String matching for same sized pattern and text

```
import time
import random as rn
import matplotlib.pyplot as plt

def stringMatch(text1,pattern):
    if (len(text1)!= len(pattern)):
        raise Exception("Invalid Input!")
    a=0

    for i in range (len(pattern)):
        if(text1[i]==pattern[i]):
            a+=1
        else:
            break

    if (a==len(pattern)):
        print("The text and the pattern match")

    else:
        print("The text and the pattern dont match")

import time
text=str(input("Please enter the text: "))
pattern=str(input("Please enter the pattern: "))

start=time.time()
stringMatch(text,pattern)
stop=time.time()

timeMatch=stop-start

Please enter the text: 999999999999999
Please enter the pattern: 999999999999999
The text and the pattern match

text=str(input("Please enter the text: "))
pattern=str(input("Please enter the pattern: "))

start=time.time()
stringMatch(text,pattern)
stop=time.time()

timeMismatchBest=stop-start

Please enter the text: 999999999999999
Please enter the pattern: 111111111111111
The text and the pattern dont match
```

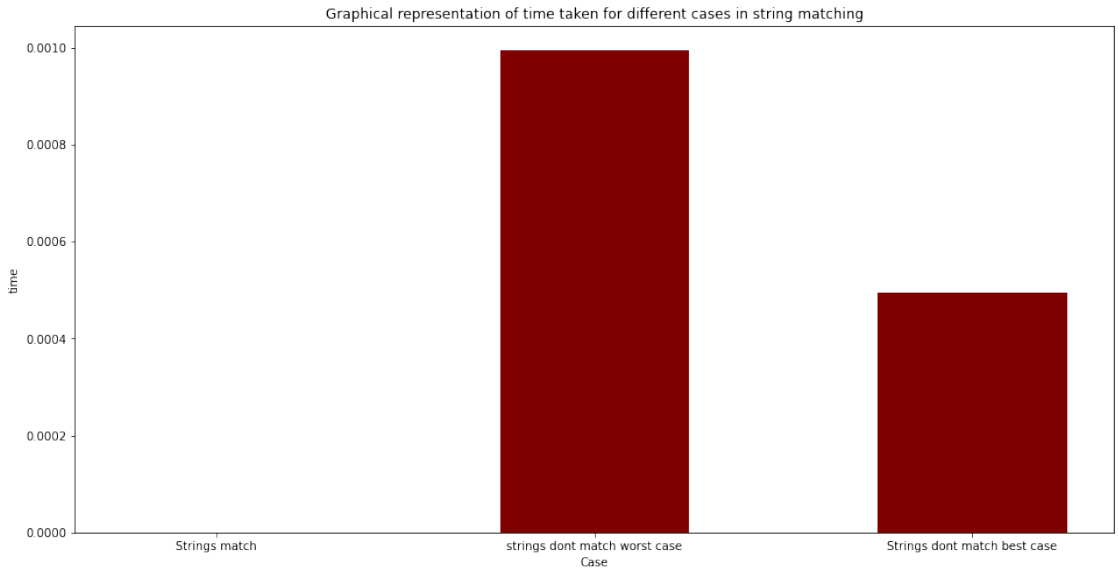
```
text=str(input("Please enter the text: "))
pattern=str(input("Please enter the pattern: "))

start=time.time()
stringMatch(text,pattern)
stop=time.time()

timeMismatchWorst=stop-start
```

```
Please enter the text: 987654321012345
Please enter the pattern: 987654321012344
The text and the pattern dont match
```

```
print(timeMismatchWorst,timeMismatchBest,timeMatch)
```

```
0.0009953975677490234 0.0004940032958984375 0.0
```

```
import numpy as np

data = {'Strings match':timeMatch, 'strings dont match worst
case':timeMismatchWorst, 'Strings dont match best
case':timeMismatchBest}

time = list(data.keys())
case = list(data.values())

fig = plt.figure(figsize = (16, 8))

# creating the bar plot
plt.bar(time,case, color ='maroon',width = 0.5)

plt.xlabel("Case")
plt.ylabel("time")
plt.title("Graphical representation of time taken for different cases
in string matching")
plt.show()
```



Time Complexity

Here the elements are compared one by one and hence the time complexity for checking if the same sized text and pattern are matching is O(n).

