Sheet 2

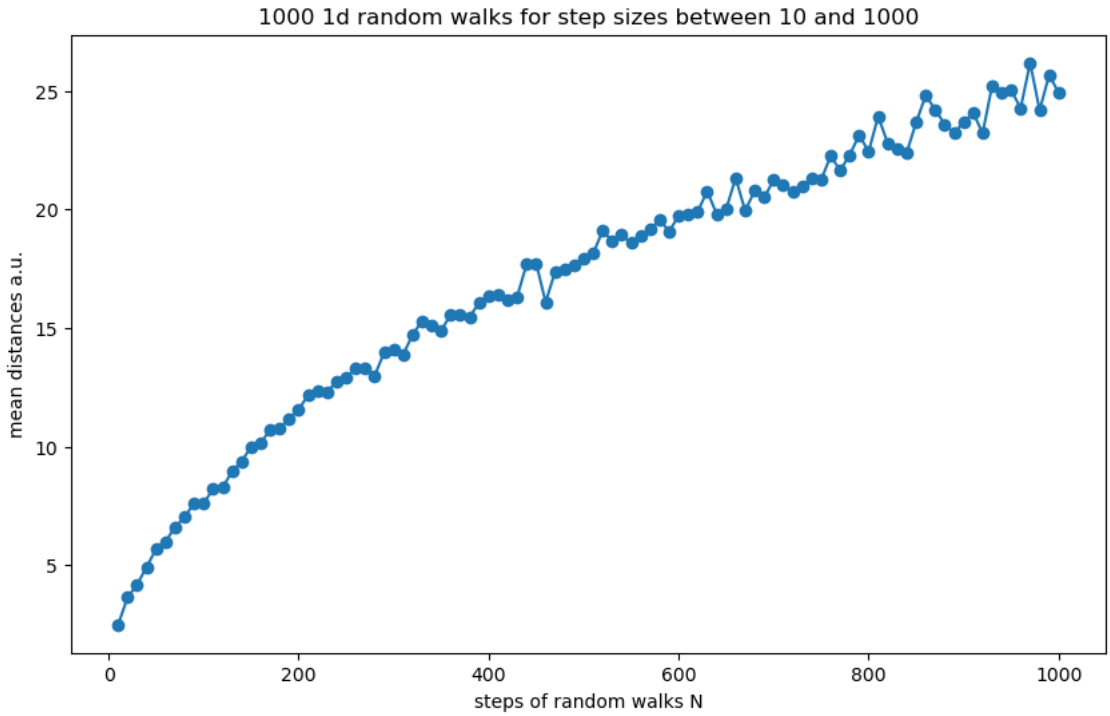
November 3, 2023

1 Sheet 2

1.1 Problem 3 a)

[1]:
```
#import of nessesary packages
import numpy as np
import matplotlib.pyplot as plt
```

[19]:
```
#number of random walks
num_rw = 1000

#lists for saving of mean distances and values of N
N_values = range(10, 1001, 10)
avg_distances = []

#going through values of N
for num_steps in N_values:
    distances = []

    #proceeding random walks and calculate the distances
    for _ in range(num_rw):

        walk = np.random.choice([-1, 1], size=num_steps).cumsum()
        distances.append(abs(walk[-1]))  #final position saved in list

    avg_distance = np.mean(distances)#calculate mean distance for each value of␣
 ↪N
    avg_distances.append(avg_distance)#save mean distance in list

#plot of data
plt.figure(figsize=(10, 6))
plt.plot(step_counts, avg_distances, marker='o', linestyle='-')
plt.xlabel('steps of random walks N')
plt.ylabel('mean distances a.u.')
plt.title('1000 1d random walks for step sizes between 10 and 1000')
plt.show()
```



