

Sep 23, 14 11:37

csc791sbse:hw4:Fu

Page 1/3

```

from __future__ import division
import sys, random, math
from models import *
from sk import *
5 from base import *
import numpy as np
from xtile import *
sys.dont_write_bytecode = True
@printlook
10 def sa(model):
    def P(old, new, t):
        prob = math.e**((old - new)/t)
        return prob
    min_energy, max_energy = model.baseline()
    s = model.generate_x()
    e = model.norm(model.getDepen(s))
    sb = s
    eb = e
    k = 1
    20 icontrol = Control(model)
    while k < Settings.sa.kmax:
        stopsign = icontrol.next(k) #true ---stop
        if stopsign:
            break
        25 sn = model.sa_neighbor(s)
        en = model.norm(model.getDepen(sn))
        icontrol.logxy(sn)
        temp = (k/Settings.sa.kmax)*Settings.sa.cooling
        if en < eb:
            30 sb = sn
            eb = en
            say('!')
            if en < e:
                s = sn
                e = en
                say('+')
            elif P(e, en, temp) < random.random():
                s = sn
                e = en
                say('?')
            say('.')
            k = k + 1
            if k % 50 == 0:
                print "\n"
                say(str(round(eb,3)))
            45 print "\n"
            printReport(model)
            print "\n-----\n:Normalized Sum of Objectives: ", str(round(eb,3)), "\n:Solution", sn
            lohi=printRange(model)
            50 return eb,lohi
#
@printlook
def mws(model):
    55 min_energy, max_energy = model.baseline()
    total_changes = 0
    total_tries = 0
    norm_energy = 0
    eraScore = []
    60 control = Control(model)
    optimalsign = False
    solution = model.generate_x()
    norm_energy = model.norm(model.getDepen(solution))
    for k in range(Settings.mws.max_tries):
        65 total_tries += 1
        for _ in range(Settings.mws.max_changes):
            stopsign = control.next(total_changes) #true ---stop
            if stopsign:
                break
            if norm_energy < Settings.mws.threshold:
                70 optimalsign = True
                break
            if random.random() < Settings.mws.prob:

```

Sep 23, 14 11:37

csc791sbse:hw4:Fu

Page 2/3

```

    solution[random.randint(0,model.n-1)] = model.generate_x()[random.ran
t(0,model.n-1)]
    75 control.logxy(solution)
    say("+")
    else:
        solution = model.mws_neighbor(solution)
        control.logxy(solution)
    80 say("!")
    say(".")
    if total_changes % 50 == 0:
        print "\n"
        say(str(round(model.norm(model.getDepen(solution)), 3)))
        total_changes +=1
    85 if optimalsign & k == Settings.mws.max_tries-1:
        say("\n")
        say(str(round(model.norm(model.getDepen(solution)), 3)))
        print "\n"
        90 print "total tries: %s" % total_tries
        print "total changes: %s" % total_changes
        print "min_energy:{0}, max_energy:{1}".format(min_energy, max_energy)
        print "min_energy_obtained: %s" % model.getDepen(solution)
        printReport(model)
        95 lohi =printRange(model)
        print "\n-----\n:Normalized Sum of Objectives: ", str(round(norm_energy,3)), "\n:Solution
",solution, "\n"
        return norm_energy, lohi

100 def printReport(m):
    for i, f in enumerate(m.log.y):
        print "\n<%s" % i
        for era in sorted(m.history.keys()):
            # pdb.set_trace()
            105 log = m.history[era].log.y[i]
            print str(era).rjust(7), xtile(log._cache, width = 33, show = "%5.2f", lo
= 0, hi = 1)

    def printRange(m):
        lo = []
        lohi = []
        110 # print sorted(m.history.keys())
        for i, f in enumerate(m.log.y):
            tlo=10**5
            thi=-10**5
            115 for era in sorted(m.history.keys()):
                # pdb.set_trace()
                if m.history[era].log.y[i].lo < tlo:
                    tlo= m.history[era].log.y[i].lo
                if m.history[era].log.y[i].hi > tlo:
                    thi= m.history[era].log.y[i].hi
            120 lohi.append(tlo)
            lohi.append(thi)
        return lohi
        # print "\n the range of f%s is %s to %s " % (i, str(tlo), str(thi))

125 @demo
def start(): #part 5 with part 3 and part4
    r = Settings.other.repeats
    rlohi=[] # stupid codes here, to be fixed
    130 f1lo = []
    f1hi = []
    f0lo = []
    f0hi = []
    f2lo = []
    f2hi = []
    135 for klass in [Schaffer, Fonseca, Kursawe, ZDT1, ZDT3, Viennet3]:
        print "\n!!!!", klass.__name__
        for searcher in [sa, mws]:
            name = klass.__name__
            n = 0.0
            reseed()
            # scorelist = []
            140 for _ in range(r):

