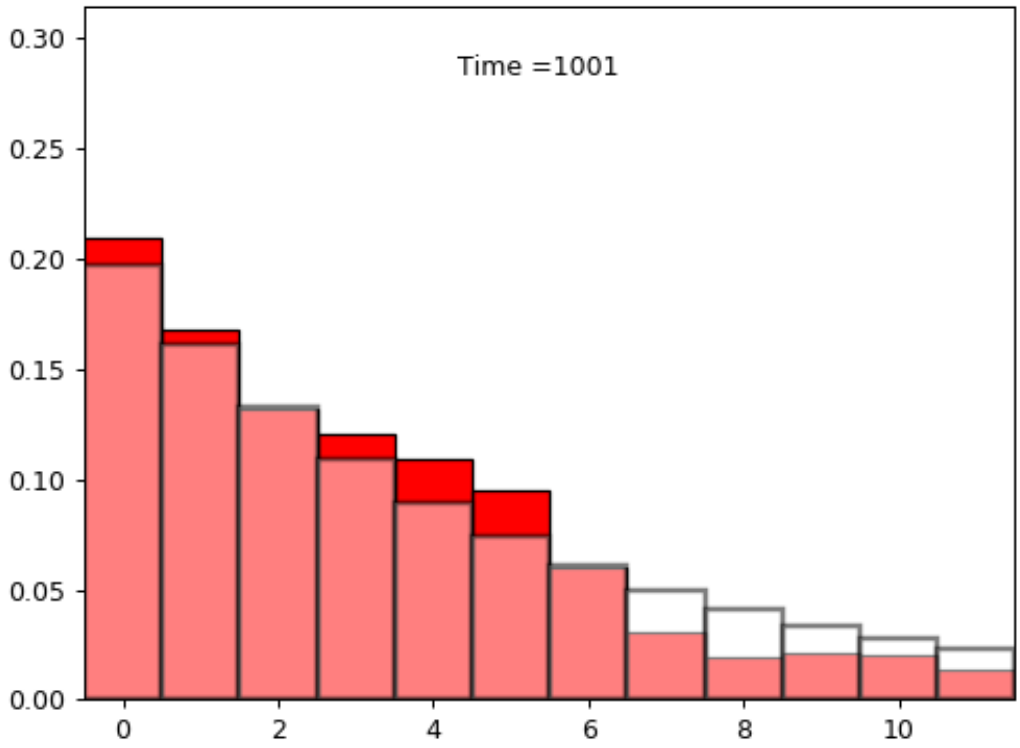
Look at that question again for examples of do loops and if else statements in Python

In [2]:
```
qdata = [4, 5, 6, 6, 6, 7, 6, 7, 6, 5, 4, 4, 5, 6, 7, 6, 5, 4, 3, 4, 5, 6, 5,
4, 3, 2, 1, 2, 3, 2, 1,
         2, 3, 4, 3, 2, 3, 2, 1, 1, 2, 2, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
2, 3, 4, 5, 4, 5, 6, 7,
         8, 7, 6, 7, 6, 6, 5, 4, 5, 4, 3, 2, 3, 2, 3, 2, 3, 2, 1, 1, 2, 3, 3,
4, 5, 6, 7, 6, 5, 6, 5,
         6, 5, 6, 5, 4, 5, 4, 3, 4, 3, 4, 3, 2, 1, 0, 0, 0, 0, 1, 2, 1, 0, 1,
2, 3, 3, 2, 3, 4, 3, 2,
         3, 2, 1, 2, 3, 2, 3, 2, 1, 0, 0, 1, 1, 2, 3, 2, 1, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 1, 2, 1, 2,
         1, 2, 3, 4, 3, 4, 5, 6, 5, 6, 5, 6, 7, 7, 6, 5, 4, 3, 4, 3, 3, 4, 3,
2, 3, 2, 3, 2, 1, 2, 1,
         2, 1, 0, 0, 0, 0, 1, 2, 3, 2, 2, 3, 2, 1, 0, 1, 2, 1, 0, 1, 0, 1, 0,
1, 2, 1, 0, 0, 1, 0, 1,
         0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1, 1, 0, 0, 1, 2, 3, 2, 3, 3, 4,
5, 6, 5, 4, 3, 2, 1, 2,
         1, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
0, 1, 0, 1, 0, 1, 1, 2,
         1, 0, 0, 1, 2, 3, 2, 3, 2, 2, 3, 4, 4, 5, 4, 3, 2, 3, 2, 3, 2, 3, 2,
1, 0, 1, 2, 3, 2, 1, 2,
         1, 0, 1, 0, 1, 2, 3, 3, 2, 1, 1, 2, 1, 2, 1, 2, 1, 2, 3, 2, 2, 1, 0,
1, 2, 1, 2, 3, 4, 4, 3,
         2, 2, 1, 2, 3, 4, 5, 4, 4, 5, 4, 3, 4, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, 1, 2,
         1, 2, 2, 3, 2, 3, 4, 5, 6, 5, 6, 5, 4, 5, 6, 7, 6, 5, 6, 6, 7, 6, 7,
7, 6, 7, 6, 5, 4, 5, 5,
         4, 5, 6, 6, 5, 6, 5, 6, 5, 6, 5, 5, 5, 4, 3, 4, 5, 6, 5, 4, 3, 3, 4,
3, 2, 3, 2, 3, 3, 2, 1,
         2, 1, 0, 0, 0, 0, 0, 1, 2, 3, 2, 3, 4, 5, 6, 7, 7, 8, 9, 8, 9, 8, 7,
6, 5, 5, 4, 5, 4, 3, 2,
         1, 0, 1, 2, 1, 2, 3, 4, 5, 6, 7, 7, 8, 7, 8, 9, 8, 7, 6, 5, 4, 3, 4,
3, 4, 4, 5, 4, 3, 4, 3,
         2, 3, 2, 3, 3, 2, 1, 1, 0, 1, 2, 1, 2, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1,
0, 1, 0, 1, 0, 0, 1, 2,
         3, 2, 1, 0, 0, 1, 2, 3, 4, 3, 2, 3, 2, 2, 1, 0, 1, 0, 0, 1, 0, 1, 2,
3, 4, 5, 6, 7, 6, 7, 7,
         6, 7, 8, 8, 9, 10, 9, 10, 9, 10, 11, 10, 9, 8, 9, 10, 11, 10, 10, 9,
10, 10, 9, 8, 7, 8, 9, 10,
         9, 8, 7, 6, 7, 6, 7, 6, 5, 4, 4, 5, 4, 3, 2, 1, 1, 0, 1, 0, 1, 0, 1,
0, 1, 2, 1, 2, 1, 0, 0,
         0, 0, 1, 0, 0, 1, 0, 1, 2, 3, 2, 3, 2, 3, 4, 3, 2, 1, 1, 2, 1, 2, 1,
2, 1, 2, 2, 1, 2, 3, 2,
         2, 2, 1, 0, 1, 2, 3, 4, 5, 4, 5, 4, 3, 2, 3, 4, 5, 5, 4, 3, 4, 3, 4,
5, 4, 3, 2, 1, 0, 1, 2,
         3, 4, 3, 2, 3, 4, 3, 2, 3, 2, 1, 0, 0, 0, 0, 0, 1, 0, 1, 2, 3, 4, 5,
4, 5, 4, 5, 6, 5, 4, 5,
         6, 5, 4, 3, 2, 1, 0, 0, 1, 0, 0, 1, 0, 1, 2, 3, 4, 3, 4, 3, 2, 1, 1,
2, 3, 2, 3, 2, 3, 2, 3,
         4, 3, 4, 4, 5, 6, 7, 6, 7, 7, 6, 5, 4, 3, 4, 3, 2, 2, 3, 4, 3, 4, 3,
2, 1, 0, 0, 1, 2, 3, 2,
         1, 2, 3, 2, 1, 0, 0, 0, 0, 1, 1, 2, 3, 2, 3, 2, 3, 4, 5, 6, 5, 5, 4,
3, 4, 5, 4, 3, 4, 5, 4,
         3, 2, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 2, 1, 0, 0, 0, 1, 0, 0, 1,
2, 3, 2, 3, 2, 3, 3, 4,
         3, 2, 3, 2, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 2, 1, 2, 1, 0, 1, 2, 1, 2,
1, 2, 3, 2, 3, 2, 1, 0,
```

```
         0, 0, 1, 2, 2, 3, 3, 2, 3, 4, 3, 2, 1, 2, 3, 3, 2, 1, 0, 1, 2, 2, 3,
4, 5, 6, 7, 7, 6, 5, 4,
         5, 4, 5, 4, 3, 2, 3, 4, 3, 2, 1, 2, 3, 4, 3, 2, 3, 2, 1, 0, 1, 2, 1,
0, 0, 0, 0, 0, 0, 1, 0,
         0, 1, 2, 2, 3, 2, 1, 0, 1, 2, 3, 4, 4, 3, 4, 3, 2, 1, 0, 0, 0, 0, 1,
0, 1, 1, 0, 1, 2, 2, 1,
         0, 0, 1, 0, 1, 2, 2, 3, 3, 4, 3, 2, 3, 2, 1, 0, 0, 1, 2, 1, 2, 3, 2,
1, 2, 3, 2, 3, 2, 1, 0,
         1, 2, 1, 2, 3, 2, 3, 2, 1, 1, 0]
```

