# PHYS 243 Homework 1

## Carlos Olivas ID# 861045506

## Problem 1

Part 1:

In [2]:
```
def Fib_rec(n):
    if n == 0:
        return 1
    elif n == 1:
        return 1
    else:
        return Fib_rec(n-1) + Fib_rec(n-2)

def Fib_iter(n):

    if n ==0:
        return 1
    if n == 1:
        return 1

    num1 = 0
    num2 = 1

    for i in range(n):
        temp = num2
        num2 += num1
        num1 = temp

    return num2
```

Part 2:

The iterative implementation is faster.

Part 3:

In [3]:
```
import time
def timer(n, k, f):

    average_time = 0

    for i in range(k):
        start = time.time()

        f(n)

        end = time.time()

        average_time += (end - start)

    average_time /= k

    return average_time
```

Part 4:

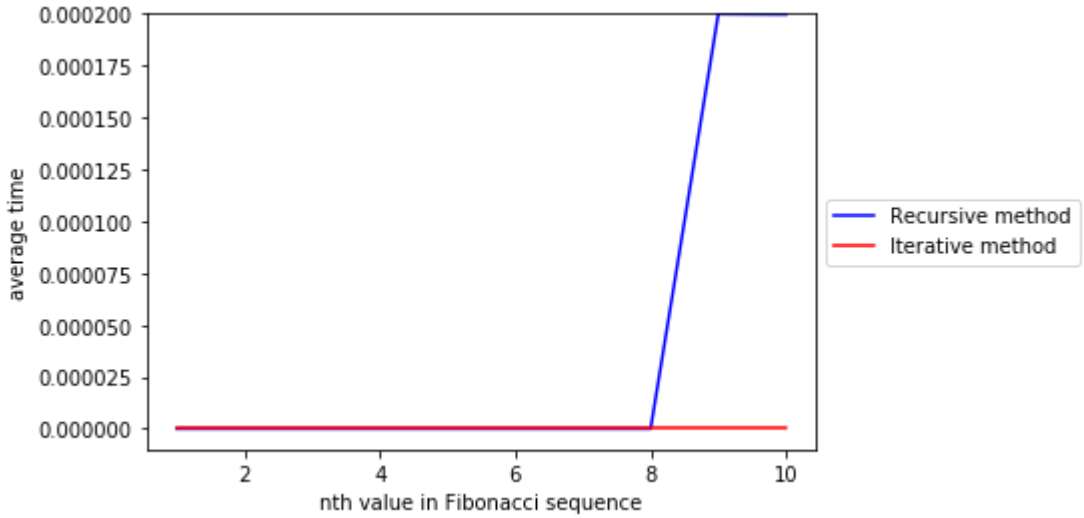
In [10]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

x_vals = np.array([i+1 for i in range(10)])

# get average times for recurssive implementation
y_vals_rec = list(map(lambda x: timer(x, 5, Fib_rec), x_vals))

y_vals_iter = list(map(lambda x: timer(x, 5, Fib_iter), x_vals))

plt.plot(x_vals, y_vals_rec, 'b')
plt.plot(x_vals, y_vals_iter, 'r')
plt.ylim(top=0.0002)
plt.ylabel("average time")
plt.xlabel("nth value in Fibonacci sequence")
plt.legend(["Recursive method", "Iterative method"], loc='center left',bbox_to_anchor=(1, 0.5))
plt.show()
```




# Problem 2

Part 1:

In [12]:
```
M = np.array([[1, -4, 2],[-4, 1, -2], [2, -2, -2]])
M_det = np.linalg.det(M)
M_t = np.transpose(M)
M_inv = []
try:
    M_inv = np.lingalg.inv(M)
except:
    pass

print("Determinant for M: ")
print(M_det)
print("Transpose of M: ")
print(M_t)
print("Inverse of M:")
print(M_inv)
```

```
Determinant for M:
54.00000000000001
Transpose of M:
[[ 1 -4  2]
 [-4  1 -2]
 [ 2 -2 -2]]
Inverse of M:
[]
```

Part 2:

In [13]:
```
M_eig = np.linalg.eig(M)
print("Eigenvalues for M:")
print(M_eig[0])
print("Eigenvectors for M:")
print(M_eig[1])
```

```
Eigenvalues for M:
[-3.  6. -3.]
Eigenvectors for M:
[[-0.74535599 -0.66666667  0.09505013]
 [-0.59628479  0.66666667  0.51960247]
 [ 0.2981424  -0.33333333  0.84910467]]
```

Part 3:

$$\nabla_A f(A) = \begin{bmatrix} 2x_{11}x_{22}x_{23} + x_{12}x_{13}x_{31} & x_{11}x_{13}x_{31} & x_{11}x_{12}x_{31} \\ -x_{33}^2x_{32} & x_{11}^2x_{23} & x_{11}^2x_{22} \\ x_{11}x_{12}x_{13} & -x_{33}^2x_{21} & -x_{32}x_{21} \end{bmatrix}$$

Part 4:

$$\begin{bmatrix} 6xy - yzsin(x) & 3x^2 + zcos(x) + 2yz^5 & ycos(x) + 5y^2z^4 \\ 3x^2 + zcos(x) + 2yz^5 & 2xz^5 & sin(x) + 10xyz^4 \\ ycos(x) + 5y^2z^4 & sin(x) + 10xyz^4 & 20xy^2z^3 \end{bmatrix}$$

# Problem 3

Part 1:

Validation samples are used to test and to adjust the parameters of a learning algorithm. This is to ensure that the learning algorithm works well with data that it hasn't seen yet. Test samples are examples that are not made available during the learning stage of development. The test samples are used to evualuate the effectiveness of the learning algorithm.

Part 2:

In supervised learning, the learning algorithm is given the labels of training sample. The learning algorithm is then meant to predict the labels of other data, based on the labels that were provided in the learning stage. In unsupervised learning, the algorithm is employed to find the possible labels for the data. The data's labels are not provided in the learning and validation stages.