```

Sep 23, 14 11:37

csc791sbse:hw4:Fu

Page 3/3

```

x, lohi=searcher(klass()) # lohi is a list containing [lo,hi] pairs of f
1&f2
145 #=====part 5=====
    rlohi.append(lohi)
    for i in range(0, r):
        f0lo.append(rlohi[i][0])
        f0hi.append(rlohi[i][1])
150        f1lo.append(rlohi[i][2])
        f1hi.append(rlohi[i][3])
        if name == "Viennet3": # f1, f2, f3
            f2lo.append(rlohi[i][4])
            f2hi.append(rlohi[i][5])
155    print "\n# The range of f0 during %s repeats is from %s to %s " \
          % (r, str(round(sorted(f0lo)[0], 3)), str(round(sorted(f0hi)[-1])))
    )
    print "\n# The range of f1 during %s repeats is from %s to %s " \
          % (r, str(round(sorted(f1lo)[0], 3)), str(round(sorted(f1hi)[-1])))
    if name == "Viennet3":
160        print "\n# The range of f1 during %s repeats is from %s to %s " \
              % (r, str(round(sorted(f2lo)[0], 3)), str(round(sorted(f2hi)[-1])))
    rlohi = []
    #=====part 5 ends=====

165    #the following codes for hw3
    # n += float(x)
    # scorelist +=[float(x)]
    # print xtile(scorelist,lo=0, hi=1.0,width = 25)
    # print "# {0}:{1}".format(name, n/r)

170 @demo
def part6():
    r = 5
    lastera = []
    searchcount = 0
175    for klass in [ZDT1]:
        print "\n!!!!", klass.__name__
        for searcher in [sa, mws]:
            reseed()
            for k in range(r):
                Settings.sa.cooling = rand() # get variants of sa, mws
                Settings.mws.prob = rand()
                Settings.mws.max_changes = int(1000*rand())
                model = klass()
                x, lohi = searcher(model)
185                for i, f in enumerate(model.log.y):
                    temp = []
                    searchername = "mws" if searchcount else "sa"
                    label = searchername + str(k) + "%s" % i
                    temp = (model.history[sorted(model.history.keys())[-1]].log.y[i])._cach
e)
190                temp = [ float(i) for i in temp]
                temp.insert(0, str(label))
                lastera.append(temp)
                rdivDemo(lastera)
                searchcount +=1
                lastera = []
195 @demo
def testmodel():
    # model = ZDT3()
    model = Schaffer()
    depen = model.getDepen(model.generate_x())
200    print depen

    if __name__ == "__main__": eval(cmd())

205

210