Naïve String matching

```
def naiveStringMatch(text,pattern):

    b=0

    for i in range (len(text)-len(pattern)+1):
        a=0
        for j in range (len(pattern)):
            if(text[j+i]==pattern[j]):
                a+=1

            else:
                b+=1
                break

        if (a==len(pattern)):
            print("The pattern is present at index",i)

    if(b==len(text)-len(pattern)+1):
        print("The pattern is not present in the text")


text=str(input("Please enter the text: "))
pattern=str(input("Please enter the pattern: "))
```

```
naiveStringMatch(text,pattern)

Please enter the text: 987987987
Please enter the pattern: 987
The pattern is present at index 0
The pattern is present at index 3
The pattern is present at index 6

text=str(input("Please enter the text: "))
pattern=str(input("Please enter the pattern: "))

naiveStringMatch(text,pattern)

Please enter the text: 987987
Please enter the pattern: 987987
The pattern is present at index 0

text=str(input("Please enter the text: "))
pattern=str(input("Please enter the pattern: "))

naiveStringMatch(text,pattern)

Please enter the text: 987987987
Please enter the pattern: 123
The pattern is not present in the text

import time
import random as rn

masterArray=[]
timeStringMatching=[]
size=[5,10,15,20,35,50,100,200,500,1000,2000,3000,4000,5000,6000,7000,
8000,9000,10000]

for i in range(len(size)):
    masterArray.append(str((rn.randint(10**(size[i]-4),10**(size[i])-
3))*10**3+932))

pattern=str(932)

for i in range (len(size)):

    start=time.time()
    naiveStringMatch(masterArray[i],pattern)
    stop=time.time()
    timeStringMatching.append(stop-start)
    print("----")
```

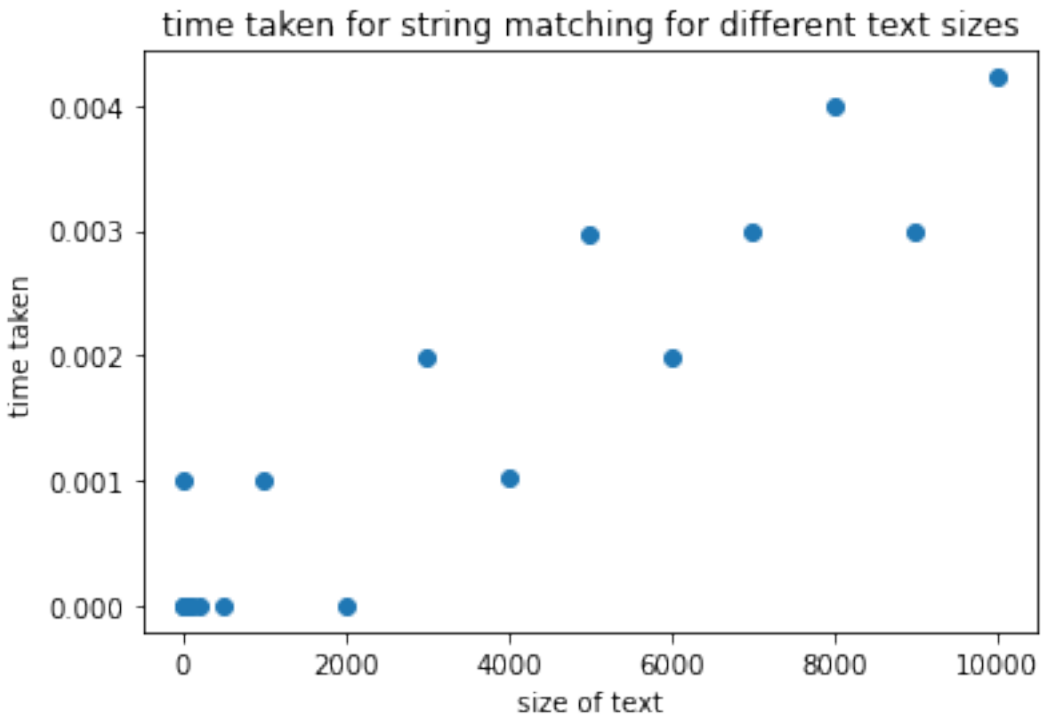
```
The pattern is present at index 5
----
The pattern is present at index 10
----
The pattern is present at index 15
----
The pattern is present at index 20
----
The pattern is present at index 34
----
The pattern is present at index 50
----
The pattern is present at index 100
----
The pattern is present at index 200
----
The pattern is present at index 500
----
The pattern is present at index 1000
----
The pattern is present at index 580
The pattern is present at index 1400
The pattern is present at index 2000
----
The pattern is present at index 3000
----
The pattern is present at index 827
The pattern is present at index 1046
The pattern is present at index 1727
The pattern is present at index 2246
The pattern is present at index 3096
The pattern is present at index 3228
The pattern is present at index 3328
The pattern is present at index 4000
----
The pattern is present at index 255
The pattern is present at index 1316
The pattern is present at index 1947
The pattern is present at index 2167
The pattern is present at index 3182
The pattern is present at index 3196
The pattern is present at index 3446
The pattern is present at index 5000
----
The pattern is present at index 523
The pattern is present at index 1579
The pattern is present at index 1595
The pattern is present at index 3206
The pattern is present at index 3500
The pattern is present at index 6000
```

```
----
The pattern is present at index 979
The pattern is present at index 990
The pattern is present at index 2367
The pattern is present at index 2846
The pattern is present at index 6999
----
The pattern is present at index 2321
The pattern is present at index 2858
The pattern is present at index 4468
The pattern is present at index 5111
The pattern is present at index 5842
The pattern is present at index 5858
The pattern is present at index 7496
The pattern is present at index 7760
The pattern is present at index 8000
----
The pattern is present at index 319
The pattern is present at index 2179
The pattern is present at index 5019
The pattern is present at index 5661
The pattern is present at index 8087
The pattern is present at index 9000
----
The pattern is present at index 164
The pattern is present at index 2436
The pattern is present at index 3045
The pattern is present at index 3781
The pattern is present at index 4202
The pattern is present at index 6556
The pattern is present at index 6835
The pattern is present at index 7600
The pattern is present at index 9526
The pattern is present at index 9710
The pattern is present at index 10000
----
```

```
print(timeStringMatching)
```

```
[0.0, 0.0, 0.000997781753540039, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0009963512420654297, 0.0, 0.001993417739868164,
0.0010228157043457031, 0.002962350845336914, 0.0019927024841308594,
0.0029904842376708984, 0.0039861202239990234, 0.002990007400512695 3,
0.004224300384521484]
```

```
plt.scatter(size,timeStringMatching)
plt.xlabel('size of text')
plt.ylabel('time taken')
plt.title('time taken for string matching for different text sizes')
plt.show()
```



Time Complexity

In this case we have to check the pattern at every index one by one. The time taken to navigate through the text element by element is O(n) and the time taken to check if the pattern exists at that index is of the order O(m), where n is the length of the text and m is the length of the pattern. Hence the time complexity for naive string matching is O(n) * O(m) or simply O(m*n).

