Project 1 : Student number 49153 and 477789

Problem 1 : Secretary problem

In this problem we are analyzing the secretary problem. The problem consideres a company, that are going to employ a secretary among n candidates. The order of the candidates is considered to be completely random, and every candidate has a value x_n, which is interpreted as the candidates qualification. The company is running the following procedure:
- Interviewing k candidates, and turning down every one of them
- Setting $x^* = max[x_1, ..., x_k]$ to be the highest value among the first k candidates
- Pick the first subsequent candidate l with value $x_l > x^*$, or the last candidate

a)

Firstly, we define two events:
We denote $Z = 1$ to be the event that the best candidate is chosen.
We let $Y = i$ be the event that the best candidate is candidate number i.
In order to determine the probability $P(Z = 1)$ we use the law of total probability.

$$P(Z = 1) = \sum_{i=1}^{n} P(Z = 1|Y = i) \cdot P(Y = i)$$

We observe that if the best candidate is located among the k first candidates, we will never employ the best candidate. This allows us to write

$$P(Z = 1) = \sum_{i=1}^{k} P(Z = 1|Y = i) \cdot P(Y = i) + \sum_{i=k+1}^{n} P(Z = 1|Y = i) \cdot P(Y = i)$$

Where the first sum is simply zero. Since the order of candidates is random, $P(Y = i) = \frac{1}{n}$. We also identify that the only way we can choose the best candidate, given that $P(Y = i)$, is if the best candidate out of all the previous $(i-1)$ candidates, is among the k first candidates, which naturally has probability $\frac{k}{i-1}$. This argument leads to

$$P(Z = 1) = \frac{k}{n} \cdot \sum_{i=k+1}^{n} \frac{1}{i-1}$$

$$= \frac{k}{n} \cdot \sum_{i=k}^{n-1} \frac{1}{i} \approx \frac{k}{n} \left[\ln(n) - \ln(k)\right] = \frac{k}{n} \ln\left(\frac{n}{k}\right)$$

b)

Now we are going to study this probability distribution, and analyze what k a company should use, in order to make this strategy the most successfull. Firstly, lets differentiate $P(Z = 1)$ with respect to k, to find an analytically optimal value for k.

$$\frac{d}{dk}(\frac{k}{n} \cdot ln(\frac{n}{k})) = \frac{1}{n}(ln(\frac{n}{k}) - 1)$$

Then, by setting this expression equal to zero, we can obtain an optimal solution for k

$$\frac{1}{n}(ln(\frac{n}{k}) - 1) = 0$$

$$k = \frac{n}{e}$$

Thus, the theoretical best value for k, is the amount of candidates, n,

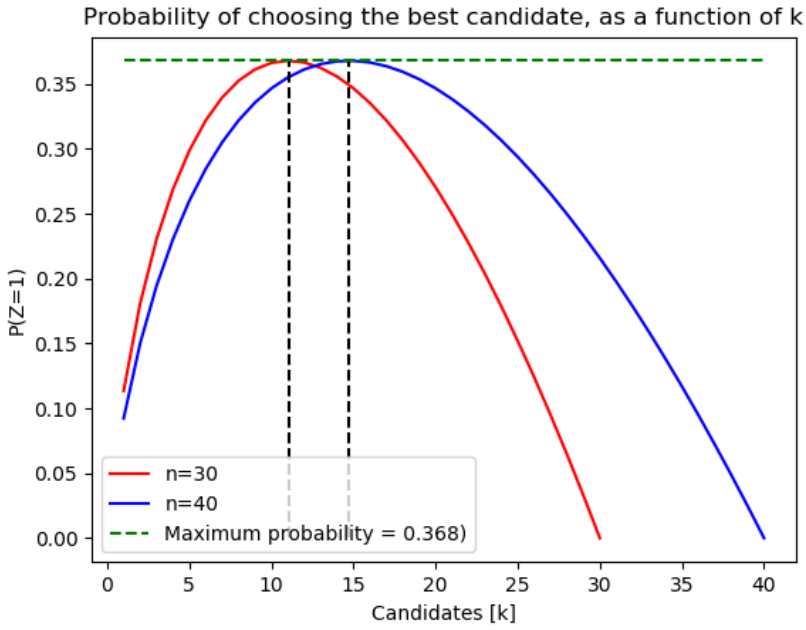

Figure 1: Plot of success probability for different values of k strategies.

divided by e. This is illustrated in our calculations in figure 1. The graph is showing the probability of choosing the best candidate, as a function of

different k's, for both $n = 30$ and $n = 40$. We have found the maximum points of the graphs, that are $k \approx 11$ for $n = 30$ and $k \approx 15$ for $n = 40$, which corresponds very well with the estimated best choice of $\frac{n}{e}$. We observe that by choosing the optimal value for k the probability of choosing the best candidate is ≈ 0.38.

c)

Now, we will simulate the secretary problem procedure for $n = 30$ and choosing $k = 11$, which is the best integer for k, with fixed $n = 30$. We drew 1000 realizations, and every candidate was randomly given a value x_n. From these 1000 realizations, we observed how often the best candidate was chosen, and how often we ended up employing the last candidate. We simulated these 1000 realizations another 1000 times, and found averages for both observations. The results is illustrated in figure 2.

As we can tell from the graph, the avergae amount of times we chose the best candidate, out of 1000 realizations, was ≈ 378. This is naturally equal to $37,8\%$, which is approximately the same fraction as we computed in the previous section. We also observe that we only get to interview all of the candidates $1,2\%$ of the time.

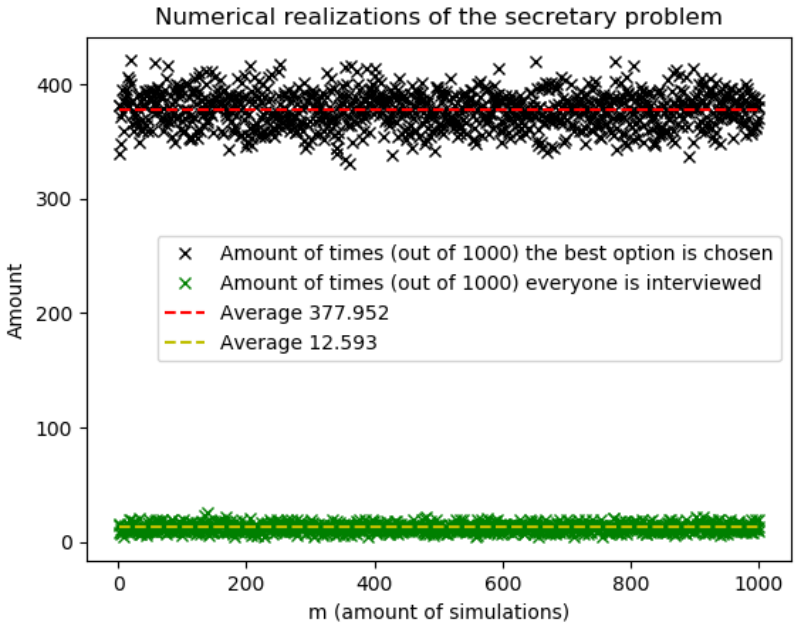

Figure 2: Simulations of the secretary problem with $n = 30$ and $k = 11$

Problem 2 : Avalanche risk

In this problem we model avalanche risk over parts of a railroad track as a Markov chain $\{X_n\}$ with state space $\Omega = \{1, 2\} = \{Low, High\}$.
The railroad is split into 50 different sections of increasing altitude. The sections are denoted by subscript n in the markov chain.

The initial probabilities are given as $P(X_1 = 1) = 0.99$
and $P(X_1 = 2) = 0.01$, and the transistion probabilities are given as $P(X_{n+1} = 1|X_n = 1) = 0.95$, $P(X_{n+1} = 2|X_n = 2) = 1$.
Thus, the transition probability matrix is given by

$$P = \begin{pmatrix} 0.95 & 0.05 \\ 0 & 1 \end{pmatrix}$$

a) Marginal probability

The marginal probability for low risk, i.e $X_n = 1$, along the railroad is given by

$$P(X_n = 1) = \sum_{k=1}^{N} P(X_n = k)P(X_n = 1|X_n = k), \tag{1}$$

where N is the total number of states.
Alternatively, as we note that state 2 is absorbing, meaning that there are no ways to exit the state once entered, the marginal probability for state 1 to occur can be calculated as

$$P(X_n = 1) = P(X_1 = 1)P(X_{n+1} = 1|X_n = 1)^{(n-1)}, \tag{2}$$

where n denotes a certain section of the railroad track.