1.2 Problem 3 b)

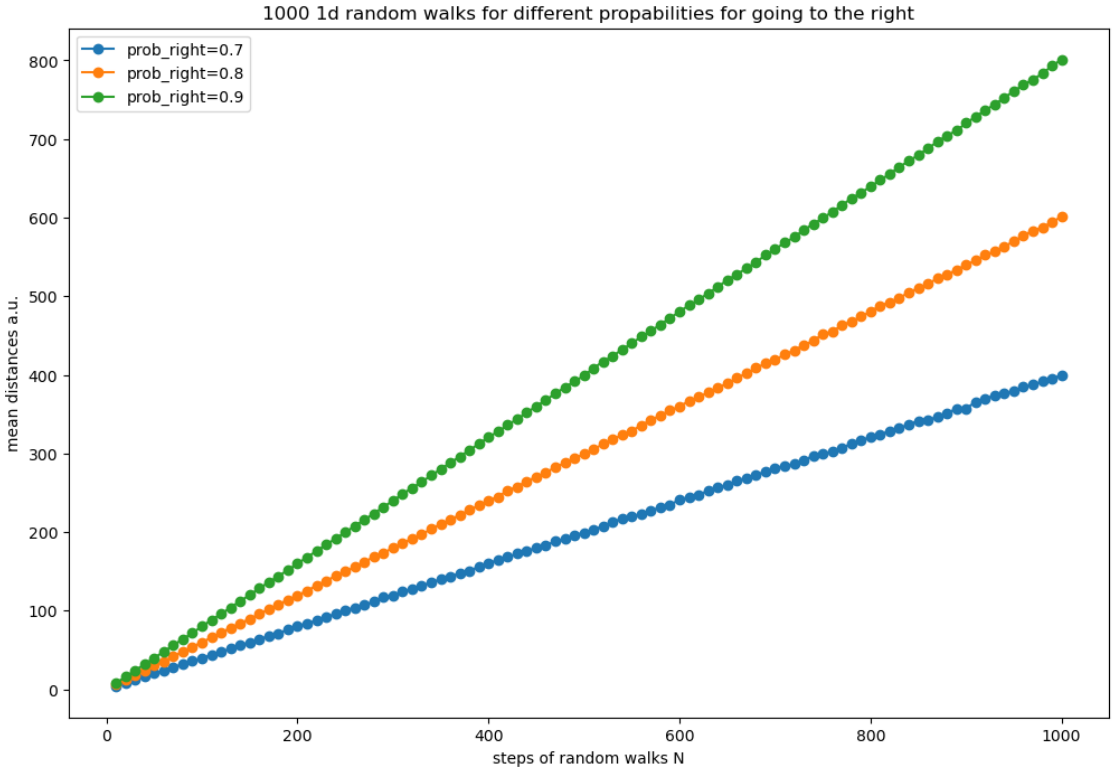
[20]:
```
#number of random walks
num_rw = 1000

#lists for saving of mean distances and values of N
N_values = range(10, 1001, 10)
avg_distances = []

#defining the propability for steps to left or right
prob_right = 0.5
prob_left = 1 - prob_right

#going through values of N
for num_steps in N_values:
    distances = []

    #proceeding random walks and calculate the distances
    for _ in range(num_rw):

        walk = np.random.choice([-1, 1], size=num_steps).cumsum()
        distances.append(abs(walk[-1]))  #final position saved in list
```

```
    avg_distance = np.mean(distances)#calculate mean distance for each value of␣
 ↪N
    avg_distances.append(avg_distance)#save mean distance in list
```

1.3 Problem 3 c)

[21]:
```
#defining of values of N
N_values = range(10, 1001, 10)

#list for values of popability for going to the right
prob_right_values = [0.7, 0.8, 0.9]

#create the figure for the plots
plt.figure(figsize=(12, 8))

#slope for each value of popability for going to the right
for prob_right in prob_right_values:
    avg_distances = []

    #proceeding for each value of N
    for num_steps in N_values:
        distances = []

        #proceeding random walks with fixed popability
        for _ in range(1000):

            prob_left = 1 - prob_right #defining of popability for going to the␣
 ↪left
            step_choices = np.random.choice([1, -1], size=num_steps,␣
 ↪p=[prob_right, prob_left])
            walk = step_choices.cumsum()
            final_distance = abs(walk[-1])
            distances.append(final_distance)

        avg_distance = np.mean(distances)
        avg_distances.append(avg_distance)

    #plot for each value of popability for going to the right
    label = f'prob_right={prob_right}'
    plt.plot(N_values, avg_distances, marker='o', linestyle='-', label=label)

#plot of data
plt.xlabel('steps of random walks N')
plt.ylabel('mean distances a.u.')
plt.title('1000 1d random walks for different propabilities for going to the␣
 ↪right')
```

```
plt.legend()
plt.show()
```



with increasing popability for going to the right the mean ditance of the end position to the start is increasing. that means with increasing popability for going to the right the data is closer to a linear trajectory. the reason for that is, that with increased popability for going to the right there are more right values then left values compared to the other propabilities and therofre the values of mean end values are higher for each value of N

Feedback: with help of the work of sheet 1 I needed around 3 hours for finishing problem 3

[ ]: