```
from __future__ import division
import sys, random, math
from common import *
import numpy as np
sys.dont_write_bytecode = True


random.seed(1)

class generate:
  def __init__(i, lo, hi, n):
    i.lo = lo
    i.hi = hi
    i.n = n
  def generate_x(i):
    x= [i.lo + (i.hi-i.lo)*random.random() for _ in range(i.n)]
    return x

class fonseca:
  def __init__(i):
    i.lo = -4
    i.hi = 4
    i.n = 3

  def gen(i):
    return generate(i.lo, i.hi,i.n)

  def f1_plus_f2(i, x_list):
    n = i.n
#     print x_list
    def f1_sum(x_list, n):
          value = []
          for item in x_list:
            value.append((item - 1/math.sqrt(n))**2)
          return sum(value)

    def f2_sum(x_list, n):
          value = []
          for item in x_list:
            value.append((item + 1/math.sqrt(n))**2)
          return sum(value)

    f1 = 1 - math.e ** (-1* f1_sum(x_list, n))
    f2 = 1 - math.e ** (-1* f2_sum(x_list, n))
#     print f1+f2
    return f1+f2

'''kusarvs'''
class kursawe:
  def __init__(i):
    i.lo = -5
    i.hi = 5
    i.n = 3

  def gen(i):
    return generate(i.lo, i.hi, i.n)

  def f1_plus_f2(i, x_list):
    n = i.n
    def f1_inner(x_list, n):
      value = []
      for i in range(n-1):
        value.append(-10 * math.e **(-0.2 * math.sqrt(x_list[i]**2 + x_list[i+1]
**2)))
      return value

    def f2_inner(x_list, n):
          value = []
          a = 0.8
          b = 3
          for item in x_list:
                value.append(abs(item)**a + 5 * math.sin(item)**b )
          return value
```



```
    f1 = sum(f1_inner(x_list, n))
    f2 = sum(f2_inner(x_list, n))
#     print f1+f2
    return f1+f2



'''hello'''
def find_max_min(model, gen):
  # model = eval(model+"()")
  min = 10**(5)
  max = -10**(5)
  for i in range(100000):
    temp = model.f1_plus_f2(gen.generate_x())
    if temp > max:
      max = temp
    if temp < min:
          min = temp
  return min, max


def energy(x, min, max):
  e = (x - min)/(max - min)
  return e
#

def neighbor(old, generator):  # can put in to generator
  for i in range(len(old)):
    if random.random() <0.33:
      old[i] = generator.generate_x()[i]
  return old

def P(old, new, t):
  prob = math.e**((old - new)/t)
  return prob

def say(mark):
  sys.stdout.write(mark)
  sys.stdout.flush()

def sa():
  model_str = raw_input("Type 1 for fonseca and 2 for kursawe:")
  if (model_str) == '1':
    model = fonseca()
  elif (model_str) == '2':
       model = kursawe()
  else:
    print "please type 1 or 2!"
    exit()
#   model = ()
#   model = kursawe()
#   x = generate(model.lo, model.hi, model.n)
  generator = model.gen()
  min, max = find_max_min(model, generator)
#   print min, max
#   min, max = 0.98, 2.0
  s = generator.generate_x()
#   print s
  e = energy(model.f1_plus_f2(s), min, max)
#   print e
  sb = s
  eb = e
  k = 1
  kmax = 1000
  while k < kmax:
    sn = neighbor(s, generator)
    en = energy(model.f1_plus_f2(sn), min,max)
    if en < eb:
          sb = sn
          eb = en
          say('!')
    if en < e:
      s = sn
```

```
          e = en
          say('+')
        elif P(e, en, (k/kmax)) < random.random():
          s = sn
          e = en
          say('?')
        say('.')