The marginal probability as a function of road section, i.e altitude, is given in figure 3 below.

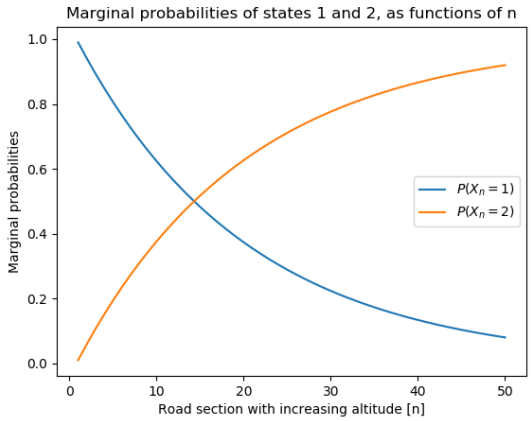

Figure 3: Marginal probabilities as a function of increasing altitude

One can observe that the probability of low risk, $P(X_n = 1)$, is greatest for lower altitudes, i.e lower n, and decrease with increasing altitude. This verifies the assertion that low risk are more likely to occur at lower altitudes, and conversely, higher risk at higher altitudes.
Note the crossing of the two marginals at $n \approx 13$. At this corresponding altitude high and low risk are equally like to occur. Above this point, the probability of high risk is higher than for low risk. Below crossing, the probability of low risk is higher than for high risk.

b) Markov chain realizations

Realizations of the Markov chain described above was drawn, and the results are plotted below for two sets of realizations, 25 and 10000. Note that we are only visualizing low risk realizations. To compare the realizations to the marginal probability calculated in section a), the realizations for each section n was divided by the total realizations in each set. It is clear from figure (4-5) that the normalized realizations tends to align the marginal probability, and that the error, i.e the difference between normalized realizations and marginal probability, decrease with increasing realizations.

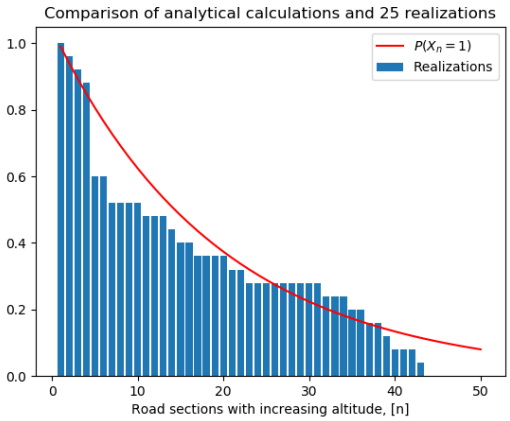

Figure 4: 25 low risk realizations of the Markov chain and $P(X_n = 1)$

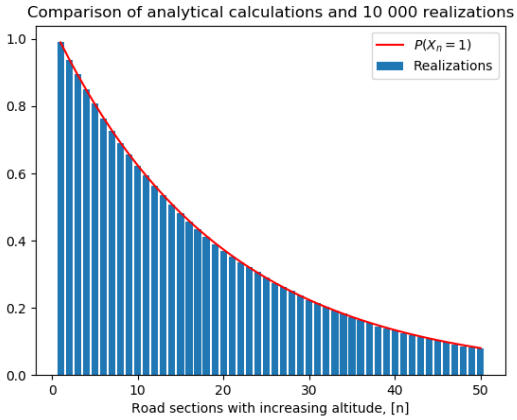

Figure 5: 10 000 low risk realizations of the Markov chain and $P(X_n = 1)$

c) Forward and backward probabilities

A sensor at location k will give perfect information at the state X_k. In this problem we fixed $k = 20$ and computed the forward- and backward probabilities

$$P(X_{l=k+1,k+2,\ldots,n} = 2|X_{k=20} = i) \tag{3}$$

$$P(X_{l=k-1,k-2,\ldots,1} = 2|X_{k=20} = i),$$

where $i = 1, 2$. Equation (3) gives the probability of being in state 2 for all sections above $k = 20$, given that $X_{k=20} = 1, 2$, and the same for all section below.
The forward probabilities are calculated as following,

$$P(X_l = 2|X_k = 1) = P_{12}^{l-k}$$

$$P(X_l = 2|X_k = 2) = P_{22}^{l-k},$$

where P_{ij}^k is element $\{i, j\}$ of the k-step transition probability matrix of the Markov chain.
The backward probabilities was calculated using Bayes rule.

$$P(X_l = 2|X_k = 1) = \frac{P(X_l = 2 \cap X_k = 1)}{P(X_k = 1)} =$$

$$\frac{P(X_l = 2)P(X_k = 1|X_l = 2)}{P(X_k = 1)} = 0,$$

As $P(X_k = 1|X_l = 2) = 0 \ \forall\{k, l : l < k\}$.

$$P(X_l = 2|X_k = 2) = \frac{P(X_l = 2 \cap X_k = 2)}{P(X_k = 2)} =$$

$$\frac{P(X_l = 2)P(X_k = 2|X_l = 2)}{P(X_k = 2)} = \frac{P(X_l = 2)}{P(X_k = 2)},$$

As $P(X_k = 2|X_l = 2) = 1 \ \forall\{k, l : l < k\}$.

Visuals of the forward- and backward-chain conditional probabilities are shown in figure 6 below.

By forward probabilities, FP, we mean the probabilities of high risk at a later road section, given a certain risk at an earlier one, namely section $k = 20$. Thus, the probability is calculated using a forward propagating Markov chain. On the other hand, the backward probability, BP, is calculated by a backwards propagating chain. That is, the probabilities of high risk at earlier sections, given a certain risk at a later one, namely $k = 20$.

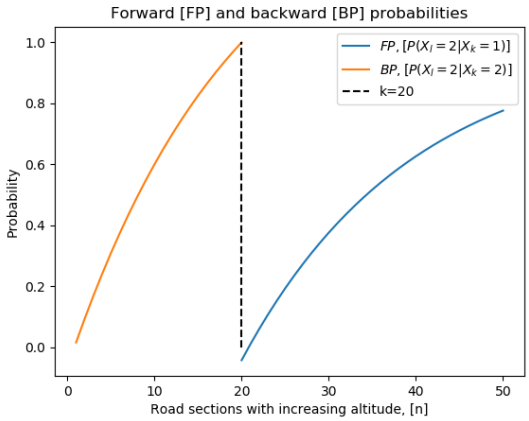

Figure 6: Conditional probabilities calculated using both forward- and backward Markov chains.

Regarding the forward probabilities, i.e the blue curve in figure 6, one can observe that the conditional probabilities increase as one moves up the road sections. That is, the probabilities for high risk increase with altitude, given that the risk was low at section $k = 20$. Observe that we only conditioned on low risk, $X_{k=20} = 1$, when calculating the forward probabilities. We chose to do so since state 2 is absorbing, revealing the constant probability $P(X_l = 2|X_{k=20} = 2) = 1 \ \forall l > k$, which is not very interesting to visualize.

When calculating the backward probabilities, we conditioned on state $X_n = 2$, i.e high risk. We chose to do so as conditioning on low risk only gave rise to zero probability, which is another result of the absorbing state $X_n = 2$. Figure 6 shows that, as we move down the road sections, the conditional probability for high risk decrease. Also note that $P(X_{19} = 2|X_{k=20} = 2) = 1$. This results from the fact that state 2 is absorbing, as is clear from the transition probability matrix for the Markov chain $\{X_N\}$.

d) Expected cost

Given costs associated with avalanches at the different risk levels, i.e states, one can calculate the expected total cost of an avalanche. Let C_1 and C_2 denote the cost associated with low and high risk areas, respectively, and C_t denote the total cost.
The total expected cost is then calculated as

$$E[C_t] = \sum_{n=1}^{50} (C_1 P(X_n = 1) + C_2 P(X_n = 2)) \tag{4}$$

Given $C_1 = 0$, i.e no cost at low risk areas, and $C_2 = 5000$, this simplifies to

$$E[C_t] = 5000 \sum_{n=1}^{50} P(X_n = 2)$$

The railroad company is given the choice to clean the tracks prior to the daily operation, at the cost of 100000 NOK, or get charged according to (4).