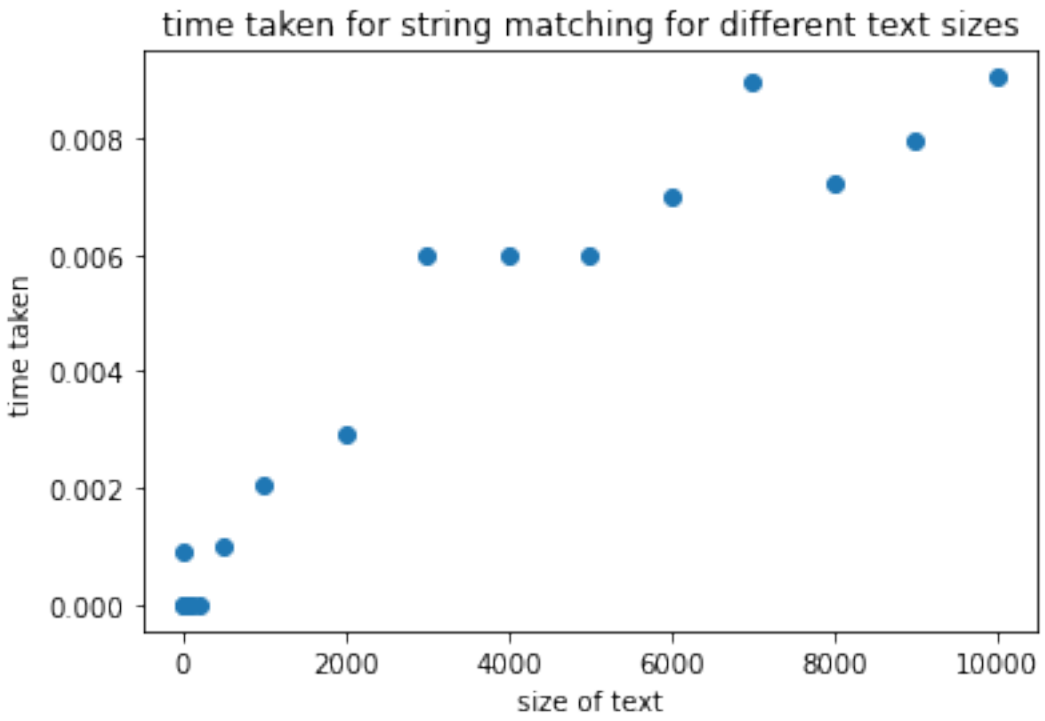
New naive string matching (treating pattern and text as numbers)

```
def newStringMatching (text,pattern):

    pattern1=int(pattern)
    for i in range (len(text)-len(pattern)+1):
        a=0
        b=0
        num=int(text[i])
        for j in range (i,i+len(pattern)-1):
            subtext=0
            num1=int(text[j+1])
            num=(10*num)+num1
            subtext=num

        if (subtext%13==pattern1%13):
            for k in range (len(pattern)):
```

```
                if(text[k+i]==pattern[k]):
                    a+=1

                else:
                    b+=1
                    break

                if (a==len(pattern)):
                    print("The pattern is present at index ",i)


timeNewNaiveStringMatching=[]

import time
for i in range (len(size)):

    start=time.time()
    newStringMatching(masterArray[i],pattern)
    stop=time.time()
    timeNewNaiveStringMatching.append(stop-start)
    print("----")

The pattern is present at index  5
----
The pattern is present at index  10
----
The pattern is present at index  15
----
The pattern is present at index  20
----
The pattern is present at index  34
----
The pattern is present at index  50
----
The pattern is present at index  100
----
The pattern is present at index  200
----
The pattern is present at index  500
----
The pattern is present at index  1000
----
The pattern is present at index  580
The pattern is present at index  1400
The pattern is present at index  2000
----
The pattern is present at index  3000
----
The pattern is present at index  827
The pattern is present at index  1046
```

```
The pattern is present at index  1727
The pattern is present at index  2246
The pattern is present at index  3096
The pattern is present at index  3228
The pattern is present at index  3328
The pattern is present at index  4000
----
The pattern is present at index  255
The pattern is present at index  1316
The pattern is present at index  1947
The pattern is present at index  2167
The pattern is present at index  3182
The pattern is present at index  3196
The pattern is present at index  3446
The pattern is present at index  5000
----
The pattern is present at index  523
The pattern is present at index  1579
The pattern is present at index  1595
The pattern is present at index  3206
The pattern is present at index  3500
The pattern is present at index  6000
----
The pattern is present at index  979
The pattern is present at index  990
The pattern is present at index  2367
The pattern is present at index  2846
The pattern is present at index  6999
----
The pattern is present at index  2321
The pattern is present at index  2858
The pattern is present at index  4468
The pattern is present at index  5111
The pattern is present at index  5842
The pattern is present at index  5858
The pattern is present at index  7496
The pattern is present at index  7760
The pattern is present at index  8000
----
The pattern is present at index  319
The pattern is present at index  2179
The pattern is present at index  5019
The pattern is present at index  5661
The pattern is present at index  8087
The pattern is present at index  9000
----
The pattern is present at index  164
The pattern is present at index  2436
The pattern is present at index  3045
The pattern is present at index  3781
```

```
The pattern is present at index  4202
The pattern is present at index  6556
The pattern is present at index  6835
The pattern is present at index  7600
The pattern is present at index  9526
The pattern is present at index  9710
The pattern is present at index  10000
----

print(timeNewNaiveStringMatching)

[0.0, 0.0, 0.0009188652038574219, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0009975433349609375, 0.002067089080810547, 0.0029168128967285156,
0.005980730056762695, 0.005978107452392578, 0.005980014801025391,
0.006986379623413086, 0.00896000862121582, 0.007205009460449219,
0.007972955703735352, 0.009040594100952148]

plt.scatter(size,timeNewNaiveStringMatching)
plt.xlabel('size of text')
plt.ylabel('time taken')
plt.title('time taken for string matching for different text sizes')
plt.show()
```



Optimized string matching

```
def optimizedStringMatching (text,pattern):

    pattern1=int(pattern)
    for i in range (len(text)-len(pattern)+1):

        num=int(text[i])
        for j in range (i,i+len(pattern)-1):
            subtext=0
            num1=int(text[j+1])
            num=(10*num)+num1
            subtext=num

        if (subtext==pattern1):
            print("The pattern exists at index ",i)




text='987987987'
pattern='987'

optimizedStringMatching(text,pattern)
```

```
The pattern exists at index  0
The pattern exists at index  3
The pattern exists at index  6
```

```
timeOptimizedNaiveStringMatching=[]

import time
for i in range (len(size)):

    start=time.time()
    optimizedStringMatching(masterArray[i],pattern)
    stop=time.time()
    timeOptimizedNaiveStringMatching.append(stop-start)
    print("----")
```

```
----
----
----
----
----
----
----
The pattern exists at index  125
----
----
----
```

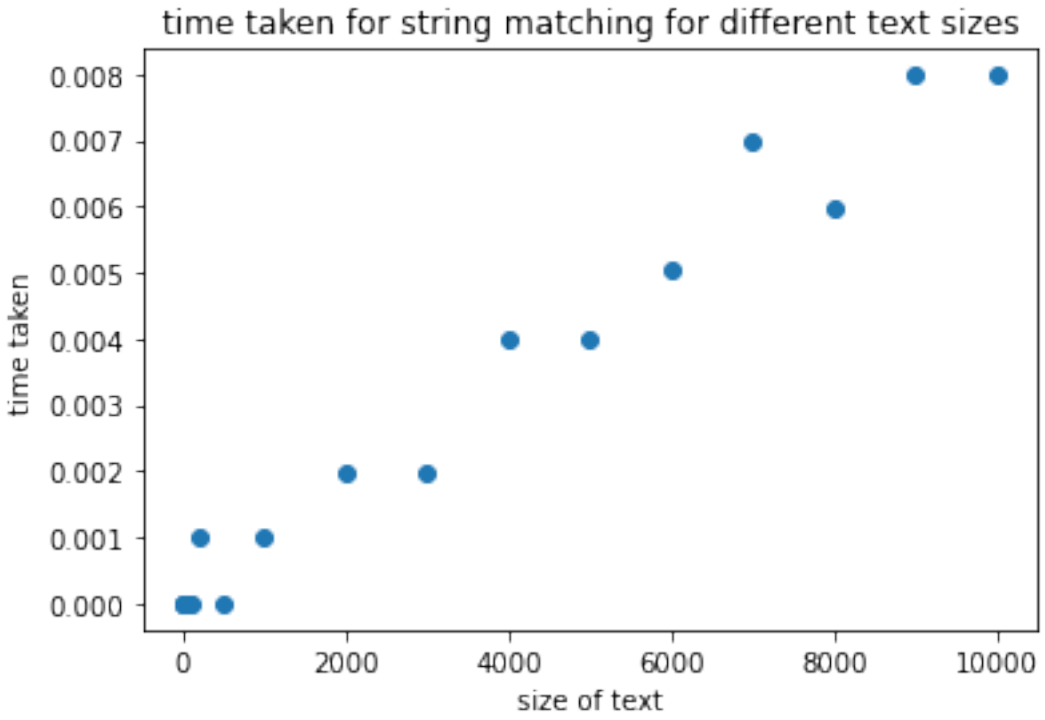
```
The pattern exists at index  115
The pattern exists at index  1615
----
The pattern exists at index  1999
The pattern exists at index  2404
----
The pattern exists at index  17
The pattern exists at index  1487
The pattern exists at index  2517
The pattern exists at index  2534
The pattern exists at index  2709
The pattern exists at index  3904
----
The pattern exists at index  1255
The pattern exists at index  2930
The pattern exists at index  3557
----
The pattern exists at index  657
The pattern exists at index  2751
The pattern exists at index  2856
The pattern exists at index  2919
The pattern exists at index  5898
----
The pattern exists at index  2193
The pattern exists at index  2458
The pattern exists at index  2571
The pattern exists at index  2711
The pattern exists at index  3838
The pattern exists at index  4245
The pattern exists at index  4604
The pattern exists at index  4887
The pattern exists at index  6124
The pattern exists at index  6672
----
The pattern exists at index  5107
----
The pattern exists at index  1874
The pattern exists at index  2037
The pattern exists at index  2210
The pattern exists at index  3024
The pattern exists at index  3319
The pattern exists at index  3423
The pattern exists at index  4997
The pattern exists at index  5436
The pattern exists at index  5573
The pattern exists at index  6596
The pattern exists at index  8441
The pattern exists at index  8451
----
The pattern exists at index  551
```

```
The pattern exists at index  834
The pattern exists at index  3514
The pattern exists at index  4124
The pattern exists at index  5062