        k = k+1
        if k % 40 == 0:
          print "\n"
          say(str(round(eb,3)))
    #   print "\n"
    #   say(str(sb))
      return sb
#
```

```
if __name__ == "__main__": sa()
```

```
from __future__ import division
import sys, random, math
from common import *
import numpy as np
sys.dont_write_bytecode = True



random.seed(1)

class generate:
  def __init__(i, lo, hi, n):
    i.lo = lo
    i.hi = hi
    i.n = n
  def generate_x(i):
    x= [i.lo + (i.hi-i.lo)*random.random() for _ in range(i.n)]
    return x

class fonseca:
  def __init__(i):
    i.lo = -4
    i.hi = 4
    i.n = 3

  def gen(i):
    return generate(i.lo, i.hi,i.n)

  def f1_plus_f2(i, x_list):
    n = i.n
#     print x_list
    def f1_sum(x_list, n):
          value = []
          for item in x_list:
            value.append((item - 1/math.sqrt(n))**2)
          return sum(value)

    def f2_sum(x_list, n):
          value = []
          for item in x_list:
            value.append((item + 1/math.sqrt(n))**2)
          return sum(value)

    f1 = 1 - math.e ** (-1* f1_sum(x_list, n))
    f2 = 1 - math.e ** (-1* f2_sum(x_list, n))
#     print f1+f2
    return f1+f2
# def score():

def score(raw_energy, min, max):
  return(raw_energy - min)/(max - min)

def find_max_min(model, gen):
  # model = eval(model+"()")
  min = 10**(5)
  max = -10**(5)
  for i in range(100000):
    temp = model.f1_plus_f2(gen.generate_x())
    if temp > max:
      max = temp
    if temp < min:
          min = temp
  return min, max

## score

def optimal_neighbor(solution, model, min, max):
  optimized_index = random.randint(0, len(solution)-1)
  increment = (model.hi - model.lo)/10
  temp_min = 10*(5)
#   print "old solution : %s" % solution
  for _ in range(10):
    solution[optimized_index] = model.lo+increment
```



```
    temp = score(model.f1_plus_f2(solution), min, max)
    if temp < temp_min:
       temp_min = temp
#   print "new solution : %s" % solution
  return solution

#   for i in range(len(old)):
#     if random.random() <=0.3:
#       old[i] = generator.generate_x()[i]
#   return old

def maxwalksat():
  max_tries = 50
  max_changes = 2000
  model = fonseca()
  generator = model.gen()
  min, max = find_max_min(model, generator)
  threshold = 0.1
  total_loop = 0
  total_tries = 0
  final_score = 0
  p = 0.25
#   print threshold
#
#   print solution
  for _ in range(max_tries):
    total_tries += 1
    solution = generator.generate_x()
#     print 'try {0} time(s) with solution {1}'.format( total_tries, solution)
    for _ in range(max_changes):
      final_score=score(model.f1_plus_f2(solution),min, max)
#       print "final score: %s" % final_score
      if final_score <= threshold:
        print "p : %s" % p
        print "threshold : %s" %threshold
        print "total tries: %s" % total_tries
        print "total changes: %s" % total_loop
        print "min_energy:{0}, max_energy:{1}".format(min, max)
        print "min_energy_obtained: %s" % model.f1_plus_f2(solution)
        print "solution : %s" % solution
        print "score: %s" % final_score
        return solution
      if p < random.random():
        solution[random.randint(0,2)] = generator.generate_x()[random.randint(0,
2)]
      else:
        solution = optimal_neighbor(solution, model, min, max)
      total_loop +=1

#       c =  generator.generate_x()

if __name__ ="__main__": maxwalksat()

```

```
##============results for fonseca======================

..!+..?..+..+...!+..?.+....?..+.+.......?......+.+.+.?..
0.932......!+.!+.?...+.+.?...+......+..+........?...+.....
0.882....+...+..+..!+..?...+...?...+.+..+.?..+...........
0.867......+...+................+......+.........
0.867.?.....+...........+..+.+..?.......+.!+.?........
0.612?..+.+..+.?...+....+...?.+......+................
0.612.................+.......................
0.612+.?.....+.+...........+.....?.......+.+....?.+...+.
0.612.+....?....+..+...?...+..+...?.+.+............+.?..
0.612.+....+.........+...........?.+...........
0.612....?.......+.+...!+.!+.?......?.+........+........
0.407..?.+..+.....?..+.....+.+..........?...+.+.....+..
0.407+.+.................+.?........+...........
0.407...........+.............?.+...+.+..+......?.+.
0.407........?.+......+...+.+..........?...........
0.407+.+.+.+....+.+.....?..........+........+........?.
0.407+.+..+.+..?...+...........+..?..+................
0.407.....................+...!+...?..............