One assumes that the company acts rational, choosing the option with lowest associated costs.

A quick, numerical calculation resulted in $E[C_t] \approx 163000 > 100000$. Hence, the optimal decision for the company is to clean the tracks prior to its daily operations at the cost of 100000 NOK.

e) Expected information gain

By installing a sensor at location k one can learn about risk levels which in turn allows for information about whether it is optimal to clean the tracks prior to the companys daily operations or not.
One can find the best location k for the sensor by asserting the information gain at different locations along the road sections.
The information gain at location k is calculated as

$$V_k = \sum_{i=1}^{2} min\{100000, 5000 \sum_{n=1}^{50} P(X_n = 2|X_k = i)\} P(X_k = i). \qquad (5)$$

The expected information gain was calculated as (5) and plotted against road sections. The results are contained in figure 7.

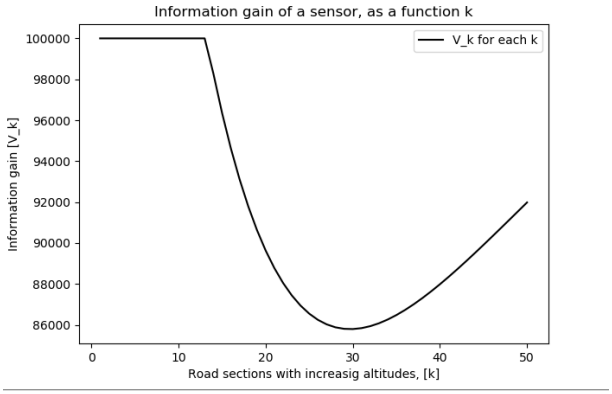

Figure 7: Expected information gain as a function of road section, i.e altitude

Now let us interpret this result. What we have actually just calculated, is the expected cost for cleaning. Obviously, we would like to minimize this as much as possible, which means the optimal location for our sensor would be at $k \approx 30$, where we would have an expected cost of ≈ 86000 NOK.

Appendix

Source code problem 1

Source code for problem 1 are contained in the screen dumps below

```python
import numpy as np
import matplotlib.pyplot as plt
import math

def prob(k,n):
    return (k/n)*math.log(n/k) #the probability distribution task 1

#marginal probabilities for states 1 and 2
def prob1(n):
    return 0.99 * 0.95**(n-1)

def prob2(n):
    return 1 - prob1(n)


|
```

Figure 8: Utilities used in solving both problem 1 and 2

```python
import numpy as np
import matplotlib.pyplot as plt
import math
import utilities as ut

#data for experiment with 30 employees:
n1 = 30
k_list1 = [] #list with k's
prob_list1 = [] #list with probabilities

#data for experiment with 40 employees:
n2 = 40
k_list2 = []
prob_list2 = []

#calculations for first experiment:
for k in range(1,n1 + 1): #iterating through all possible k's
    prob_list1.append(ut.prob(k,n1)) #calculating the probability
                                     #of choosing the best employe with this k
    k_list1.append(k) #filling the k list with k's

#calculations for second experiment:
for k in range(1, n2 + 1):
    prob_list2.append(ut.prob(k,n2))
    k_list2.append(k)

#MAKING LISTS TO ILLUSTRATE THE MAXIMUMS OF THE PLOT
max1 = n1/math.e
max2 = n2/math.e
maxprob = ut.prob(max1,n1)
maxlist = np.linspace(0,maxprob,10)
maxlist2 = 40*[maxprob]
n1list = 10*[max1]
n2list = 10*[max2]


#plotting the results:
plt.plot(k_list1,prob_list1,'r-',label="n=30")
plt.plot(k_list2,prob_list2,'b-',label="n=40")
plt.plot(n1list,maxlist,'k--')
plt.plot(n2list,maxlist,'k--')
plt.plot(k_list2,maxlist2,'g--',label='Maximum probability = 0.368)')
plt.xlabel("Candidates [k]")
plt.ylabel("P(Z=1)")
plt.legend()
plt.title('Probability of choosing the best candidate, as a function of k')
plt.show()
plt.savefig('1bplot')
```

