Time Complexity Analysis:

void GraphOperator::calculateDegrees( ):

Time complexity: $O(|V|)$
It traverses all the vertices and calculate the degrees.

void GraphOperator::calculateEccentricities( ):

Time complexity: $O(|V|^2 + |E|)$
It traverses all the vertices using two for loops and it also traversed all edges, which leads to $O(|V|^2 + |E|)$ time complexity.

double GraphOperator::FindAverageDegree( ):

Time complexity: $O(|V|)$
It traverses all the vertices and find the average degree.

int GraphOperator::FindHighestDegree( ):

Time complexity: $O(|V|)$
It traverses all the vertices and find the highest degree.

void GraphOperator::calculateConnected( ):

Time complexity: $O(|V| + |E|)$
Since it traverses every node and all the edges to find the connected parts, the time complexity is the same as above.

std::vector< std::vector<double> > GraphOperator::FindConnectedParameters( ):

Time complexity: $O(|V|)$
It traverses the connected vector and get all of its parameters.

double GraphOperator::FindTrianglesRatio( ):

Time complexity: $O(|V| + |E|)$
It traverses all the vertices and edges and used the algorithm calculated to find the ratio.

int GraphOperator::FindClosestNode( ):

Time complexity: $O(|V|)$

int GraphOperator::FindHighestInterest( ):

Time complexity: $O(|V|)$

std::pair<int,int> GraphOperator::FindDistanceRatio( ):

Time complexity: $O(|V|^2)$

Below is the plot of the time tooken by each functions:

In [1]:
```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from skimage import io
import cv2
from matplotlib import cm
import random
```

In [2]:
```
vertices=[]
calc_deg=[]
calc_eccen=[]
find_ave=[]
find_highest=[]
find_connect=[]
calc_connect=[]
find_parameter=[]
find_ratio=[]
find_closest=[]
find_intere=[]
find_dis_ratio=[]
```

read data for $V = 100$:

In [3]:
```
with open('record100.txt') as f:
    vertices.append(100)

    contents = f.readlines()
    val=contents[0].split('\n')
    val=val[0].split(' ')
    val = [float(i) for i in val]
    print(val)

    calc_deg.append(val[0])
    calc_eccen.append(val[1])
    find_ave.append(val[2])
    find_highest.append(val[3])
    find_connect.append(val[4])
    calc_connect.append(val[5])
    find_parameter.append(val[6])
    find_ratio.append(val[7])
    find_closest.append(val[8])
    find_intere.append(val[9])
    find_dis_ratio.append(val[10])
```

```
[0.02175, 52.7283, 0.002292, 0.080042, 0.000833, 0.069459, 0.069167, 0.113541, 0.012666, 0.0023
75, 211.342]
```

read data for $V = 250$:

In [4]:
```
with open('record250.txt') as f:
    vertices.append(250)

    contents = f.readlines()
    val=contents[0].split('\n')
    val=val[0].split(' ')
    val = [float(i) for i in val]
    print(val)

    calc_deg.append(val[0])
    calc_eccen.append(val[1])
    find_ave.append(val[2])
    find_highest.append(val[3])
    find_connect.append(val[4])
    calc_connect.append(val[5])
    find_parameter.append(val[6])
    find_ratio.append(val[7])
    find_closest.append(val[8])
    find_intere.append(val[9])
    find_dis_ratio.append(val[10])
```

```
[0.047333, 778.286, 0.004042, 0.407334, 0.000875, 0.176916, 0.161542, 0.264458, 0.030333, 0.005
25, 3158.3]
```

read data for $V = 500$:

In [5]:
```
with open('record500.txt') as f:
    vertices.append(500)

    contents = f.readlines()
    val=contents[0].split('\n')
    val=val[0].split(' ')
    val = [float(i) for i in val]
    print(val)

    calc_deg.append(val[0])
    calc_eccen.append(val[1])
    find_ave.append(val[2])
    find_highest.append(val[3])
    find_connect.append(val[4])
    calc_connect.append(val[5])
    find_parameter.append(val[6])
    find_ratio.append(val[7])
    find_closest.append(val[8])
    find_intere.append(val[9])
    find_dis_ratio.append(val[10])
```

```
[0.086667, 6050.13, 0.006625, 1.4515, 0.000792, 0.342, 0.307666, 0.536, 0.053333, 0.020083, 242
13.6]
```

read data for $V = 750$

In [6]:
```
with open('record750.txt') as f:
    vertices.append(750)

    contents = f.readlines()
    val=contents[0].split('\n')
    val=val[0].split(' ')
    val = [float(i) for i in val]
    print(val)

    calc_deg.append(val[0])
    calc_eccen.append(val[1])
    find_ave.append(val[2])
    find_highest.append(val[3])
    find_connect.append(val[4])
    calc_connect.append(val[5])
    find_parameter.append(val[6])
    find_ratio.append(val[7])
    find_closest.append(val[8])
    find_intere.append(val[9])
    find_dis_ratio.append(val[10])
```

```
[0.1395, 20030.4, 0.008417, 3.13204, 0.000917, 0.526708, 0.456875, 0.830125, 0.084583, 0.037828
, 80256.2]
```

read data for $V = 1000$:

In [7]:
```
with open('record1000.txt') as f:
    vertices.append(1000)

    contents = f.readlines()
    val=contents[0].split('\n')
    val=val[0].split(' ')
    val = [float(i) for i in val]
    print(val)

    calc_deg.append(val[0])
    calc_eccen.append(val[1])
    find_ave.append(val[2])
    find_highest.append(val[3])
    find_connect.append(val[4])
    calc_connect.append(val[5])
    find_parameter.append(val[6])
    find_ratio.append(val[7])
    find_closest.append(val[8])
    find_intere.append(val[9])
    find_dis_ratio.append(val[10])
```

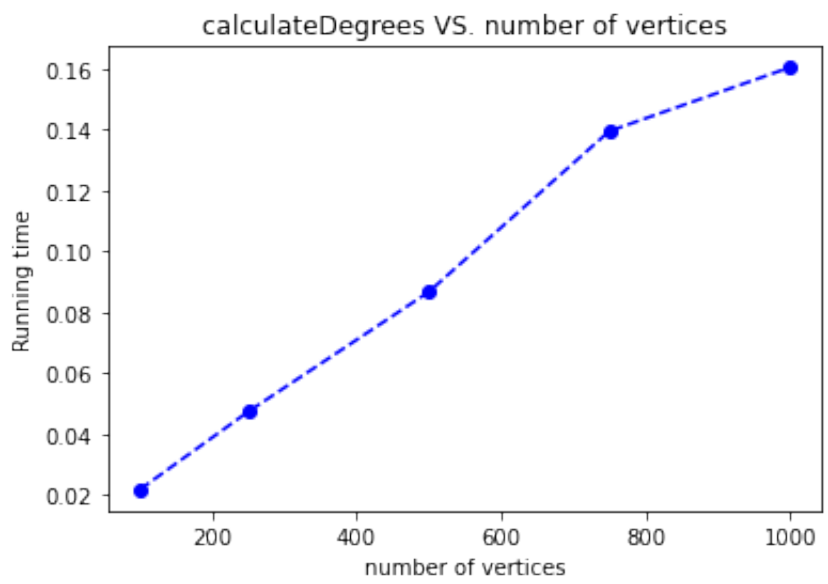
```
[0.160458, 47131.6, 0.010917, 5.51621, 0.000875, 1.2141, 1.01861, 1.08996, 0.097792, 0.061167,
188399.0]
```

Then,let's plot the relationship between running time and number of vertices:

1. calculateDegrees( ):