```

Sep 22, 14 23:50

csc791sbse:hw4:Fu

Page 1/4

```

from __future__ import division
from log import *
import sys, random, math, datetime, time, re, pdb
sys.dont_write_bytecode = True

5
exp = math.e
sqrt = math.sqrt
sin = math.sin
10 pi = math.pi

class Model:

    def name(i):
        return i.__class__.__name__
    def setup(i):
        i.xy = Options(x = [i.generate_x()], y = [i.f1, i.f2])
        i.log = Options(x = [ Num() for _ in range(i.n)], y = [ Num() for _ in range
(i.fn)]) # hardcoded 2
        i.history = {} # hold all logs for eras
    def generate_x(i):
        x = [i.lo + (i.hi-i.lo)*random.random() for _ in range(i.n)]
        return x
    def getDepen(i, xlst):
        # y = [i.f1, i.f2]
        return sum([f(xlst) for f in i.xy.y])
    def getDepenlst(i, xlst):
        return [f(xlst) for f in i.xy.y]
    def cloneModel(i): # from Dr.Menzies'
        return i.__class__()
    def logxy(i, x):
        for val, log in zip(x, i.log.x): log += val
        y = i.getDepenlst(x)
        for val, log in zip(y, i.log.y): log += val
    def better(news, olds): # from Dr.Menzies'
    def worsed():
        return ((same ^ ¬ betterIqr) ∨
                (¬ same ∧ ¬ betterMed))
    def bettered():
        return ¬ same ∧ betterMed
    40 out = False
        for new, old in zip(news.log.y, olds.log.y):
            betterMed, same, betterIqr = new.better(old)
            # print betterMed, same, betterIqr
            # pdb.set_trace()
    45 if worsed() : return False # never any worsed
            if bettered(): out = out ∨ True # at least one bettered
        return out
    def sa_neighbor(i, old):
        p = 1/i.n
        new = old
        for j in range(len(old)):
            if random.random() < p:
                new_gen = i.generate_x()
                old[j] = new_gen[random.randint(0, i.n-1)]
    55 return old
    def mws_neighbor(i, solution):
        optimized_index = random.randint(0, len(solution)-1)
        increment = (i.hi - i.lo)/10
        temp_min = 10*(5)
    60 # print "old solution : %s" % solution
        for _ in range(10):
            solution[optimized_index] = i.lo + increment
            temp = i.norm(i.getDepen(solution))
            if temp < temp_min:
    65 temp_min = temp
    # print "new solution : %s" % solution
        return solution
    def baseline(i):
        # model = eval(model+"()")
    70 i.min = 10*(5)
        i.max = -10*(5)
        for _ in xrange(100000):

```

Sep 22, 14 23:50

csc791sbse:hw4:Fu

Page 2/4

```

        temp = i.getDepen(i.generate_x())
        if temp > i.max:
            i.max = temp
    75         if temp < i.min:
            i.min = temp
        return i.min, i.max
    def norm(i, x):
    80         e = (x - i.min)/(i.max - i.min)
        return max(0, min(e, 1)) #avoid values <0 or >1

class Control(object): # based on Dr.Menzies' codes
    def __init__(i, model):
    85         i.kmax = Settings.sa.kmax
            i.era = Settings.other.era
            i.lives = Settings.other.lives
            i.logAll = {}
            i.model = model
    90         def __call__(i, k):
            i.next(k)
        def logxy(i, results):
            both = [i.model.history, i.logAll]
            for log in both:
    95                 if ¬ i.era in i.logAll:
                    log[i.era] = i.model.cloneModel()
                for log in both:
                    log[i.era].logxy(results)
        def checkimprove(i):
    100             if len(i.logAll) ≥ 2:
                current = i.era
                before = i.era - Settings.other.era
                currentLog = i.logAll[current]
                beforeLog = i.logAll[before]
                # pdb.set_trace()
    105                 if ¬ currentLog.better(beforeLog):
                    pass
                else:
                    i.lives += 1
    110             def next(i, k):
                if k ≥ i.era:
                    i.checkimprove()
                    i.era += Settings.other.era
                    if i.lives == 0:
    115                         return True
                    else:
                        i.lives -= 1
                        return False

    120 '''Schaffer'''
    class Schaffer(Model):
        def __init__(i):
    125             i.lo = -2
                i.hi = 2
                i.n = 1
                i.fn = 2
                i.setup()
    130             def f1(i, x):
                return x[0] * x[0]
            def f2(i, x):
                return (x[0]-2) ** 2

    135 '''Fonseca'''
    class Fonseca(Model):
        def __init__(i):
            i.lo = -4
            i.hi = 4
    140             i.n = 3
                i.fn = 2
                i.setup()
            def f1(i, xlst):
                return (1 - exp**(-1 * sum([(xlst[k] - 1/sqrt(i.n))**2 for k in xrange(i.n)]
)))