0.404...+.?........+.......+..?.+.....+....+.....?...+.
0.404+...?..+....+...+..+.+.+......?.........+..+...?....
0.404.+.+.+....+.........?.+...+........?...........
0.404..............+.........!+..?.....?.+..+......
0.401.........+...+.!+.?.........................
0.292.........?.....+..+.......................
0.292...........+.+...?........+....+...?..+.!+.!+.?.....
0.037

##============results for kursawe======================

?.?.!+..?..?.?.+....?..+..+.+.+...?.?...+..?.+.+..?.?.+.+.?..+..
0.271?.+.?..+.!+..?.?...+.?...?....?.?.+.+...+..+.!+..?.?..?.+..+.?.?.+.
0.2.?.+..+.....+..+.+..?.+..?....?...+..+.+.?..+.?..?..+.?.+..!+.
0.16..?.+.+....+..?.?..?.+.+.+.+..?..?..?...+..+....?.+.+....+.+.
0.16.?..+.....+..+.?..?.+...+.+.+....?...+..+.?....?...+.?...
0.16.?....?.+....?.+.+..?.+.+..+.+..+.?.+.+...?.+...+..?...+....
0.16?..+.+..?....?.?..+.+....+..?.+..+.+...+.....?.+.+.?.....+.
```



```
0.16?..?.+...+..+..+.?.?..?..+.+...+.+.?...+...+.?.+.+......?.+.?.
0.16+.+...?.+...?.?...?.+.........?....+.?.?.+.+.....+.....
0.16+....?.+.?.+....?.+.+..?.+.+.........?.+.+.+..+..?...?.+.+..
0.16..?..+.+.+.?.+.......?..?.+.+.+..?.+...+.?....+..?.+.+.+....
0.16.....+........?..+.+.!+.?..+.........?.+.+.+.....+...
0.088...?....+..+..+....+.?.+....+...?.......+.........
0.088?...+.+.......?....+.+.+.+...+...?...+..+...+.+..?..?..+.
0.088+...+.!+....?..+.?....?..+..+.?.+...?.+.+..?.+...........
0.086+.....?..+.?..?..+..+.+.....?.+...+.............?.+.
0.086+.....?..+....+..!+....?...?......+.?.?.+.+.+..?.+..?....
0.059.?.+.+.......?.......+.+.....?...+..+.+..?.....+.?.+..
0.059.................?.+.+..+..?...+......?...?.+.+..?.+.
0.059.?.+..+.+.+......?...+..?..+..+.+..+.+...+.+.+....?..+....
0.059+.....+..+............?..+............?....+..
0.059...?.+.+.+..+....?...+.+..?.+.+.+........?.+......+.?.?..
0.059..+.+.......+...?.+..+.+.....?....+.+...+......+.+.+..
0.059........?.?.+....+..?.+...?.....+...?..+..+.+.+.+...+..
0.059..+....?..+..?..+...+...+.........+.+...?.+.?..+.+...?.
0.059
```

```
#=======================experiment 1=========================================

p : 0.25
threshold : 0.1
total tries: 4
total changes: 7444
min_energy:0.98516179182, max_energy:2.0
min_energy_obtained: 1.06356696583
solution : [0.6219060794448481, 0.6048121141877392, 0.2708728258004989]
score: 0.0772587919733

#=======================experiment 2=========================================

p : 0.5
threshold : 0.1
total tries: 4
total changes: 6484
min_energy:0.98516179182, max_energy:2.0
min_energy_obtained: 1.04467426371
solution : [−0.4844270550441525, −0.34626814383535365, −0.3654653426669592]
score: 0.0586423248635

#=======================experiment 3=========================================
p : 0.75
threshold : 0.1
total tries: 6
total changes: 10157
min_energy:0.98516179182, max_energy:2.0
min_energy_obtained: 1.08458753824
solution : [0.20948483978895016, 0.4859757044184825, 0.49765892744329854]
score: 0.0979720172304

#=======================conclusion===========================================

Based on the results from my experiments with different p's, I can conclude
that p = 0.5 is a better probability for maxwalksat than other values in
terms of total changes used in the algorithm. The change times is 6484 for
p = 0.5, which is less than the other two, 7444 and 10157. Probably because
p = 0.5 will give the equal chance for both random jump or local search.
```