In [8]:
```
plt.figure()
plt.title('calculateDegrees VS. number of vertices')
plt.plot(vertices,calc_deg,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

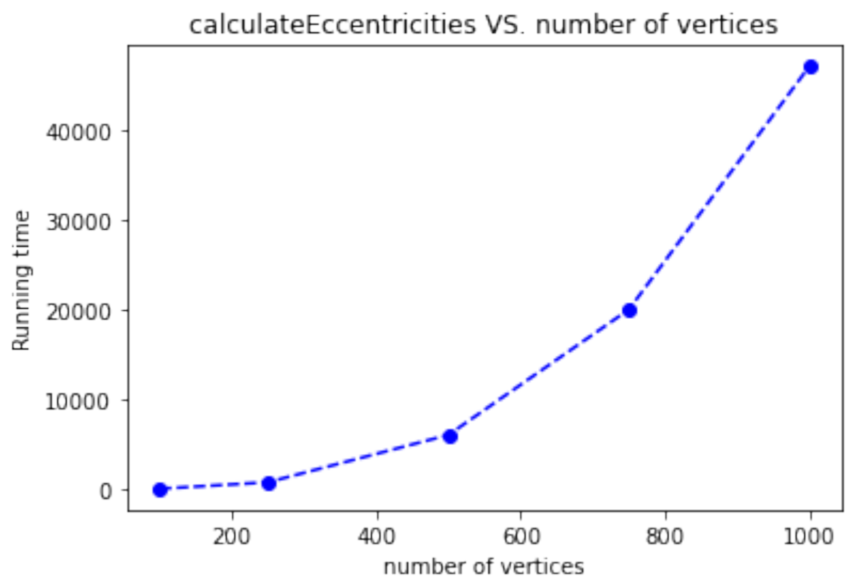
Out[8]: Text(0, 0.5, 'Running time')



2. calculateEccentricities( ):

In [9]:
```
plt.figure()
plt.title('calculateEccentricities VS. number of vertices')
plt.plot(vertices,calc_eccen,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

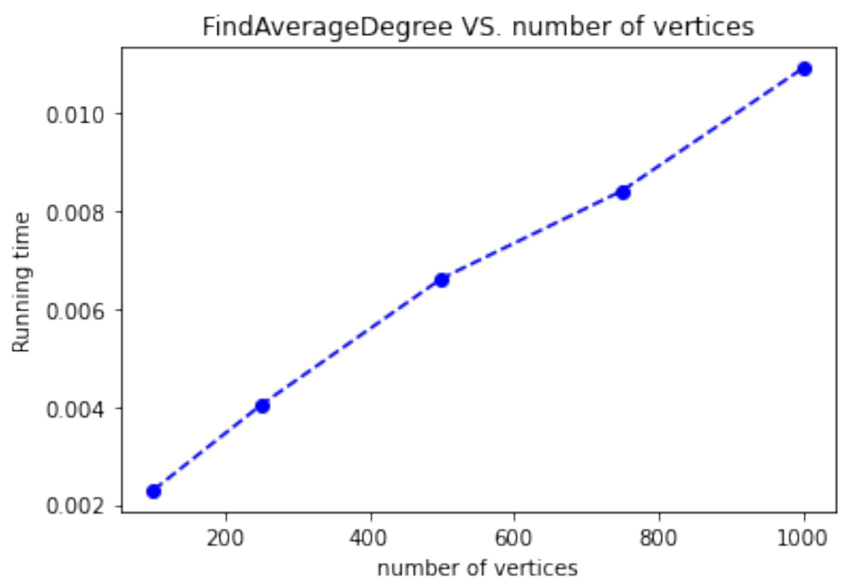
Out[9]: Text(0, 0.5, 'Running time')



3. FindAverageDegree( ):

In [10]:
```
plt.figure()
plt.title('FindAverageDegree VS. number of vertices')
plt.plot(vertices,find_ave,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

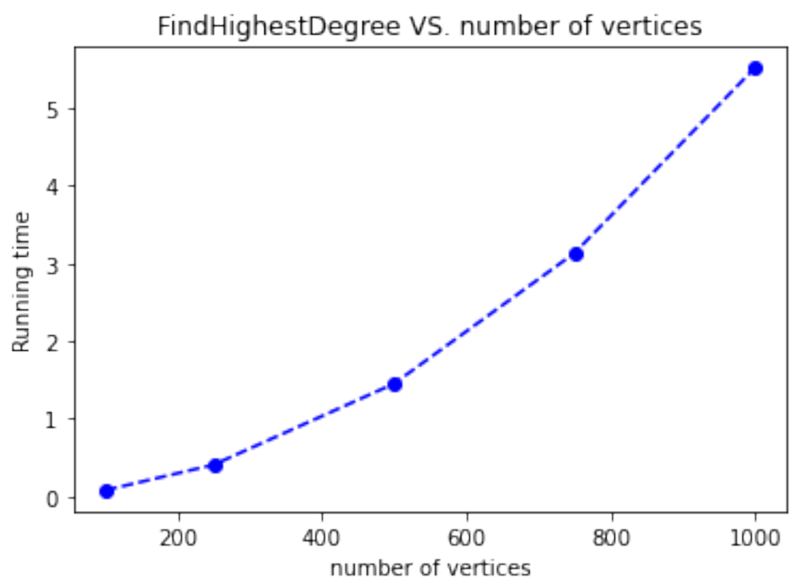
Out[10]: Text(0, 0.5, 'Running time')



4. FindHighestDegree( ):

In [11]:
```
plt.figure()
plt.title('FindHighestDegree VS. number of vertices')
plt.plot(vertices,find_highest,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