In [11]:
```
#find pup , pdown using what we did last week

N=len(qdata) #N counts no of observations
nup = 0 #nup counts number of steps
ndown = 0 #ndown counts no of down steps
n0 = 0 #n0 counts no of times queue is 0

for i in range(1,N):
    if qdata[i] > qdata[i-1]:
        nup = nup+1
    elif qdata[i] < qdata[i-1]:
        ndown = ndown + 1

    #1st if statements complete
    if qdata[i-1] == 0:
        n0 = n0+1

p_up = nup/(N-1)
p_down = ndown/(N-1-n0)


r = 11  #maximum queue size is observed to be 11 here

#Nearneigh(pup,pdown,r)
P = Nearneigh(p_up,p_down,r)
print("P =", P )



#find equilibrium probabilities
Pi = Equilibrium(P)
print("Pi = ",Pi)

#(c)
#re-simulate queue for 1000 iterations
qdata1 = Queue(P,4,1000)


Qplot(qdata1, Pi)#plot queues for 1000 iterations

#compare the frequencies in qdata1 to the equilibrium probabilities Pi
```

```
P = [[0.58258258 0.41741742 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.        ]
 [0.50784077 0.07474181 0.41741742 0.         0.         0.
  0.         0.         0.         0.         0.         0.        ]
 [0.         0.50784077 0.07474181 0.41741742 0.         0.
  0.         0.         0.         0.         0.         0.        ]
 [0.         0.         0.50784077 0.07474181 0.41741742 0.
  0.         0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.50784077 0.07474181 0.41741742
  0.         0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.50784077 0.07474181
  0.41741742 0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.50784077
  0.07474181 0.41741742 0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.50784077 0.07474181 0.41741742 0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.50784077 0.07474181 0.41741742 0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.50784077 0.07474181 0.41741742 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.50784077 0.07474181 0.41741742]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.50784077 0.49215923]]
Pi =  [0.19676402 0.16172929 0.13293266 0.10926339 0.08980855 0.07381773
 0.06067415 0.04987084 0.04099111 0.03369246 0.02769336 0.02276243]
```



Out[11]: <matplotlib.animation.FuncAnimation at 0xb41ef232b0>

We see the System above is Ergodic as it visits every possible queue size up to the maximum queue size of 11 in the queue data provided. We use python functions to find the transition matrix for this system and then we find the array Pi consisting of the Equilibrium Probabilities.

Using the Qplot() function, I plotted the frequency of each queue size appearing in the re-simulation of the qdata that we are given and compared it to the Equilibirum probabilities Pi on the same plot. It appears that the frequencies tend towards the Equilibrium probabilities. There is a higher frequency of lower queue values than the Equilibrium probability values, similarly for the higher queue values, there is a lower frequency of those values appearing in the simulation than expected. The simulated queue is slightly quieter than the predicted queue as the low queue numbers have a higher frequency than the higher queue numbers when compared to the equilibrium probabilities. Both the simulation and the expectation probabilities show that it is more likely for the queue to have a smaller queue size closer to zero than to have a large queue size closer to the max queue size of 11. This is a system that does not become overwhelmed.

If we ran the simulation with more time intervals (i.e. 10000), we would expect the frequencies to tend closer towards the Equilibrium probabilities than in this example with 1000 time intervals.

Q.3(*)

A queue is observed 10000 time intervals with data as given in the array qdata below. This is the same as Question Question 5 of Practical 1.

(a) Construct a simple model for this queue as a Markov chain with only nearest neighbour interactions.

1. Estimate the transition probabilities.
2. What is the expected behaviour of the queue as time continues?
3. Is the system ergodic ?

(b) Suppose that the time step used is known to be 10 sec.

1. What is the average waiting time for customer service/arrival?
2. What is average number of servicings/arrivals per minute?

(c) Simulate this system and compare the actual frequencies of events with the simulated ones.

In [15]:
```
qdata = [5, 5, 5, 6, 5, 6, 6, 7, 6, 5, 6, 5, 4, 5, 6, 6, 6, 7, 8, 7, 6, 6, 6,
5, 4, 4, 5, 5, 4, 5, 6, 6, 5, 4, 3, 3, 2, 2, 1, 2, 2, 3, 3, 2, 3, 4, 3, 2, 3,
3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 4, 4, 5, 5, 5, 6, 6, 7, 6, 7, 6, 5, 4, 4, 3,
3, 2, 1, 1, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 1, 2, 1, 0, 0, 1,
1, 0, 1, 1, 0, 1, 1, 2, 3, 4, 4, 3, 2, 2, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
1, 2, 3, 3, 2, 1, 2, 1, 2, 1, 2, 3, 4, 5, 5, 6, 5, 6, 5, 4, 4, 4, 3, 2, 2, 1,
2, 2, 3, 2, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
0, 0, 1, 1, 2, 2, 2, 1, 2, 1, 1, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 1, 2, 3, 4, 3,
2, 1, 2, 3, 3, 2, 1, 0, 0, 0, 0, 1, 2, 1, 0, 1, 2, 1, 1, 1, 1, 2, 3, 4, 5, 6,
6, 7, 6, 5, 4, 5, 5, 4, 3, 2, 1, 0, 0, 1, 0, 1, 1, 2, 2, 3, 2, 1, 2, 3, 2, 3,
4, 4, 4, 5, 6, 7, 6, 5, 4, 3, 3, 2, 3, 3, 2, 1, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
0, 0, 1, 1, 2, 2, 2, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 2, 2, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 1, 2, 1, 0, 0, 1, 1, 2, 2, 3, 2, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,
2, 2, 3, 2, 1, 2, 1, 1, 1, 2, 3, 4, 4, 3, 3, 2, 1, 2, 3, 4, 5, 4, 5, 6, 5, 4,
4, 4, 4, 3, 2, 1, 1, 2, 2, 1, 1, 0, 1, 1, 2, 3, 2, 2, 1, 0, 0, 0, 0, 1, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
2, 2, 2, 2, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 2, 2, 2, 3, 2, 2, 2, 1, 1, 1, 1, 1,
1, 0, 0, 1, 2, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
1, 1, 1, 2, 1, 1, 0, 0, 1, 1, 1, 1, 1, 2, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 2, 2,
2, 2, 3, 2, 1, 1, 1, 1, 0, 0, 1, 0, 1, 2, 2, 1, 2, 1, 0, 0, 1, 1, 0, 0, 0, 0,
0, 0, 1, 2, 2, 3, 3, 2, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 2, 3, 2, 3, 3, 2, 3, 2, 3, 2, 3, 3, 3,
3, 4, 4, 4, 4, 3, 4, 4, 4, 5, 4, 5, 5, 5, 4, 3, 4, 4, 5, 6, 7, 6, 6, 7, 7, 6,
7, 7, 7, 7, 8, 8, 7, 7, 8, 9, 10, 11, 10, 11, 10, 9, 8, 8, 7, 8, 9, 8, 8, 7, 7
, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 6, 5, 5, 6, 7, 8, 7, 7, 6, 5, 4, 4, 3, 3, 4, 5
, 5, 4, 3, 2, 3, 4, 4, 4, 4, 4, 3, 3, 4, 5, 5, 6, 6, 6, 6, 6, 5, 5, 4, 4, 3, 2
, 2, 1, 0, 1, 2, 2, 3, 2, 3, 3, 2, 1, 2, 3, 2, 1, 2, 2, 3, 4, 4, 4, 5, 6, 5, 6
, 7, 6, 6, 5, 4, 3, 3, 4, 4, 5, 4, 4, 5, 4, 3, 2, 2, 1, 0, 1, 2, 2, 1, 2, 2, 2
, 2, 2, 1, 2, 2, 2, 3, 4, 5, 4, 3, 3, 2, 3, 3, 3, 4, 3, 4, 5, 6, 7, 8, 9, 10,
10, 11, 12, 12, 11, 11, 10, 11, 11, 11, 10, 9, 9, 10, 10, 9, 8, 8, 7, 7, 6, 7,
8, 8, 8, 8, 7, 8, 7, 7, 6, 5, 6, 6, 6, 7, 8, 7, 7, 6, 6, 5, 5, 6, 7, 7, 7, 6,
7, 8, 9, 8, 7, 7, 6, 6, 6, 7, 8, 9, 9, 10, 11, 11, 12, 12, 13, 14, 14, 15, 16,
15, 15, 16, 15, 14, 13, 14, 15, 14, 13, 14, 15, 15, 14, 14, 13, 14, 14, 15, 14
, 15, 14, 13, 12, 12, 11, 12, 12, 13, 13, 14, 14, 15, 15, 15, 16, 17, 18, 17,
17, 18, 19, 19, 18, 17, 16, 17, 18, 18, 18, 17, 18, 17, 16, 15, 14, 13, 12, 12
, 13, 12, 12, 12, 11, 11, 10, 11, 12, 11, 12, 11, 11, 10, 10, 10, 9, 8, 9, 10,
9, 9, 8, 8, 9, 8, 8, 8, 7, 7, 8, 9, 10, 10, 11, 10, 11, 12, 12, 11, 10, 10, 11
, 11, 11, 10, 11, 11, 10, 9, 10, 11, 10, 11, 12, 12, 11, 10, 9, 8, 8, 9, 8, 7,
8, 8, 8, 9, 9, 10, 11, 11, 12, 11, 10, 10, 9, 8, 9, 10, 11, 12, 12, 13, 12, 13
, 12, 11, 10, 11, 10, 11, 10, 9, 8, 8, 7, 6, 7, 6, 6, 7, 8, 8, 7, 6, 7, 6, 6,
5, 4, 4, 3, 2, 3, 4, 3, 4, 4, 3, 4, 3, 3, 2, 2, 2, 2, 2, 3, 2, 1, 1, 2, 2, 2,
1, 0, 0, 0, 0, 1, 1, 2, 1, 0, 0, 1, 2, 2, 1, 2, 1, 2, 2, 2, 1, 2, 1, 1, 2, 1,
2, 1, 2, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 1, 1, 0, 1, 2, 3, 4, 4, 5, 5, 4, 4, 3, 3, 2, 1, 2, 1, 1, 2, 3, 2, 1, 2, 2,
1, 2, 1, 0, 1, 2, 3, 2, 3, 4, 4, 3, 4, 3, 3, 2, 2, 1, 0, 1, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 2, 2, 3, 3, 4, 3, 3, 2, 1, 1,
2, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 2,
2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 2, 1, 2, 3, 3, 2, 2, 3, 4, 5, 5, 5, 5, 4, 5,
4, 5, 4, 4, 4, 3, 4, 3, 4, 5, 4, 5, 4, 3, 2, 3, 2, 3, 4, 3, 4, 5, 6, 6, 5, 5,
5, 6, 5, 6, 6, 7, 7, 8, 9, 10, 9, 10, 11, 12, 12, 13, 13, 13, 12, 12, 12, 11,
11, 10, 10, 9, 8, 7, 6, 6, 7, 7, 8, 8, 9, 10, 9, 8, 9, 10, 10, 10, 10, 10, 11,
11, 11, 11, 12, 13, 13, 12, 12, 11, 12, 11, 10, 9, 8, 7, 7, 8, 7, 8, 7, 7, 7,
6, 6, 6, 6, 5, 6, 5, 6, 5, 6, 7, 6, 5, 6, 5, 5, 4, 5, 5, 6, 6, 7, 6, 5, 4, 4,
```