The pattern exists at index  7569
----
```

```
print(timeOptimizedNaiveStringMatching)
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.000997304916381836, 0.0,
0.0009958744049072266, 0.0019931793212890625, 0.001993894577026367,
0.0039865970611572266, 0.00398707389831543, 0.0050525665283203125,
0.0069751739501953125, 0.005982875823974609, 0.007979631423950195,
0.007973670959472656]
```

```
plt.scatter(size,timeOptimizedNaiveStringMatching)
plt.xlabel('size of text')
plt.ylabel('time taken')
plt.title('time taken for string matching for different text sizes')
plt.show()
```



```
import time
```

```
text='85020052806708387373410506747387220252664473404173647874367378263262220335647371828007416332115274264384513287151082216316555456012200
78345001557218417051824287770607716444284307836101467235181462675302443182245362745688230063847302006533852763468202080156784710540553485331
```

615070662564613032305177186753760818220404031056370452587328410862041677667272671258764885635571010477548427387833820552558077873411616111573808688444161785515015567247755356050238442045314586806377136474722374561187645121870133653286782574307651611417420466388416646760065027284278774652146100085185082607768218824267276713707006343368736023384632083412803135434401810045166383128423374517712171838258536567300281801825052647811573811713731565486776850117315427025772282014544321846512115042145627480582570570085053481610348252668811444673215668556088656145060340224452271774125703337361203815054028216547402620543021445163046633552050366268562716357538812451170258368835172238404118543185241534401824232714137102051153075187517661202648821275505323186828782522603461704848086871345155676350115274786074342353324063522828728515878468720050733640543724356533684045385563344336276173882117008886744716283066586883424286613464137754638111241133362763816804270625338883442154402585457070170427825571513432805075086331223224207547325443603802007518235418715860857267815205351371470707800776600887762330463306225813332247276374513473532287784328878072530583143552658883502636787807188114515066287768327180885483586324885517201113053656542366028246621412510354733350547664350156313620728207807746860620083658060826136445045624161847105840734317841513321673740771151842604640272086863286541037331555225633133454402235600754101373834218130173111242741876065334240132218332052020740834248344455528673325170215384053554573252621543666737272730786341828780565506853531071648788745524627841321181367871884074167324320400580740431007412112283751385116257512302477211827110570327578454527507303386264813412365208288318104215688471637160653650514708036152217267637657622820514640756464274850278752584285135333647525541847754512501647375175217356644322177481800688168661120801667234058435462136014018345860307002746342562752155568375854441253744244645123043370722180382520482651838288841656537127880775045540344380626405505714737007717264415606243238805380814267055068306437564058188656400160017803833183226722062070288748333804058835110302737680073548114253001728554651743752235433407805712760137181025337255821608304710234756816044316830360145144528578866014416360531054032541676224356168551286302340352060886287686161751033385422658347124680270123832728501200476610014113440878082777303318807445421164345725837421262177003242146345744682123048611134475532404184482010527602643