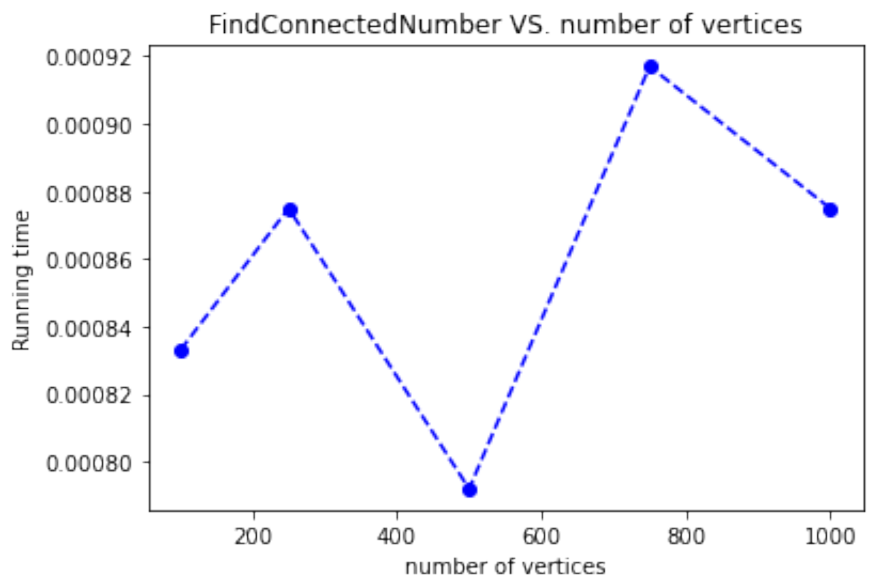
Out[11]: Text(0, 0.5, 'Running time')



5. FindConnectedNumber( ):

In [12]:
```
plt.figure()
plt.title('FindConnectedNumber VS. number of vertices')
plt.plot(vertices,find_connect,'bo--')
# a, b = np.polyfit(vertices, find_connect, 1)
# plt.plot(vertices, a*vertices+b)
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

Out[12]: `Text(0, 0.5, 'Running time')`

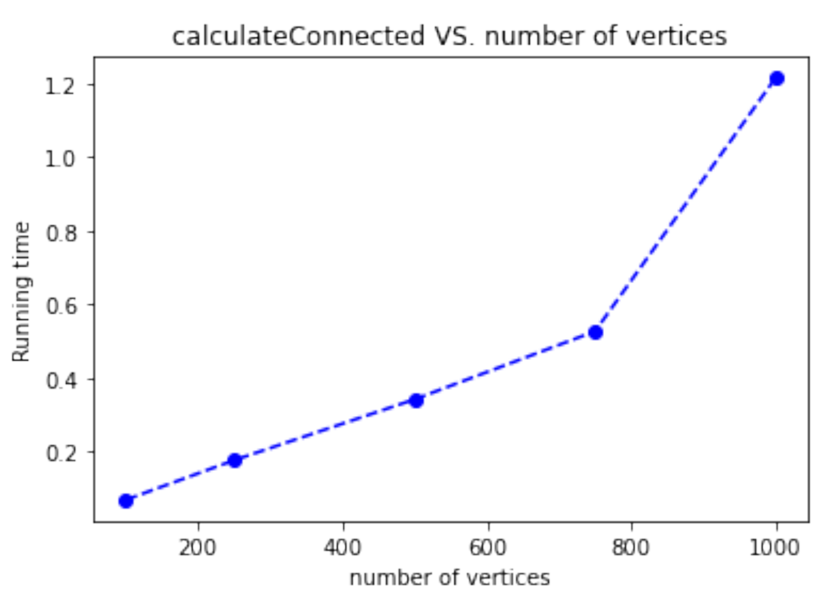


The fluctuation shown here may be the result of a pretty small running time value, which may lead to large deviation from the actual value.

6. calculateConnected( ):

In [13]:
```
plt.figure()
plt.title('calculateConnected VS. number of vertices')
plt.plot(vertices,calc_connect,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