```

Sep 22, 14 23:50

csc791sbse:hw4:Fu

Page 3/4

```

145 def f2(i, xlst):
    return (1 - exp**(-1 * sum([(xlst[k] + 1/sqrt(i.n))**2 for k in xrange(i.n)]
    )))

'''Kusarvs'''
class Kursawe(Model):
150 def __init__(i):
    i.lo = -5
    i.hi = 5
    i.n = 3
    i.fn = 2
155 i.setup()
    def f1(i, xlst):
        return sum([-10*exp**(-0.2 * sqrt(xlst[k]**2 + xlst[k+1]**2)) for k in xrange
e(i.n -1)])
    def f2(i, xlst):
        a = 0.8
        b = 3
160 return sum([abs(x)**a + 5*sin(x)**b for x in xlst])

'''ZDT1'''
class ZDT1(Model):
165 def __init__(i):
    i.lo = 0
    i.hi = 1
    i.n = 30
    i.fn = 2
170 i.setup()
    def f1(i, xlst):
        return xlst[0]
    def g(i, xlst):
        return (1 + 9 * (sum(xlst[1:]))/(i.n-1))
175 def f2(i,xlst):
    g1 = i.g(xlst)
    return g1*(1-sqrt(xlst[0]/g1))

'''ZDT3'''
180 class ZDT3(Model):
    def __init__(i):
        i.lo = 0
        i.hi = 1
        i.n = 30
        i.fn = 2
185 i.setup()
    def f1(i, xlst):
        return xlst[0]
    def g(i, xlst):
        return (1 + (9/(i.n-1)) * sum(xlst[1:]))
190 def h(i,f1,g):
    return (1 - sqrt(f1/g) - f1/g) * sin(10 * pi * f1)
    def f2(i, xlst):
        return i.g(xlst) * i.h(i.f1(xlst),i.g(xlst))

195 '''Viennet3'''
class Viennet3(Model):
    def __init__(i):
        i.lo = -3
        i.hi = 3
200 i.n = 2
        i.fn = 3
        i.setup1()
    def setup1(i):
        i.xy = Options(x = [i.generate_x()], y = [i.f1, i.f2, i.f3])
        i.log = Options(x = [ Num() for _ in range(i.n)], y = [ Num() for _ in range
(i.fn)]) # hardcoded 2
        i.history = {} # hold all logs for eras
    def f1(i, xlst):
        xy2 = xlst[0]**2 + xlst[1]**2
205 return 0.5* (xy2) + sin(xy2)
210 def f2(i, xlst):
    x = xlst[0]
    y = xlst[1]
    return ((3*x -2*y +4)**2/8 + (x-y+1)**2/27 + 15)

```

Sep 22, 14 23:50

csc791sbse:hw4:Fu

Page 4/4

```

215 def f3(i, xlst):
    xy2 = xlst[0]**2 + xlst[1]**2
    return (1/(xy2+1) - 1.1* exp**(-xy2))

```

Sep 23, 14 11:37

csc791sbse:hw4:Fu

Page 1/2

```

from __future__ import division
import sys, random, math
from base import *
from a12 import *
5 sys.dont_write_bytecode = True

''' All these are based on Dr.Menzies' tricks ^ sample codes'''

10 class Log():
    def __init__(i, tolog = []):
        i._cache, i.n, i._report = [], 0, None
        i.setup()
        map(i.__iadd__, tolog)
15 def __iadd__(i, tolog):
    if tolog == None: return tolog
    i.n += 1
    updated = False
    if len(i._cache) < Settings.other.keep:
20         i._cache += [tolog]
        updated = True
    else:
        if rand() <= Settings.other.keep/i.n:
            i._cache[int(rand()*Settings.other.keep)] = tolog
25         updated = True
    if updated:
        i._report = None
        i.updateLoHi(tolog)
        return i
30 def has(i):
    if i._report == None:
        i._report = i.report()
        return i._report

35 class Num(Log):
    def setup(i):
        i.lo = 10**5
        i.hi = -10**5
    def updateLoHi(i,x):
40         i.lo = min(i.lo, x)
        i.hi = max(i.hi, x)
    def median(i):
        n = len(i._cache)
        p = n//2
45         if (n % 2) : return i._cache[p]
        q = p + 1
        q = max(0, min(q,n))
        return (i._cache[p] + i._cache[q])/2
    def better(new,old):
50         "better if (1)less median or (2)same and less iqr"
        t = Settings.other.a12
        betterIqr = new.has().iqr < old.has().iqr
        new.lessp = False
        if new.lessp:
55             betterMed = new.has().median >= old.has().median
            same = a12(old._cache, new._cache) <= t
        else:
            betterMed = new.has().median <= old.has().median
            same = a12(new._cache, old._cache) <= t
60         return betterMed, same, betterIqr
    def report(i):
        sortedCache = sorted(i._cache)
        n = len(sortedCache)
        return Options(
65             median = i.median(),
            iqr = sortedCache[int(n*0.75)-int(n*0.5)],
            lo = i.lo,
            hi = i.hi)

70 @demo
def demoNum():
    for size in [16,32, 64,128, 256]:
        Settings.other.keep = size