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

```
1, 1, 1, 0, 1, 2, 3, 3, 4, 5, 6, 5, 5, 5, 4, 5, 4, 5, 4, 5, 4, 3, 3, 3, 2, 1,
1, 2, 2, 3, 4, 5, 6, 7, 7, 6, 7, 6, 7, 8, 8, 9, 10, 11, 12, 13, 14, 14, 14, 15
, 16, 16, 15, 15, 16, 17, 18, 17, 16, 15, 15, 14, 15, 14, 13, 14, 15, 14, 13,
13, 12, 12, 13, 12, 13, 13, 13, 13, 12, 13, 14, 13, 13, 13, 13, 13, 12, 11, 11
, 10, 9, 9, 10, 9, 10, 9, 9, 10, 9, 9, 10, 10, 11, 10, 9, 9, 10, 10, 10, 10, 1
0, 9, 8, 8, 7, 7, 8, 7, 8, 8, 7, 7, 7, 8, 7, 7, 7, 6, 5, 4, 3, 4, 5, 5, 4, 3,
2, 1, 1, 1, 2, 3, 2, 2, 3, 2, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
2, 2, 2, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 2, 1, 0, 0,
0, 1, 2, 2, 2, 2, 3, 2, 1, 0, 0, 0, 0, 1, 2, 1, 2, 2, 2, 3, 4, 5, 6, 5, 6, 5,
4, 5, 6, 6, 5, 5, 5, 4, 5, 4, 4, 4, 3, 4, 3, 2, 1, 0, 0, 1, 2, 3, 4, 4, 4, 5,
5, 5, 5, 4, 5, 4, 5, 6, 7, 6, 5, 5, 6, 5, 6, 5, 5, 6, 7, 7, 7, 8, 9, 9, 10, 10
, 9, 9, 8, 7, 8, 9, 10, 9, 9, 9, 9, 10, 10, 10, 9, 8, 8, 9, 8, 7, 8, 9, 9, 10,
10, 11, 12, 11, 10, 11, 10, 9, 9, 10, 10, 10, 10, 10, 10, 10, 9, 9, 8, 9, 9, 1
0, 11, 12, 12, 12, 12, 12, 13, 12, 12, 11, 10, 9, 10, 10, 10, 9, 10, 9, 9, 9,
9, 8, 7, 6, 7, 7, 6, 6, 7, 8, 9, 8, 8, 9, 9, 10, 9, 10, 9, 8, 8, 8, 9, 10, 11,
10, 11, 12, 13, 14, 14, 14, 13, 13, 12, 12, 11, 12, 13, 14, 13, 13, 12, 13, 13
, 12, 12, 12, 13, 13, 13, 13, 12, 13, 12, 13, 14, 15, 15, 15, 16, 16, 16, 16,
16, 16, 16, 15, 15, 16, 15, 16, 15, 15, 16, 15, 14, 15, 14, 14, 13, 12, 13, 14
, 15, 16, 16, 16, 16, 15, 15, 14, 14, 15, 15, 16, 16, 16, 16, 17, 18, 19, 19,
18, 19, 18, 19, 20, 19, 18, 19, 20, 20, 19, 19, 19, 20, 20, 19, 18, 18, 19, 18
, 17, 16, 15, 15, 15, 14, 14, 14, 14, 14, 14, 13, 12, 12, 13, 13, 14, 15, 15,
16, 17, 18, 18, 17, 18, 17, 17, 17, 16, 15, 14, 14, 15, 14, 13, 14, 14, 15, 16
, 17, 16, 17, 18, 19, 18, 19, 19, 20, 19, 18, 19, 18, 18, 17, 17, 18, 19, 20,
19, 18, 19, 20, 19, 20, 20, 20, 20, 19, 18, 19, 19, 19, 18, 18, 18, 18, 19, 18
, 17, 16, 15, 14, 15, 16, 15, 14, 14, 14, 13, 12, 11, 10, 11, 10, 9, 10, 9, 9,
8, 8, 7, 7, 7, 6, 5, 6, 7, 7, 8, 7, 8, 8, 9, 10, 11, 10, 11, 12, 11, 10, 10, 9
, 8, 7, 8, 8, 7, 7, 7, 8, 8, 8, 7, 8, 7, 7, 6, 6, 7, 8, 8, 8, 7, 8, 7, 8, 7, 6
, 6, 7, 6, 6, 5, 4, 5, 6, 7, 7, 8, 7, 6, 5, 4, 3, 4, 3, 2, 3, 2, 3, 2, 1, 0, 0
, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 1, 0, 0, 0, 1, 2, 1, 0, 0, 0, 1, 2, 3, 4, 3, 2, 2, 2, 2, 3, 4, 5
, 5, 6, 5, 6, 5, 6, 5, 5, 6, 6, 5, 4, 4, 4, 4, 3, 3, 2, 2, 2, 1, 0, 0, 0, 1, 2
, 3, 3, 2, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 2
, 3, 4, 3, 2, 1, 2, 2, 1, 2, 2, 3, 4, 3, 4, 3, 2, 1, 1, 0, 1, 2, 3, 4, 4, 3, 4
, 5, 6, 6, 6, 5, 5, 6, 5, 4, 3, 2, 2, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 2, 1, 1, 2, 3, 3, 2, 3, 3, 3, 4, 3, 3
, 2, 2, 1, 0, 1, 2, 2, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 2, 1, 1, 0, 1, 1, 0, 1, 0
, 1, 0, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 2, 2, 3, 4, 3, 4, 4, 5, 6, 5, 4, 3, 2, 2
, 1, 1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 1, 0, 0, 0, 1, 1, 2, 3, 4
, 4, 3, 4, 4, 3, 4, 3, 4, 3, 3, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0
, 1, 0, 1, 1, 1, 1, 2, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1, 2, 1, 2, 2
, 2, 3, 2, 3, 2, 1, 2, 1, 1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0
, 1, 2, 1, 1, 2, 2, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 2, 3, 2, 2, 1, 2, 2, 3, 2
, 2, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 5, 6, 6, 7, 6, 5, 5, 5, 6, 5
, 4, 4, 3, 2, 3, 2, 1, 0, 0, 1, 2, 1, 2, 1, 1, 1, 1, 2, 3, 4, 4, 3, 3, 4, 3, 4
, 3, 4, 4, 3, 2, 3, 3, 4, 4, 3, 2, 3, 3, 2, 2, 1, 2, 2, 1, 0, 1, 1, 1, 0, 0, 0
, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 2, 2, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0
, 1, 1, 2, 2, 1, 2, 1, 0, 0, 0, 0, 1, 0, 1, 2, 3, 2, 3, 4, 5, 5, 6, 6, 6, 5, 6
, 5, 5, 5, 4, 3, 4, 3, 2, 2, 3, 4, 3, 2, 1, 2, 3, 2, 1, 2, 1, 1, 1, 0, 0, 0, 1
, 2, 1, 2, 3, 2, 2, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0
, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 2, 1, 1, 0, 0, 0, 0
, 0, 1, 1, 1, 0, 0, 0, 1, 2, 2, 1, 1, 1, 0, 1, 2, 2, 3, 3, 4, 3, 4, 4, 3, 4, 3
, 4, 3, 4, 5, 5, 4, 3, 4, 3, 3, 3, 3, 3, 4, 4, 3, 4, 3, 3, 3, 2, 1, 2, 3, 2, 1
, 2, 2, 3, 4, 5, 5, 6, 7, 7, 6, 6, 5, 4, 3, 4, 4, 4, 5, 4, 3, 2, 1, 2, 2, 3, 2
, 1, 0, 0, 0, 1, 2, 1, 1, 2, 3, 3, 2, 2, 3, 2, 1, 2, 3, 4, 4, 3, 4, 4, 4, 3, 3
, 2, 1, 2, 1, 2, 3, 4, 5, 5, 5, 6, 6, 7, 8, 8, 8, 9, 8, 9, 8, 7, 8, 7, 7, 6, 6
, 6, 7, 6, 6, 5, 4, 5, 5, 5, 6, 6, 6, 6, 7, 6, 5, 6, 5, 6, 6, 6, 5, 5, 5, 4, 3
, 3, 2, 3, 2, 1, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 1, 2, 1, 2, 2, 1, 0, 1, 0, 0
```

```
, 0, 1, 0, 1, 1, 0, 0, 0, 1, 2, 2, 2, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1
, 0, 1, 1, 1, 1, 2, 1, 1, 2, 3, 2, 3, 3, 4, 4, 5, 4, 3, 2, 1, 2, 1, 2, 2, 1, 0
, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 2, 1, 0, 0, 1, 1, 0, 0
, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 2
, 1, 1, 2, 2, 3, 2, 2, 3, 4, 4, 4, 3, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 1, 1, 0, 1, 0, 1, 2, 1, 2, 1, 1, 0, 0, 0, 0, 1, 0, 1, 2, 2, 1, 1, 0, 0
, 1, 1, 0, 1, 1, 2, 3, 4, 5, 6, 6, 5, 4, 4, 3, 3, 2, 3, 2, 2, 2, 2, 2, 2, 1, 0
, 1, 0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 2, 2, 1, 2, 2, 3, 2, 3, 4, 4, 5, 5, 4, 5, 6
, 5, 5, 5, 5, 6, 5, 5, 6, 6, 7, 8, 8, 9, 8, 9, 9, 8, 7, 7, 6, 6, 5, 4, 4, 3, 3
, 2, 2, 1, 1, 0, 0, 1, 0, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 3, 4, 3, 3, 2, 2, 3, 4
, 4, 3, 2, 3, 4, 4, 3, 4, 5, 5, 5, 5, 4, 3, 3, 4, 3, 3, 4, 5, 4, 4, 3, 4, 3, 4
, 3, 4, 5, 5, 4, 5, 6, 7, 6, 5, 4, 5, 6, 5, 4, 4, 4, 4, 5, 4, 3, 4, 3, 2, 1, 0
, 0, 0, 1, 1, 2, 1, 2, 2, 2, 3, 3, 4, 5, 5, 5, 6, 5, 4, 3, 2, 3, 3, 4, 3, 3, 2
, 2, 1, 2, 2, 1, 2, 3, 3, 3, 3, 4, 5, 5, 4, 5, 5, 4, 3, 3, 2, 2, 1, 2, 1, 2, 3
, 4, 4, 4, 5, 4, 4, 4, 3, 3, 4, 4, 3, 4, 4, 5, 6, 5, 5, 6, 7, 7, 7, 7, 6, 6, 5
, 4, 4, 3, 3, 2, 3, 2, 1, 1, 2, 3, 2, 1, 1, 1, 1, 2, 2, 2, 1, 0, 0, 0, 1, 0, 0
, 1, 0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 3, 3, 4, 3, 3, 4, 5, 5, 6, 6, 7, 6, 6, 7
, 7, 6, 5, 6, 7, 8, 8, 9, 9, 9, 10, 9, 8, 7, 7, 7, 7, 6, 5, 4, 4, 5, 4, 4, 3,
2, 1, 2, 3, 2, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
1, 1, 2, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 0, 0, 1, 2, 1, 2, 2, 2, 2, 2, 3, 2, 3,
4, 3, 4, 5, 4, 4, 5, 6, 6, 7, 7, 6, 7, 6, 6, 5, 5, 6, 6, 7, 6, 5, 5, 4, 5, 5,
4, 3, 2, 2, 2, 2, 3, 3, 3, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2,
2, 2, 3, 4, 4, 5, 4, 4, 5, 5, 4, 4, 5, 4, 4, 3, 2, 1, 0, 0, 1, 0, 0, 1, 1, 2,
2, 2, 3, 2, 3, 3, 2, 1, 2, 3, 3, 3, 3, 2, 2, 2, 3, 4, 3, 3, 4, 5, 6, 7, 6, 6,
7, 6, 5, 4, 3, 4, 3, 2, 3, 3, 3, 2, 3, 2, 2, 2, 1, 0, 1, 1, 2, 2, 1, 2, 3, 2,