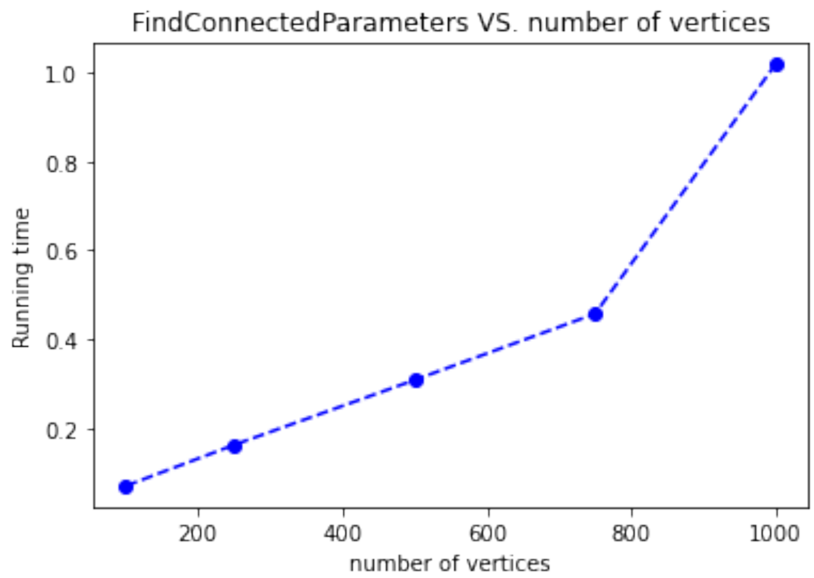
Out[13]: `Text(0, 0.5, 'Running time')`



7. FindConnectedParameters( ):

In [14]:
```
plt.figure()
plt.title('FindConnectedParameters VS. number of vertices')
plt.plot(vertices,find_parameter,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

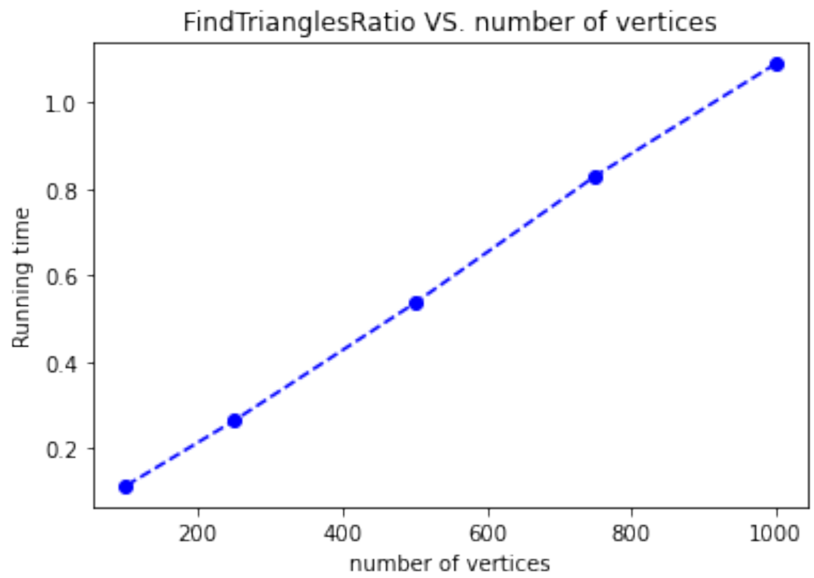
Out[14]: Text(0, 0.5, 'Running time')



8. FindTrianglesRatio( ):

In [15]:
```
plt.figure()
plt.title('FindTrianglesRatio VS. number of vertices')
plt.plot(vertices,find_ratio,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