137257778386451076681122721541408866386016682403057687645222112856864728101241400601675763066376421856742676786565037453372881132220142433813446668671352275147656087040567741828518707563430832844488047860141381325810611320173067533605370250378458103485510212854375015358253448637317031806431102461434824182603581380245120320021254032570832736702341186684478082066127410431257566643378046134772655113240442372016811776776748538016434702076858326242162186173820520782814834560245000802335244204533162741325301704877041452181335081575661245567740674846526537517871817433072572127483512328482622328761358255843746540370527646504230513563001371340168054771240544432655000483102500875635408287085487727020110535107755384045338856030862056048244701263123616261671172361134123666160153513026045233273370337218063853154686104276114561602515381122740438437445471802371532877460366737325573105233073658570177778811536680856073268606354074752676708020845888144038784124628401401172067177003155133382224333345631745403720540646277480370347727155

```
77842362061666340557186315802240507800736501105255455613374746432417316501873108812802027300216031627710478025785637420345613654237512442834041116018526333560252425548048386077858288225665620711135216836465364741681726467305386380820088371816622137125750413871417721841083633424112678242845287323035883888504364720302317154483337551024020342511264770638042571280647531171007786157662600837227330863322033271436405824540412838357608501551853256770316302628706423040335666173628700573427362105555820332008843220378220816468444375862560255543714277127128431566325600838657803856601225264814405414475231802480027677464101117747148826277867054340533424018177530623264883336235558533446163805115151864345251825012620146288263276428182365510574051276866487127447816114103283036860685167746452237642557234147031555628022868111344813561084217562107170658465363126842141185111482167764660664242851278850035677383580532462164487761558054422138613111875067605368035217174810265535835238851084725142482330536120601637727342542284872300673268245770154251073771075424475718200413243432377261228072061463183728747304814545750716023780270504477414812553304710760828185557417534126545360642110362027445468413817834413527035111421302823383231663'
pattern='257512302477211827110570327578454527507303386264813412365208288318104215688471637160653650514708036 1'

print("By using modulo method")
start=time.time()
newStringMatching(text,pattern)
stop=time.time()

timeModulo=stop-start

print("Time taken for execution is ",timeModulo)

print("By using type casting method")

start=time.time()
optimizedStringMatching(text,pattern)
stop=time.time()

timeTypecasting=stop-start

print("Time taken for execution",timeTypecasting)
```

```
By using modulo method
The pattern is present at index  1911
Time taken for execution is  0.11899566650390625
By using type casting method
The pattern exists at index  1911
Time taken for execution 0.13049912452697754
```

```
data = {'Modulo method':timeModulo, 'type casting
method':timeTypecasting}
```

```
time = list(data.keys())
case = list(data.values())

fig = plt.figure(figsize = (16, 8))

# creating the bar plot
plt.bar(time,case, color ='maroon',width = 0.5)

plt.xlabel("Technique")
plt.ylabel("time")
plt.title("Graphical representation of time taken for different string
matching techniques")
plt.show()
```