2, 1, 0, 0, 0, 1, 2, 3, 3, 2, 2, 2, 3, 4, 5, 5, 4, 5, 4, 3, 4, 5, 4, 3, 3, 4,
3, 2, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 1, 0, 0, 1, 2, 3,
3, 3, 2, 2, 3, 4, 3, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 4, 4, 5, 4, 3, 2, 3, 3,
4, 3, 2, 1, 1, 1, 2, 3, 4, 5, 5, 5, 6, 5, 5, 4, 5, 4, 5, 5, 4, 4, 4, 5, 5, 4,
3, 3, 4, 5, 5, 4, 4, 4, 3, 2, 3, 3, 3, 2, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 2, 3,
2, 1, 0, 1, 1, 2, 3, 4, 3, 4, 4, 3, 2, 1, 0, 0, 0, 1, 0, 1, 2, 1, 2, 2, 3, 2,
1, 1, 2, 1, 2, 3, 4, 4, 5, 6, 5, 5, 5, 4, 3, 2, 1, 2, 3, 2, 2, 2, 1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 1, 0, 1, 2, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
2, 1, 1, 2, 3, 2, 2, 1, 1, 2, 1, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 2, 1, 2, 1, 0, 0, 1, 1, 2, 1, 2, 1, 2, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 2, 1, 0, 1, 1,
2, 3, 4, 3, 3, 2, 3, 2, 3, 3, 3, 2, 1, 0, 0, 0, 1, 2, 1, 2, 3, 2, 3, 3, 4, 4,
5, 6, 7, 7, 7, 7, 7, 8, 7, 8, 7, 7, 6, 7, 6, 5, 6, 5, 5, 4, 4, 4, 3, 2, 2, 3,
3, 2, 3, 2, 2, 3, 4, 4, 4, 5, 6, 7, 7, 6, 6, 5, 4, 5, 4, 3, 4, 4, 5, 6, 6, 6,
7, 7, 7, 6, 5, 6, 6, 7, 7, 8, 7, 8, 8, 9, 10, 10, 10, 9, 9, 9, 8, 7, 7, 6, 7,
6, 5, 6, 7, 7, 7, 8, 9, 10, 10, 11, 11, 12, 11, 11, 12, 11, 11, 10, 11, 12, 11
, 10, 10, 11, 10, 10, 9, 8, 8, 7, 8, 9, 10, 9, 8, 9, 8, 8, 9, 9, 10, 10, 11, 1
0, 10, 11, 10, 10, 9, 8, 8, 9, 10, 11, 11, 12, 12, 13, 14, 14, 13, 12, 11, 12,
13, 14, 13, 12, 11, 10, 11, 10, 10, 10, 11, 11, 11, 10, 11, 10, 9, 9, 8, 9, 9,
10, 11, 12, 13, 12, 11, 12, 13, 13, 13, 12, 12, 12, 11, 12, 12, 11, 10, 9, 9,
9, 9, 9, 10, 9, 9, 9, 10, 11, 11, 12, 11, 10, 11, 10, 11, 11, 10, 9, 8, 7, 7,
8, 8, 7, 7, 6, 5, 6, 7, 8, 8, 8, 9, 8, 7, 6, 7, 6, 7, 7, 6, 5, 5, 6, 7, 8, 7,
6, 5, 5, 6, 7, 6, 7, 7, 8, 7, 6, 5, 4, 3, 2, 1, 2, 3, 4, 3, 2, 3, 4, 3, 2, 2,
2, 1, 1, 2, 3, 4, 4, 5, 4, 4, 3, 2, 2, 3, 2, 1, 0, 1, 1, 2, 1, 0, 0, 0, 0, 0,
1, 2, 1, 0, 1, 2, 1, 0, 1, 1, 2, 2, 3, 2, 3, 2, 1, 2, 3, 4, 4, 5, 6, 7, 7, 6,
6, 7, 6, 7, 8, 8, 8, 8, 9, 8, 8, 9, 9, 9, 10, 9, 9, 8, 9, 9, 10, 9, 9, 10, 9,
10, 9, 8, 8, 9, 9, 10, 9, 10, 9, 8, 9, 9, 8, 8, 9, 9, 9, 8, 7, 6, 5, 4, 4, 3,
4, 4, 3, 2, 3, 2, 3, 3, 2, 3, 2, 3, 4, 5, 5, 4, 3, 2, 1, 0, 0, 1, 2, 1, 0, 0,
0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 2, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 2,
3, 4, 4, 5, 4, 4, 4, 3, 4, 5, 5, 4, 3, 2, 3, 2, 2, 1, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 1, 0, 1, 0, 1, 2, 3, 3, 2, 2, 3, 3, 2, 1, 0, 0, 1, 2, 1, 0, 1, 0, 0, 0,
```

```
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 2, 2, 1, 2, 3, 3, 3,
2, 3, 2, 3, 3, 3, 4, 3, 3, 3, 3, 4, 4, 5, 4, 4, 3, 3, 3, 3, 2, 3, 3, 3, 4, 3,
4, 5, 4, 5, 6, 5, 6, 6, 6, 6, 6, 7, 6, 7, 6, 5, 5, 5, 4, 4, 5, 5, 4, 3, 3, 4,
4, 5, 4, 3, 3, 2, 1, 1, 1, 2, 3, 4, 5, 4, 4, 4, 3, 3, 4, 4, 4, 3, 3, 4, 3, 4,
5, 4, 4, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
2, 1, 0, 1, 2, 2, 2, 2, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 2, 1, 1, 0, 0,
1, 1, 0, 0, 1, 2, 3, 4, 3, 2, 3, 4, 3, 4, 3, 3, 4, 5, 4, 5, 4, 5, 4, 3, 3, 3,
2, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 2, 2, 3,
4, 5, 4, 4, 3, 4, 4, 3, 4, 5, 6, 6, 6, 5, 5, 4, 3, 3, 3, 4, 4, 3, 2, 1, 0, 1,
2, 1, 0, 1, 2, 1, 1, 0, 0, 0, 1, 0, 0, 1, 2, 1, 2, 1, 1, 0, 0, 1, 2, 1, 1, 0,
0, 1, 0, 0, 1, 1, 2, 3, 3, 3, 2, 1, 2, 1, 2, 1, 0, 1, 2, 1, 1, 0, 1, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 0, 0, 1, 2, 2, 3, 4, 4, 3, 3, 3, 3, 4, 4, 5, 6, 5, 4, 3, 3,
3, 3, 2, 2, 1, 1, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 2, 2, 2, 1, 0, 0, 0, 0, 0, 1, 1, 0,
0, 1, 1, 1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 2, 2, 2, 1, 1, 0, 0, 0,
0, 1, 1, 0, 0, 1, 1, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 1, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 2, 1, 0, 0, 0, 1,
0, 1, 2, 3, 4, 5, 6, 7, 6, 7, 8, 7, 6, 7, 8, 9, 8, 9, 10, 9, 8, 7, 6, 7, 7, 6,
7, 6, 5, 4, 3, 3, 2, 3, 4, 3, 3, 3, 3, 4, 3, 4, 5, 4, 5, 4, 3, 2, 3, 3, 2, 2,
3, 4, 4, 3, 2, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 2, 2, 3, 4, 5, 4, 3,
2, 2, 3, 3, 4, 3, 4, 4, 3, 3, 4, 3, 3, 2, 1, 0, 1, 0, 1, 1, 2, 1, 0, 0, 0, 0,
0, 0, 1, 2, 3, 3, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 0, 1, 2, 2, 1, 2, 1, 1, 2, 1,
2, 2, 3, 4, 3, 2, 3, 3, 3, 3, 4, 5, 4, 3, 2, 1, 1, 0, 1, 0, 1, 1, 2, 2, 3, 3,
3, 2, 3, 2, 1, 2, 1, 1, 1, 0, 0, 0, 0, 1, 2, 3, 3, 2, 3, 2, 2, 2, 1, 1, 2, 1,
0, 0, 0, 0, 1, 1, 2, 3, 3, 3, 4, 3, 3, 2, 2, 3, 2, 3, 4, 3, 2, 3, 2, 1, 1, 2,
1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 2, 1, 1, 1, 2, 3, 4, 5, 4, 4, 3, 3,
3, 4, 4, 3, 4, 4, 3, 2, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 2, 2, 2, 3, 4, 3,
4, 3, 2, 1, 2, 3, 2, 1, 2, 2, 1, 1, 1, 0, 1, 2, 3, 4, 5, 5, 6, 7, 6, 5, 5, 4,
5, 4, 4, 3, 2, 3, 3, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 2, 2, 1, 0,
0, 1, 1, 0, 0, 1, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 1, 1, 1, 2, 3, 3, 3, 2,
2, 2, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 1, 2, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 2, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
0, 0, 0, 0, 1, 2, 2, 2, 1, 2, 3, 3, 2, 3, 3, 2, 3, 2, 1, 0, 1, 2, 3, 2, 2, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
1, 2, 3, 3, 4, 3, 4, 3, 2, 2, 2, 1, 2, 2, 2, 1, 2, 3, 2, 3, 2, 3, 3, 4, 5, 6,
6, 5, 6, 7, 8, 8, 7, 6, 5, 4, 3, 3, 2, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 2, 2, 1, 1, 0, 0, 0, 1, 1, 2, 1, 2, 1, 0, 1,
0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 2, 1, 1, 0, 1, 2, 1, 0, 0, 1, 2, 2, 1, 2, 1, 1, 2, 2, 1, 1, 0, 1, 2, 1, 0,
1, 0, 0, 1, 2, 1, 1, 0, 0, 0, 0, 1, 1, 2, 3, 4, 5, 5, 5, 4, 4, 4, 3, 3, 2, 2,
2, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2,
3, 2, 3, 3, 2, 1, 2, 1, 0, 1, 0, 0, 0, 0, 1, 1, 2, 1, 2, 3, 3, 2, 2, 3, 2, 2,
1, 2, 3, 4, 5, 5, 6, 5, 6, 5, 4, 3, 3, 4, 3, 4, 4, 4, 5, 6, 5, 5, 5, 4, 4, 3,
4, 3, 2, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1,
0, 1, 0, 0, 1, 2, 2, 1, 2, 3, 4, 3, 3, 4, 5, 4, 3, 2, 2, 3, 4, 3, 4, 3, 2, 2,
3, 2, 2, 3, 2, 3, 4, 5, 4, 5, 6, 5, 4, 3, 2, 1, 1, 2, 2, 3, 3, 3, 2, 2, 1, 0,
1, 0, 0, 0, 1, 1, 2, 2, 1, 0, 1, 2, 1, 0, 0, 1, 2, 1, 1, 2, 2, 1, 0, 0, 0, 0,
1, 1, 2, 3, 3, 4, 3, 3, 3, 3, 3, 3, 4, 3, 4, 5, 5, 4, 4, 3, 3, 4, 3, 2, 2, 2,
3, 2, 1, 2, 2, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
2, 2, 3, 2, 1, 2, 2, 2, 3, 2, 2, 3, 2, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 4, 3, 2, 1,
1, 1, 2, 2, 1, 2, 1, 0, 0, 0, 1, 1, 2, 3, 4, 3, 2, 2, 1, 1, 2, 2, 3, 2, 2, 1,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 2, 1, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 3, 4,
```

