Ejercicios examen

April 22, 2018

```
In [3]: def cadenaMarkov(punto,dimCadena,N,delta):
            cadena = [punto]
            for i in xsrange(dimCadena):
                rand = randint(0,N-1)
                pN = copy(cadena[i])
                pN[rand] = pN[rand] + 2*random()*delta - delta
                res = 0
                for j in xsrange(len(pN)):
                    res = res + pN[j]^2
                if res <= 1:
                    cadena.append(pN)
                else:
                    cadena.append(cadena[i])
            return cadena
```

```
In [61]: def V(N,dimCadena,delta):
             dentro = 0
             hipercilindro = []
             punto = [0 for int in xsrange(N-1)]
             puntos = cadenaMarkov(punto,dimCadena,N-1,delta)
             for i in xsrange(len(puntos)):
                 nP = copy(puntos[i])
                 nP.append(2*random() - 1)
                 modulo = 0
                 for j in xsrange(len(nP)):
                     modulo = modulo + nP[j] ^2
                 if modulo <= 1:
                     dentro = dentro + 1
                 hipercilindro.append(nP)
             return dentro/dimCadena.n()
```

```
In [72]: V(3,10,0.3)
```

```
Out[72]: 0.700000000000000
```

```
In [74]: def prob():
             dado1 = randint(1,6)
             dado2 = randint(1,6)
```

```
            if dado1 + dado2 >= 10:
                dado3 = randint(1,6)
                if dado1 + dado2 + dado3 >= 15:
                    return True
            return False
```

```
In [76]: def ejercicio1(N):
             res = 0
             for int in xsrange(N):
                 if prob():
                     res = res + 1
             return res/N.n()
         print ejercicio1(10^5)
```

```
0.0735400000000000
```

```
In [97]: def semip1(R):
             res = 0
             for i in xsrange(1,R+1):
                 L = list(i.factor())
                 if len(L) == 1:
                     if L[0][1] == 2:
                         res = res + 1
                 elif len(L) == 2:
                     if L[0][1] == 1 and L[1][1] == 1:
                         res = res + 1
             return res/R.n()
```

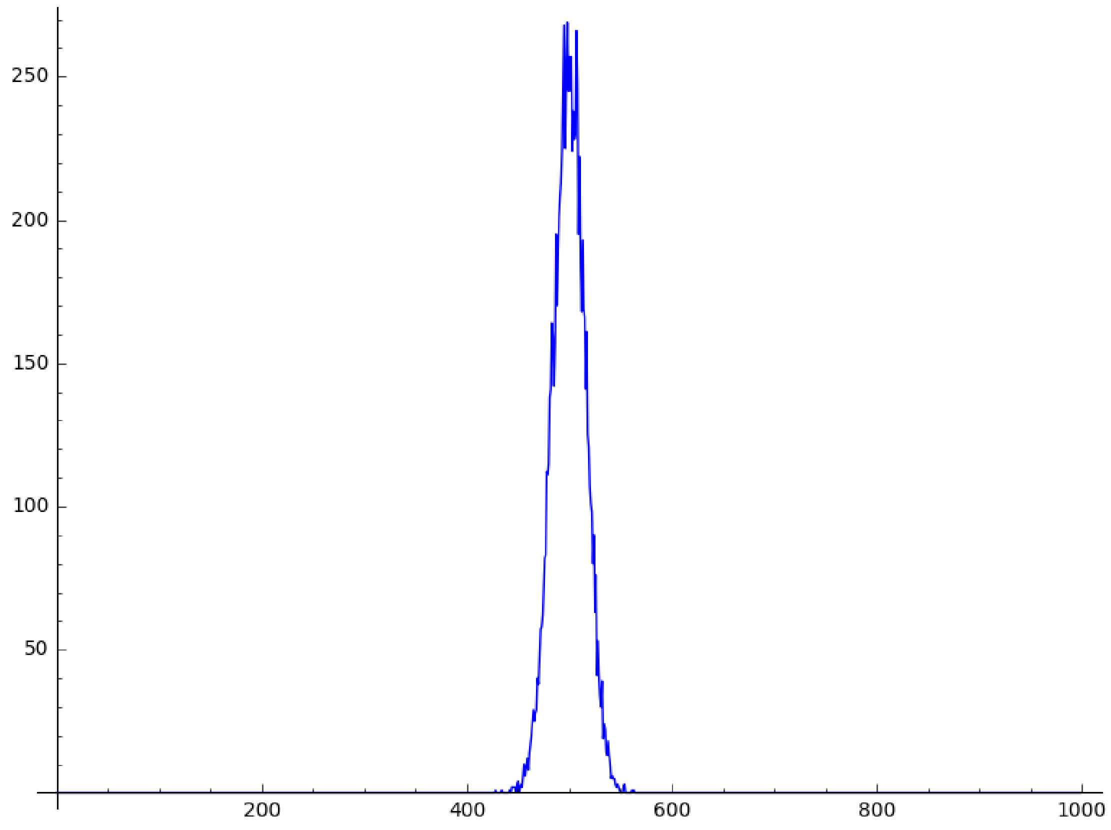
```
In [98]: time semip1(10^5)
```

```
CPU times: user 1.79 s, sys: 108 ms, total: 1.9 s
Wall time: 1.76 s
```

```
Out[98]: 0.233780000000000
```

```
In [99]: def semip2(R,N):
             res = 0
             for int in xsrange(N):
                 num = randint(1,R)
                 L = list(factor(num))
                 if len(L) == 1:
                     if L[0][1] == 2:
                         res = res + 1
                 elif len(L) == 2:
                     if L[0][1] == 1 and L[1][1] == 1:
                         res = res + 1
             return res/N.n()
```

```
In [101]: time semip2(10^5,10^4)

CPU times: user 372 ms, sys: 60 ms, total: 432 ms
Wall time: 378 ms


Out[101]: 0.232700000000000

In [1]: def rachas(L):
            rachas = 0
            for i in xsrange(len(L)):
                if i == 0:
                    rachas = rachas + 1
                    continue
                if L[i] != L[i - 1]:
                    rachas = rachas + 1
            return rachas

In [31]: media = 0
         L = [[i,0] for i in xsrange(1000)]
         for int in xsrange(10^4):
             unos = [randint(0,1) for int in xsrange(1000)]
             racha = rachas(unos)
             media = media + racha
             L[racha - 1][1] = L[racha - 1][1] + 1
         media = media/10^4
         print media

2503029/5000


In [33]: line2d(L)

Out[33]:
```



```
In [34]: def monedatrucada(k):
             if random() < 0.5 + k*0.01:
                 return 0
             return 1

In [121]: def ciudades(n,semilla):
              set_random_seed(semilla)
              L = [[random(),random()] for int in xsrange(n)]
              set_random_seed()
              return L
          def distancia(x,y):
              return sqrt((x[0] - y[0])^2+(x[1] - y[1])^2)
          def distanciaT(L):
              res = 0
              for i in xsrange(len(L)-1):
                  res = res + distancia(L[i],L[i+1])
              return res

In [122]: def barajar(L):
              res = []
              prob = random()
```

```
            if prob < 0.2:
                ran = randint(0,len(L)//2)
                res.append(L[ran:])
                res.append(L[:ran-1])
            elif prob > 0.2 and prob < 0.6:
                elem = L.pop(randint(0,len(L) - 1))
                L.insert(randint(0,len(L) - 1),elem)
                res = copy(L)
            else:
                i1 = randint(0,len(L) - 1)
                i2 = randint(0,len(L) - 1)
                elem = L[i1]
                L[i1] = L[i2]
                L[i2] = elem
                res = copy(L)
            return res
```

```
In [123]: def TSP0(L,N):
              res = copy(L)
              print res[0][0]
              for int in xsrange(N):
                  L = barajar(L)
                  if distanciaT(L) < distanciaT(res):
                      res = copy(L)
              return distanciaT(res),L
```

```
In [131]: print TSP0(ciudades(20,54321),1000)
```

```
0.27632720136



    ---------------------------------------------------------------------------

    TypeError                                 Traceback (most recent call last)

    <ipython-input-131-995f5e5d2c70> in <module>()
  ----> 1 print TSP0(ciudades(Integer(20),Integer(54321)),Integer(1000))


    <ipython-input-123-7dd6d0db245c> in TSP0(L, N)
      4     for int in xsrange(N):
      5         L = barajar(L)
  ----> 6         if distanciaT(L) < distanciaT(res):
      7             res = copy(L)
      8     return distanciaT(res),L


    <ipython-input-121-6c1655d81363> in distanciaT(L)
```

```
          9     res = Integer(0)
         10     for i in xsrange(len(L)-Integer(1)):
    ---> 11         res = res + distancia(L[i],L[i+Integer(1)])
         12     return res


        <ipython-input-121-6c1655d81363> in distancia(x, y)
          5     return L
          6 def distancia(x,y):
    ----> 7     return sqrt((x[Integer(0)] - y[Integer(0)])**Integer(2)+(x[Integer(1)] - y[Int
          8 def distanciaT(L):
          9     res = Integer(0)


        TypeError: unsupported operand type(s) for -: 'list' and 'list'
```

```
In [ ]:
```