Figure 9: Source code problem 1b)

```python
import numpy as np
import matplotlib.pyplot as plt
import math
import utilities as ut

n = 30 #amount of interviews in a process
k = 11 # teoretically and numerically determined to be the optimal value for k from b)
|
b = 1000 #simulations of a interview process

m = 1000 #simulation of the whole experiment


count_all_list= [] #list for how often everyone is being interviewed
count_best_list = [] #list for how often the best one is chosen
m_list = [] #list for number of simulations
for j in range(1,m+1): #simulating the experiment
    m_list.append(j)
    count_best = 0 #a count for how often the best one is chosen, resets for every simulation
    count_all = 0 #a count for how often every one is being interviewed, resets for every simulation
    for i in range(b): #simulating an interview process
        x = np.random.uniform(1,100,n) #list with uniformly distributed candidates
        max = np.amax(x) #the best candidate
        max_index = np.where(max) #chronological location of the best candidate
        xk_max = np.amax(x[0:11]) #the best candidate out of them who guaranteed to be denied
        x_relevant = x[11:n] #the rest of the candidates
        for element in x_relevant: #iterating through the rest of candidates
            if element > xk_max: #checking whether we should employ the candidate
                if element == max: #checking whether we have employed the best candidate
                    count_best += 1
                if element == x[-1]: #checking whether we have interviewd all candidates
                    count_all +=1
                break
    count_all_list.append(count_all) #adding the results of this simulation to lists
    count_best_list.append(count_best)

#determining the averages and making lists for plots:
average_bestrate = sum(count_best_list) / len(count_best_list)
average_allrate = sum(count_all_list) / len(count_all_list)

average_bestrate_list = [average_bestrate]*n
average_allrate_list = [average_allrate]*n

#plotting the results:
plt.plot(m_list,count_best_list,'kx',label="Amount of times (out of 1000) the best option is chosen")
plt.plot(m_list,count_all_list,'gx',label="Amount of times (out of 1000) everyone is interviewed")
plt.plot(m_list,average_bestrate_list,"c--",label='Average '+str(average_bestrate))
plt.plot(m_list,average_allrate_list,"y--",label='Average '+str(average_allrate))
plt.xlabel("n (amount of simulations)")
plt.ylabel("Amount")
plt.title('Numerical realizations of the secretary problem')
plt.legend()
plt.show()
plt.savefig('1cplot')
```

Figure 10: Source code problem 1c)

Source code problem 2

Source code for solving problem 2 are shown in the screen dumps below

```
import numpy as np
import matplotlib.pyplot as plt
import math
import utilities as ut


nlist = [] #list for n's
prob1_list = []
for i in range(1,51):
    nlist.append(i)
    prob1_list.append(ut.prob1(i))#calculating the marginal probability of state 1,
                                  #similarly for 2 later.

prob2_list = []
for i in range(1,51):
    prob2_list.append(ut.prob2(i))

plt.plot(nlist,prob1_list,label='$P(X_{n} = 1)$')
plt.plot(nlist,prob2_list,label='$P(X_{n} = 2)$')
plt.xlabel("Road section with increasing altitude [n]")
plt.ylabel("Marginal probabilities")
plt.legend()
plt.title('Marginal probabilities of states 1 and 2, as functions of n')
plt.show()
plt.savefig('2aplot')
```

Figure 11: Source code problem 2a)

```
import numpy as np
import matplotlib.pyplot as plt
import math
import utilities as ut

real = 10000 #number of realizations

countarray = 50*[0] #counter array for each realization


nlist = [] #list for n's
prob1_list = [] #list for marginal probabilites, in order to compare it to the realizations
for i in range(1,51):
    nlist.append(i)
    prob1_list.append(ut.prob1(i))


for i in range(real): #iterating through all realizations
    state = np.random.choice([1,2],p=[0.99,0.01]) #randomly generating the first state
    count = 0
    while state == 1: #running a loop as long as we are in state 1
        countarray[count] += 1
        count += 1
        if count == 50: #if we are at the end of railroads, we stop
            break
        state = np.random.choice([1,2],p=[0.95,0.05]) #creating a new state, randomly

for i in range(50):
    countarray[i] = countarray[i] / real #scaling our realizations to a value between 0 and 1

plt.plot(nlist,prob1_list,'r-',label='$P(X_{n} = 1)$')
plt.bar(nlist,countarray,label='Realizations')
plt.xlabel("Road sections with increasing altitude, [n]")
#plt.ylabel("Marginal probability, (P(X=1))")
plt.legend()
plt.title('Comparison of analytical calculations and 10 000 realizations')
plt.show()
plt.savefig('2bplot')
```

Figure 12: Source code probelm 2b)

```python
import numpy as np
import matplotlib.pyplot as plt
import math
import utilities as ut

k = 20 #location of the sensor
forward_l_list = [] #list of l's
backward_l_list = []

forwardlist = [] #vector for forward probabilities
backwardlist = [] #vector for backward probabilities

for i in range(k,51):
    forward_l_list.append(i)
    forwardlist.append(ut.prob2(i-k)) #calculating the forward probabilites for each l > k

for i in range(1,k+1):
    backward_l_list.append(i)
    backwardlist.append(ut.prob2(i)/ut.prob2(k)) #calculating the backward probabilities
                                                 #for each l < k

#making lists to mark k in the plot
yklist = np.linspace(0,1,10)
xklist = 10*[20]

plt.plot(forward_l_list,forwardlist,label='$FP, [P(X_{l} = 2 | X_{k} = l)]$')
plt.plot(backward_l_list,backwardlist,label='$BP, [P(X_{l} = 2 | X_{k} = 2)]$')
plt.plot(xklist,yklist, 'k--',label='k=20')
plt.xlabel("Road sections with increasing altitude, [n]")
plt.ylabel("Probability")
plt.legend()
plt.title('Forward [FP] and backward [BP] probabilities')
plt.show()
plt.savefig('2cforward')
```

Figure 13: Source code problem 2c)

```python
import numpy as np
import matplotlib.pyplot as plt
import math
import utilities as ut


def optimalChoice(): #function for calculating the optimal choice
    constCost = 100000
    expectedCost = 0
    for i in range(1,51):
        expectedCost += 5000*ut.prob2(i)
    return min(constCost,expectedCost)

print(optimalChoice())
```

Figure 14: Source code problem 2d)

```python
import numpy as np
import matplotlib.pyplot as plt
import math
import utilities as ut


klist = [] #list of k's
infogain = [] #list of information gain for each k
for k in range(1,51):
    klist.append(k)
    gain = 0 #variable for the gain for each k
    for i in range(1,3):
        summ = 0 # variable for the summ for each i
        if i == 1:
            for n in range((k+1),51): #if i = 1, we only calculate forward probabilites from k
                summ += 5000*ut.prob2(n-k)
            gain += min(100000,summ)*ut.prob1(k) #choosing the minimum and adding to "gain"
        if i == 2:
            for n in range(1,51):
                if n < k:
                    summ += 5000*(ut.prob2(n)/ut.prob2(k)) #if i = 2, we calculate backward probability to k
                else:
                    summ += 5000 #and 2 is an absorbing state, such that the forward probability is 1
            gain += min(100000,summ)*ut.prob2(k) #choosing the minimum, and adding to "gain"

    infogain.append(gain) #adding the total gain to the vector og gains.

plt.plot(klist,infogain,'k-',label='V_k for each k')
plt.xlabel("Road sections with increasig altitudes, [k]")
plt.ylabel('Information gain [V_k]')
plt.legend()
plt.title('Information gain of a sensor, as a function k')
plt.show()
plt.savefig('2eforward')
```

Figure 15: Source code problem 2e)