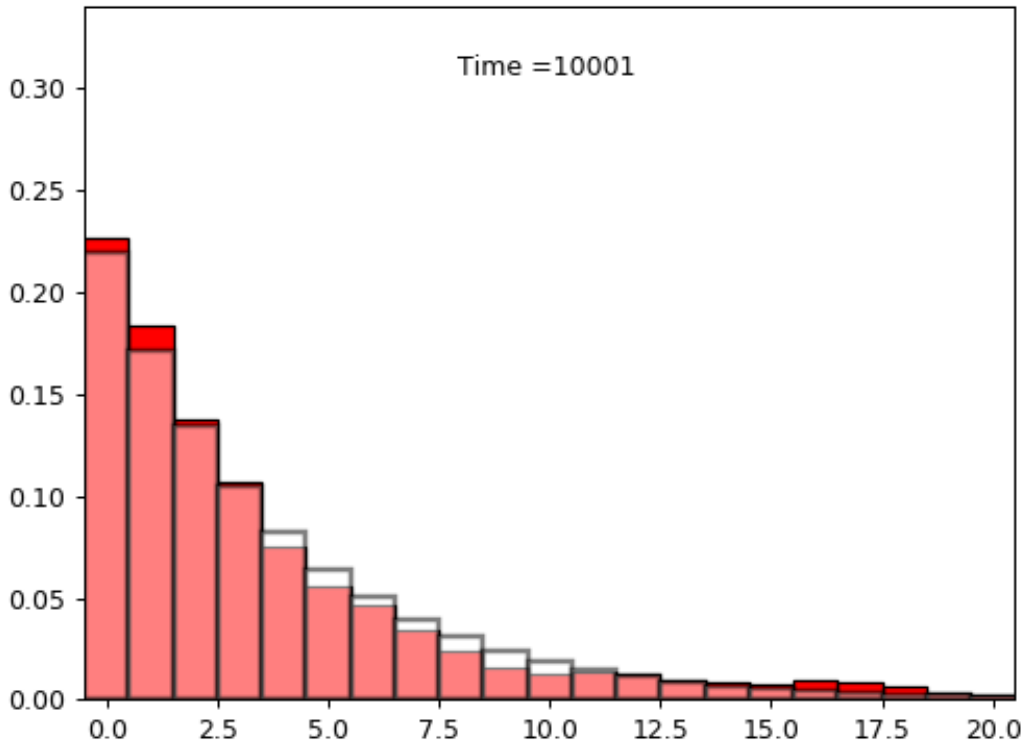
```
4, 3, 4, 5, 4, 5, 4, 3, 4, 3, 2, 1, 2, 3, 4, 5, 4, 4, 4, 4, 4, 3, 4, 5, 4, 5,
4, 4, 4, 5, 5, 4, 5, 6, 7, 8, 7, 6, 7, 6, 6, 6, 5, 4, 5, 4, 4, 5, 4, 4, 4, 4,
4, 5, 5, 4, 4, 5, 6, 5, 4, 3, 4, 4, 3, 4, 3, 4, 5, 5, 4, 5, 6, 7, 8, 8, 9, 8,
9, 9, 10, 9, 9, 10, 10, 10, 9, 9, 10, 11, 10, 9, 8, 7, 6, 5, 5, 4, 4, 3, 2, 2,
1, 2, 1, 2, 3, 4, 3, 4, 4, 3, 2, 2, 1, 2, 1, 1, 2, 1, 2, 1, 0, 0, 0, 1, 0, 0,
1, 2, 2, 1, 1, 2, 3, 2, 3, 3, 3, 3, 4, 3, 4, 5, 5, 5, 5, 4, 3, 2, 3, 2, 3, 2,
3, 4, 5, 5, 5, 5, 4, 5, 6, 5, 5, 4, 3, 3, 3, 2, 1, 2, 1, 1, 0, 1, 1, 1, 0, 1,
2, 1, 2, 2, 1, 1, 2, 1, 1, 2, 2, 2, 3, 2, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 2, 1, 2, 3, 2,
3, 4, 3, 4, 3, 4, 5, 5, 6, 6, 5, 6, 6, 5, 6, 6, 5, 5, 4, 5, 6, 7, 8, 7, 7, 8,
7, 6, 6, 5, 6, 7, 8, 8, 7, 6, 5, 4, 5, 5, 4, 4, 4, 3, 3, 2, 1, 2, 3, 3, 2, 2,
1, 1, 2, 2, 1, 1, 2, 1, 2, 3, 4, 4, 3, 2, 3, 3, 2, 3, 3, 2, 3, 2, 2, 1, 2, 1,
1, 0, 1, 2, 2, 1, 1, 2, 1, 2, 3, 2, 1, 2, 2, 2, 3, 4, 3, 2, 1, 2, 1, 0, 0, 0,
0, 0, 0, 1, 2, 3, 3, 3, 2, 3, 3, 4, 5, 5, 5, 6, 5, 4, 3, 2, 2, 1, 1, 2, 3, 2,
1, 0, 0, 0, 1, 1, 2, 2, 3, 4, 5, 6, 7, 8, 7, 8, 7, 7, 7, 8, 7, 8, 9, 9, 8, 8,
8, 8, 8, 8, 7, 7, 7, 6, 6, 7, 8, 9, 10, 9, 8, 9, 8, 7, 6, 7, 8, 7, 6, 6, 7, 8,
9, 10, 10, 9, 10, 9, 10, 10, 9, 9, 8, 7, 7, 6, 7, 6, 7, 6, 5, 6, 5, 4, 5, 5, 6
, 7, 8, 8, 7, 7, 8, 8, 7, 6, 6, 5, 5, 4, 3, 3, 4, 5, 5, 4, 4, 3, 2, 3, 4, 3, 4
, 5, 4, 5, 5, 6, 5, 6, 5, 4, 4, 5, 4, 3, 2, 2, 2, 3, 3, 3, 2, 2, 3, 3, 2, 1, 1
, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 2, 3, 2, 3, 3, 4, 4, 3, 4, 5, 5
, 6, 5, 5, 6, 7, 8, 7, 7, 6, 7, 8, 7, 6, 6, 7, 6, 6, 7, 7, 7, 8, 9, 8, 9, 8, 7
, 8, 7, 6, 6, 7, 6, 6, 6, 5, 6, 7, 7, 6, 7, 7, 7, 8, 9, 9, 9, 8, 7, 8, 9, 10,
11, 12, 12, 11, 11, 11, 12, 11, 11, 11, 11, 12, 13, 13, 14, 13, 14, 15, 14, 13
, 12, 13, 12, 12, 13, 13, 14, 13, 13, 12, 13, 13, 14, 14, 14, 15, 16, 15, 16,
17, 17, 17, 18, 18, 19, 18, 17, 17, 18, 17, 16, 16, 17, 16, 16, 17, 16, 15, 15
, 16, 15, 16, 15, 14, 13, 14, 13, 14, 15, 14, 15, 14, 15, 16, 15, 15, 16, 17,
16, 16, 16, 16, 16, 17, 17, 16, 17, 18, 17, 17, 16, 16, 17, 16, 15, 15, 14, 13, 13
, 12, 11, 11, 12, 12, 11, 11, 12, 12, 11, 11, 11, 12, 13, 14, 13, 13, 14, 13,
14, 13, 14, 15, 16, 16, 16, 16, 17, 16, 15, 14, 15, 14, 14, 13, 14, 14, 14, 13
, 14, 14, 14, 15, 16, 16, 15, 16, 17, 18, 17, 18, 17, 18, 18, 19, 19, 20, 19,
20, 19, 19, 20, 19, 18, 19, 20, 19, 18, 17, 16, 16, 15, 16, 17, 16, 15, 16, 15
, 16, 15, 14, 15, 14, 13, 14, 15, 14, 15, 14, 15, 14, 14, 15, 16, 17, 17, 16,
15, 15, 14, 14, 13, 12, 13, 13, 12, 11, 11, 11, 10, 11, 11, 11, 11, 11, 11, 12
, 13, 14, 15, 14, 13, 12, 12, 11, 10, 10, 10, 9, 10, 11, 10, 10, 11, 10, 11, 1
0, 9, 8, 8, 8, 8, 7, 7, 6, 6, 5, 5, 4, 3, 2, 2, 3, 2, 3, 4, 3, 2, 1, 2, 2, 1,
0, 0, 0, 1, 2, 2, 3, 2, 3, 3, 4, 5, 5, 5, 4, 3, 2, 1, 2, 1, 1, 0, 0, 1, 0, 0,
0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 2, 3, 4, 3, 3, 3, 4, 5, 5, 4, 3,
2, 2, 3, 2, 1, 1, 2, 1, 0, 0, 1, 2, 2, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 1, 2, 2, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 2, 2, 2, 2, 2, 3, 4, 3, 3, 3, 4,
3, 3, 4, 5, 4, 5, 5, 5, 6, 7, 6, 6, 6, 5, 6, 6, 7, 8, 9, 9, 10, 9, 8, 7, 7, 8,
9, 10, 11, 11, 11, 10, 11, 10, 11, 12, 13, 14, 13, 14, 13, 12, 11, 10, 11, 10,
9, 8, 9, 8, 7, 8, 7, 6, 7, 8, 7, 6, 6, 5, 4, 3, 2, 3, 4, 3, 3, 4, 5, 4, 4, 5,
4, 3, 3, 2, 3, 3, 2, 2, 2, 1, 2, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 2, 2, 2,
3, 2, 2, 2, 1, 2, 1, 0, 1, 2, 2, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
1, 1, 2, 2, 1, 0, 0, 1, 2, 2, 3, 4, 5, 6, 5, 5, 4, 4, 3, 3, 3, 2, 2, 2, 2, 2,
3, 3, 2, 3, 3, 3, 2, 1, 0, 0, 0, 0, 0, 0, 1, 2, 2, 3, 3, 2, 3, 4, 4, 4, 4, 4,
5, 6, 5, 5, 6, 6, 6, 7, 6, 6, 7, 8, 7, 8, 7, 8, 9, 10, 9, 8, 7, 7, 8, 7, 8, 9,
9, 8, 8, 7, 8, 9, 9, 8, 8, 7, 8, 7, 6, 5, 5, 5, 6, 7, 6, 7, 8, 8, 9, 10, 10, 9
, 8, 9, 9, 10, 10, 11, 10, 10, 9, 8, 8, 8, 8, 9, 8, 7, 7, 7, 8, 7, 6, 5, 6, 6,
6, 5, 4, 4, 4, 3, 4, 3, 2, 1, 2, 1, 1, 2, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 2,
3, 3, 2, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 2, 1, 2, 3, 2, 2, 1, 0, 1, 1,
1, 2, 2, 2, 2, 3, 2, 3, 4, 5, 5, 6, 5, 5, 6, 5, 5, 6, 7, 7, 8, 7, 6, 5, 5, 6,
5, 6, 5, 6, 5, 5, 6, 7, 8, 7, 8, 9, 8, 7, 6, 6, 7, 8, 7, 6, 6, 5, 4, 5, 4, 5,
4, 3, 2, 2, 2, 1, 1, 0, 1, 2, 2, 1, 2, 2, 3, 3, 2, 1, 0, 1, 0, 0, 0, 0, 0, 0,