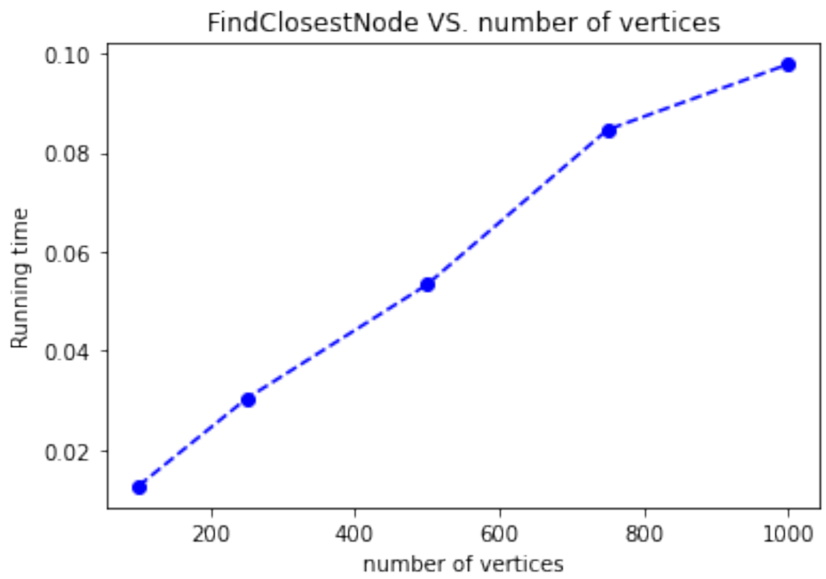
Out[15]: Text(0, 0.5, 'Running time')



9. FindClosestNode( ):

In [16]:
```
plt.figure()
plt.title('FindClosestNode VS. number of vertices')
plt.plot(vertices,find_closest,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

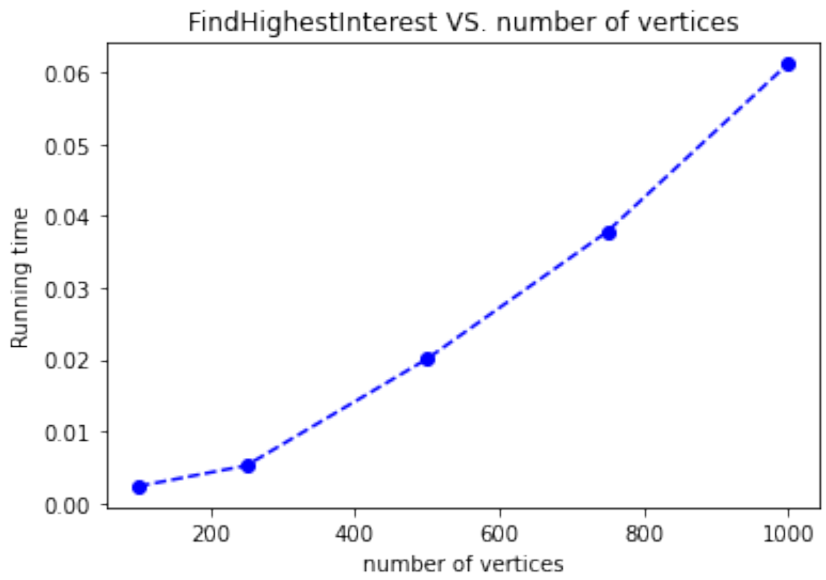
Out[16]:
```
Text(0, 0.5, 'Running time')
```



10. FindHighestInterest( ):

In [17]:
```
plt.figure()
plt.title('FindHighestInterest VS. number of vertices')
plt.plot(vertices,find_intere,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

Out[17]:
```
Text(0, 0.5, 'Running time')
```



11. FindDistanceRatio( ):

In [18]:
```
plt.figure()
plt.title('FindDistanceRatio VS. number of vertices')
plt.plot(vertices,find_dis_ratio,'bo--')
plt.xlabel('number of vertices')
plt.ylabel('Running time')
```

Out[18]: Text(0, 0.5, 'Running time')