1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 2, 3, 2, 3, 2, 3, 2, 2, 1, 0, 0, 0, 0,
0, 1, 1, 2, 3, 4, 3, 3, 2, 1, 2, 3, 2, 2, 2, 3, 4, 5, 5, 5, 4, 4, 5, 6, 7, 6,
5, 5, 5, 5, 4, 3, 2, 1, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 1, 2, 3,
```

```
2, 3, 2, 1, 0, 1, 1, 1, 1, 0, 1, 1, 2, 3, 3, 3, 4, 3, 2, 2, 2, 1, 0, 0, 1, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 1, 2, 2, 1, 1, 0, 0, 0, 0, 1, 2, 2,
2, 1, 0, 0, 0, 0, 1, 2, 1, 0, 1, 1, 1, 1, 2, 1, 0, 0, 0, 0, 1, 2, 1, 2, 3, 4,
4, 3, 2, 3, 4, 5, 4, 3, 4, 5, 6, 7, 7, 8, 7, 8, 9, 8, 9, 8, 9, 9, 9, 10, 9, 9,
9, 10, 11, 11, 10, 9, 9, 8, 7, 8, 8, 7, 8, 7, 6, 7, 6, 7, 7, 8, 8, 7, 7, 8, 9,
10, 9, 10, 9, 9, 8, 7, 7, 6, 5, 5, 4, 3, 3, 4, 5, 4, 3, 3, 4, 4, 5, 5, 5, 4, 5
, 6, 6, 5, 6, 6, 7, 8, 8, 9, 8, 7, 8, 7, 8, 8, 9, 10, 9, 8, 8, 7, 6, 5, 6, 6,
7, 8, 7, 7, 6, 5, 4, 3, 2, 1, 1, 1, 2, 2, 1, 0, 0, 0, 0, 0, 1, 2, 1, 0, 1, 0,
0, 0, 0, 0, 0, 1, 1, 1, 2, 3, 3, 2, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 2, 1,
2, 2, 2, 3, 2, 2, 3, 2, 3, 2, 1, 2, 3, 3, 4, 4, 4, 4, 3, 2, 3, 2, 2, 2, 3, 3,
3, 3, 2, 3, 2, 2, 1, 1, 0, 0, 0, 0, 0, 1, 2, 1, 2, 1, 2, 3, 2, 1, 2, 1, 1, 0,
0, 1, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 1, 2, 1, 1, 1, 1, 2, 3, 4, 4, 5, 5, 6,
5, 4, 4, 4, 3, 3, 3, 2, 1, 1, 0, 0, 0, 1, 0, 1, 2, 3, 2, 3, 4, 5, 6, 5, 5, 4,
3, 4, 5, 5, 4, 5, 5, 4, 4, 5, 6, 6, 5, 4, 4, 3, 3, 4, 3, 4, 3, 3, 3, 4, 3, 4,
3, 2, 2, 3, 3, 4, 3, 2, 2, 2, 2, 3, 3, 2, 3, 2, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 2, 3, 4, 3, 4, 3, 3, 2, 2, 1, 1, 2,
2, 1, 0, 0, 0, 0, 1, 1, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 2, 1, 0,
1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 4, 5, 4, 3, 2, 1, 2, 3, 3, 4, 3, 2,
1, 1, 0, 1, 2, 3, 2, 3, 2, 1, 0, 0, 0, 1, 2, 2, 3, 3, 2, 1, 0, 1, 2, 2, 3, 3,
2, 2, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 2, 2, 2, 3, 3, 3, 3, 4, 3, 3, 4, 5, 4, 5, 4, 3, 2, 2, 1, 0,
0, 0, 1, 2, 1, 0, 0, 1, 2, 1, 2, 3, 2, 2, 3, 3, 2, 1, 2, 3, 3, 4, 3, 2, 3, 2,
1, 1, 1, 1, 2, 3, 2, 2, 2, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 2, 2, 1, 2, 1, 2, 2,
1, 0, 0, 0, 1, 2, 1, 2, 3, 3, 2, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 0, 0, 0, 1, 0, 0, 0,
1, 0, 1, 1, 0, 1, 1, 2, 2, 3, 4, 3, 2, 3, 2, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 2, 3, 2, 3, 4, 5, 6, 5, 5, 5, 5, 4, 3, 4, 3, 2,
3, 2, 2, 1, 1, 2, 2, 2, 3, 4, 5, 4, 4, 5, 5, 6, 5, 4, 5, 4, 3, 3, 4, 3, 3, 4,
5, 6, 7, 7, 7, 7, 7, 6, 7, 8, 7, 8, 7, 8, 9, 9, 8, 8, 8, 8, 7, 7, 8, 8, 8, 7,
8, 7, 8, 7, 6, 5, 5, 4, 4, 5, 4, 5, 6, 5, 4, 5, 4, 3, 3, 2, 2, 3, 4, 4, 3, 3,
4, 5, 4, 4, 3, 4, 4, 3, 4, 3, 3, 2, 1, 2, 1, 2, 1, 2, 2, 3, 4, 4, 3, 2, 3, 3,
3, 2, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 2,
3, 4, 5, 6, 5, 6, 5, 6, 6, 7, 6, 6, 7, 6, 5, 5, 6, 6, 5, 6, 5, 4, 3, 2, 3, 4,
5, 4, 4, 3, 4, 3, 2, 3, 4, 3, 4, 5, 4, 3, 2, 3, 3, 4, 5, 4, 4, 4, 4, 5, 5, 5,
6, 5, 4, 5, 6, 7, 6, 5, 4, 3, 3, 4, 3, 3, 4, 4, 4, 4, 3, 2, 1, 0, 0, 1, 2, 3,
4, 3, 3, 4, 5, 6, 7, 8, 8, 9, 9, 8, 9, 8, 8, 7, 6, 5, 5, 5, 5, 4, 4, 3, 2, 2,
1, 2, 1, 0, 1, 1, 2, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 1, 0, 1, 1, 2, 2, 2, 3, 4, 4, 4, 3, 3, 3, 2, 1, 2, 1, 0, 1, 0,
1, 1, 0, 0, 0, 1, 1, 2, 3, 2, 2, 1, 0, 0, 0, 0, 1, 1, 2, 2, 3, 2, 3, 3, 4, 5,
4, 4, 5, 6, 5, 6, 7, 8, 8, 9, 9, 9, 8, 8, 7, 7, 7, 6, 5, 6, 6, 6, 5, 6, 6, 5,
4, 3, 3, 2, 3, 3, 4, 4, 4, 3, 3, 2, 3, 3, 2, 3, 2, 2, 2, 1, 1, 0, 0, 0, 0, 0,
0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
0, 0, 1, 1, 2, 2, 3, 2, 1, 0, 0, 1, 2, 1, 2, 3, 3, 4, 3, 2, 1, 2, 2, 1, 0, 0,
0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 2, 1, 1, 0, 1, 2, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 2, 3, 4, 3, 3, 4, 5, 4, 5, 5, 6, 5, 5, 4, 5, 6,
5, 5, 4, 4, 3, 2, 3, 3, 2, 1, 1, 2, 1, 0, 0, 0, 1, 1, 2, 1, 0, 0, 0, 1, 1, 0,
0, 0, 0, 1, 1, 2, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 2,
1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 2, 2, 3, 2, 1, 1, 1,
1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 2, 1, 2, 3, 2, 3, 2, 2, 1, 0, 0,
1, 2, 2, 1, 0, 1, 2, 3, 2, 1, 1, 2, 1, 2, 3, 4, 5, 4, 3, 2, 1, 1, 2, 3, 2, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 2,
2, 1, 0, 0, 1, 2, 3, 2, 2, 2, 3, 3, 2, 3, 3, 4, 5, 6, 7, 8, 7, 7, 7, 8, 7, 6,
5, 6, 5, 5, 6, 6, 7, 7, 7, 7, 8, 9, 8, 7, 7, 7, 8, 7, 6, 6, 5, 6, 6, 7, 6, 6,
7, 8, 8, 9, 9, 10, 9, 8, 9, 9, 9, 8, 8, 9, 9, 9, 10, 9, 8, 8, 7, 6, 7, 6, 6]
```

In [29]:
```
#part a)

N=len(qdata) #N counts no of observations
nup = 0 #nup counts number of steps
ndown = 0 #ndown counts no of down steps
n0 = 0 #n0 counts no of times queue is 0

for i in range(1,N):
    if qdata[i] > qdata[i-1]:
        nup = nup+1
    elif qdata[i] < qdata[i-1]:
        ndown = ndown + 1

    #1st if statements complete
    if qdata[i-1] == 0:
        n0 = n0+1

p_up = nup/(N-1)
p_down = ndown/(N-1-n0)

print("pup =",p_up)
print("pdown =",p_down)


#p_up = alpha * delta_t
delta_t = 10 #(delta t = 10 seconds here)

alpha = p_up/delta_t
print("alpha = " ,alpha)

beta = p_down/delta_t
print("beta = " ,beta)

print("average time taken for a customer to arrive = {0:0.3f} seconds".format(
1/alpha)) #1/alpha is the average time taken for 1 customer to arrive in 1 tim
e interval
print("average time taken for a customer to be serviced = {0:0.3f} seconds".fo
rmat(1/beta)) #1/beta is the average time taken for 1 customer to be serviced
 in 1 time interval

print("average number of customers arriving per minute = {0:0.3f} customers".f
ormat(p_up * 6))
print("average number of customers served per minute = {0:0.3f} customers".for
mat(p_down*6))


r = max(qdata)
print("r =", r)

P = Nearneigh(p_up,p_down,r)


Pi = Equilibrium(P)
print("Pi = ",Pi)
```

```
#re-simulate queue for 10000 iterations
qdata1 = Queue(P,5,10000)


Qplot(qdata1, Pi)#plot queues for 10000 iterations and compare with equilibriu
m probabilities
```

```
pup = 0.3068
pdown = 0.39209920736384557
alpha =  0.030680000000000002
beta =  0.03920992073638456
average time taken for a customer to arrive = 32.595 seconds
average time taken for a customer to be serviced = 25.504 seconds
average number of customers arriving per minute = 1.841 customers
average number of customers served per minute = 2.353 customers
r = 20
Pi =  [0.21881179 0.17121039 0.13396443 0.10482114 0.08201783 0.06417527
 0.05021426 0.0392904  0.03074297 0.02405499 0.01882195 0.01472733
 0.01152347 0.0090166  0.00705508 0.00552029 0.00431938 0.00337972
 0.00264448 0.00206918 0.00161904]
```



Out[29]: <matplotlib.animation.FuncAnimation at 0xb421887710>

q3 a)

1. My estimate of the transition probabilities is p_up = 0.307 and p_down = 0.392
2. Since the value of p_down is greater than the value of p_up, we expect the queue to tend towards a size of 0 since there is a greater probability of the queue decreasing than the probability of the queue increasing.
3. The system is Ergodic since it visits every queue size value from 0 to r (r is 20 here). The Equilibrium function gives us the Equilibrium probabilities of the system, it would return an error if the system was not Ergodic. The frequencies of the queue size values tends towards the Equilibrium probabilities on the plot which further shows how the system is Ergodic.

b) The average time taken for a customer arrival is 32.60 seconds. The average time taken for a customer to be serviced is 25.50 seconds.

The average number of customers arriving per minute 1.84 customers. The average number of customers served per minute is 2.35 customers.

c)The simulated frequencies of events are very close to the actual frequencies of events in this case. The simulated and actual frequencies are closer to eachother than they were in the previous question since there is a larger amount of time intervals where the queue size was measured here. Both the simulated frequencies and the equilibrium probabilities decay down towards zero. This occurs since the probability density is less than 1, alpha is less than beta so the probability of someone in the queue being serviced is greater than the probability of someone joining the queue.

Q.4 (*)

Telephone Exchange Queue.

A telephone exchange consists of N operators on N lines. Assume that the calling and servicing patterns are Poisson with parameters $\alpha_k = \alpha,\ \beta_k = k\beta$ respectively for queue size of $0 \leq k \leq N$ and for some $\alpha, \beta > 0$. The equilibrium distribution for the queue is:

$$\pi_k = \frac{\rho^k}{k!}\left(1 + \rho + \frac{\rho^2}{2!} + \ldots + \frac{\rho^N}{N!}\right)^{-1},\ \rho = \frac{\alpha}{\beta}$$

Of particular interest, in this model, is the frequency with which the exchange becomes saturated over a long period of time i.e. π_N. This should be a small number for an efficient exchange since it provides a measurement of the number of calls lost to the system.

1. A telephone exchange has 5 operators who can handle 40 calls an hour each on average. If calls arrive at the exchange on average every 40 secs, show that there is approximately a 5.5\% chance that the exchange will become saturated after a long time.
2. Simulate this system over 3 hours with 10 sec time intervals. For what proportion of your simulated events do you find all operators are busy?
3. Compare the results of your simulation to the equilibirum probabilities π_k.

Hints

Calculate α, β based on the information given for the 5 operator exchange. Be consistent with use of time units. In part (2) choose 10 secs as the time interval size and construct a transition matrix P before simulating.

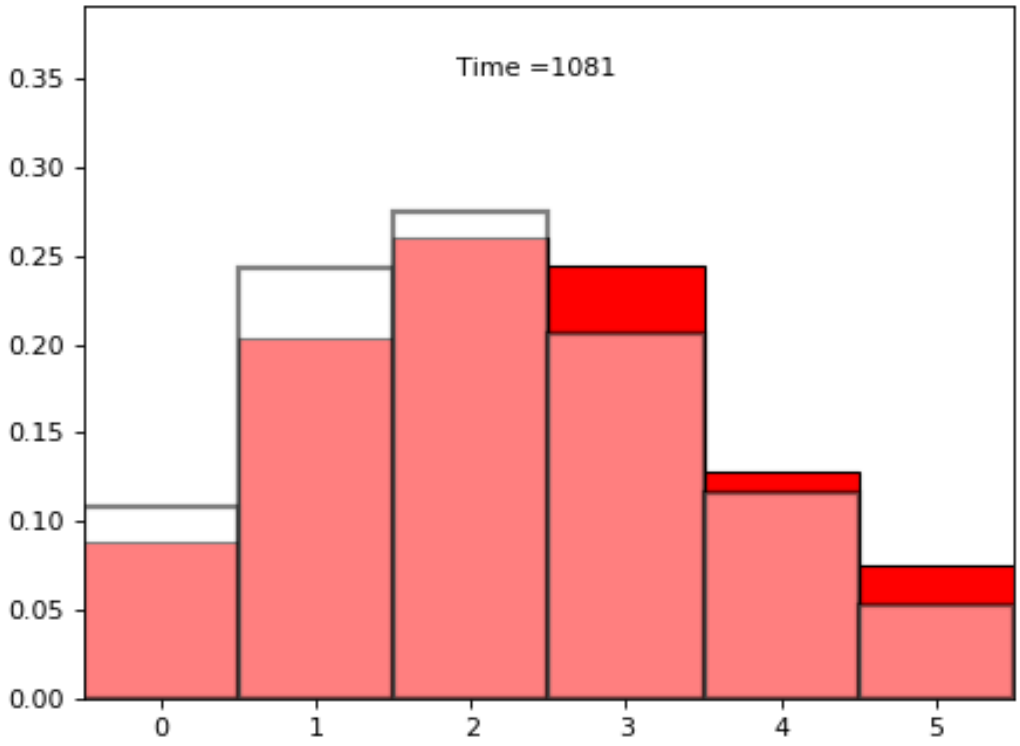
In [8]:
```
#Q4 part 1

#average mu = alpha * t

#avg/t = alpha = 1/40


alpha = 1/40 #one customer arrives every 40 seconds

beta = 1/90 #average of 40 servicings in 60 minutes -> beta = 40/(60*60)



def Nearneighbour2(pup,pdown,r):
    #creates transition matrix P for nearest neighbour model with r operators
    #on r lines with the probabilities pup for one step up transition and
    #pdown per operator for one step down transition

    P = np.zeros((r+1,r+1))
    P[0,0] = 1 - pup
    P[0,1] = pup
    P[r,r] = 1- r*pdown #pdown must be multiplied by number of lines that are
 busy
    P[r,r-1] = r*pdown

    for i in range(1,r):
        P[i,i-1] = i*pdown
        P[i,i] = 1 - i*pdown - pup
        P[i,i+1] = pup
    return P #output is transition matrix p



P = Nearneighbour2(alpha, beta, 5)

print("P = ",P)

Pi = Equilibrium(P)

print("Pi = ", Pi)
```

```
P =  [[0.975      0.025      0.         0.         0.         0.        ]
 [0.01111111 0.96388889 0.025      0.         0.         0.        ]
 [0.         0.02222222 0.95277778 0.025      0.         0.        ]
 [0.         0.         0.03333333 0.94166667 0.025      0.        ]
 [0.         0.         0.         0.04444444 0.93055556 0.025     ]
 [0.         0.         0.         0.         0.05555556 0.94444444]]
Pi =  [0.10836466 0.24382049 0.27429805 0.20572354 0.11571949 0.05207377]
```

1. We see here from the final value in Pi that there is a 5.21% chance the exchange will be saturated after a long time which agrees with the approximate value of 5.5% we are asked to show in the question.

In [12]:
```
#q4 part 2

#10 second time intervals

p_up = alpha * 10
p_down = beta *10

P = Nearneighbour2(p_up,p_down,5) #make matrix P

print("P = ", P)


Pi = Equilibrium(P)

print("Pi = ", Pi)


qdata = Queue(P,0,1080) #3hrs = 10800 seconds, then there are 1080 10 second i
ntervals here


Qplot(qdata, Pi)
```

```
P =  [[0.75       0.25       0.         0.         0.         0.        ]
 [0.11111111 0.63888889 0.25       0.         0.         0.        ]
 [0.         0.22222222 0.52777778 0.25       0.         0.        ]
 [0.         0.         0.33333333 0.41666667 0.25       0.        ]
 [0.         0.         0.         0.44444444 0.30555556 0.25      ]
 [0.         0.         0.         0.         0.55555556 0.44444444]]
Pi =  [0.10836466 0.24382049 0.27429805 0.20572354 0.11571949 0.05207377]
```



Out[12]: <matplotlib.animation.FuncAnimation at 0x3e5a9ef908>

Part 2 From the 3 hour simulation of the telephone exchange with 10 second time intervals, I found ,from the plot above, that all operators were busy over 7.7% of all simulated events. This translates to about 13.9 minutes accross the 3 hour period where all the operators were busy and the system is saturated.

Part 3. The resulting frequencies are close in similarity to the values of the equilibrium probabilities Pi. They both follow the same rough shape on the plot with low frequency values for 0 and 5 operators busy and high values in the middle around 2 or 3 operators busy. There are discrepancies between the frequencies in the simulation and the equilibrium probabilities. We see that there is a higher frequency of having 3,4 or 5 operators busy than expected and similarly there is a lower frequency of 0,1 or 2 operators being busy during this 3 hour interval. This is expected when the interval is not really long. If we ran the simulation for say 12 hours, the frequencies would better match the equilibrium probabilities. The traffic density is greater than 1, since alpha is greater than beta here meaning there is a greater chance the queue will increase in length than the chance of a customer being serviced. Hence, we expect the frequency of 2 or more operators to be busy to be quite high as we see in the plot above. The simulation is busier than what would be expected from the equilibrium probabilities, this variance is due to the traffic density being greater than 1 here and the system is not observed for long enough for the simulated frequencies to match equilibrium probabilities very closely.