```

Sep 23, 14 11:37

csc791sbse:hw4:Fu

Page 2/2

```

log = Num()
75 for x in xrange(100000): log +=x
    print size, ":", log.has().median

80 if __name__ == "__main__": eval(cmd())

```

Sep 23, 14 11:37

csc791sbse:hw4:Fu

Page 1/1

```
from __future__ import division
import sys, random, math
from sk import *
sys.dont_write_bytecode = True
5
def rdiv8():
    rdivDemo([
        ["orange", 5.7, 5.1, 4.6, 4.9, 4.6, 6],
        ["apple", 5.3, 5.6, 4.8, 5.2, 4.7, 6],
10    ["pears", 3.9, 4.4, 4.7, 4.5, 4.8, 5.0, 6],
        ["strawberry", 0.33, 0.30, 0.31, 0.36, 0.34, 6]
    ])
if __name__ == "__main__": eval("rdiv8()")
```

Sep 22, 14 23:59

csc791sbse:hw4:Fu

Page 1/2

```

from __future__ import division
import sys, random, math, datetime, time, re
sys.dont_write_bytecode = True

5 rand= random.random

class Options: #Thanks for Peter Norvig's trick
    def __init__(i, **d): i.__dict__.update(d)

10 Settings = Options(sa = Options(kmax = 1000,
                                baseline = 1000,
                                score = {},
                                cooling = 0.5),
15 mws = Options(threshold = 0.0001,
                max_tries = 20,
                max_changes = 1000,
                prob = 0.25,
                score = {}
20 ),
                other = Options(keep = 128,
                                era = 50,
                                lives = 3,
                                al2 = [0.56, 0.64, 0.71][0],
                                repeats = 1))

25 def atom(x):
    try : return int(x)
    except ValueError:
        try : return float(x)
    except ValueError : return x

30 def cmd(com="demo('-h')"):
    "Convert command line to a function call."
    if len(sys.argv) < 2: return com
    def strp(x): return isinstance(x, basestring)
    def wrap(x): return "%s"%x if strp(x) else str(x)
    words = map(wrap, map(atom, sys.argv[2:]))
    return sys.argv[1] + '(' + ','.join(words) + ')'

40 def demo(f=None, cache=[]):
    def doc(d):
        return '#'+d.__doc__ if d.__doc__ else ""
    if f == '-h':
        print '# sample demos'
        for n,d in enumerate(cache):
            print '%3s)' % (n+1), d.func_name, doc(d)
    elif f:
        cache.append(f)
    else:
        s='|'+ '='*40 + '\n'
        for d in cache:
            print '\n==|', d.func_name, s, doc(d), d()
        return f

55 def reseed():
    seed = 1
    return random.seed(seed)

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

def printlook(f):
    def wrapper(*lst): #tricks from Dr.Menzies
65 ShowDate = datetime.datetime.now().strftime
        print "\n###", f.__name__, "##" * 50
        print "##", ShowDate("%Y-%m-%d %H:%M:%S")
        beginTime = time.time()
        x = f(*lst)
        endTime = time.time()
        print "\n" + ("-"*60)
        dump(Settings, f.__name__)
        print "\n# Runtime: %.3fsecs" % (endTime-beginTime)

```

Sep 22, 14 23:59

csc791sbse:hw4:Fu

Page 2/2

```

    return x # return the searcher name and the results
    return wrapper

75 def dump(d, searchname, lvl = 0): # tricks from Dr. Menzies
    d = d if isinstance(d, dict) else d.__dict__
    callableKey, line, gap = [], "", "" * lvl
    80 for k in sorted(d.keys()):
        val = d[k]
        if isinstance(val, (dict, Options)):
            callableKey += [k]
        else:
            85 #if callable(val):
                # val = val.__name__
                line += (" {0}:{1}".format(k, val))
            print gap + line
            for k in callableKey:
                90 if k == searchname or k == "other":
                    print gap + (" {0}:{1}".format(k, "options"))
                    dump(d[k], lvl+1)

```

Sep 22, 14 23:50

csc791sbse:hw4:Fu

Page 1/1

```

from __future__ import division
import sys, random, math, datetime, time, re, pdb
sys.dont_write_bytecode = True

5
def pairs(lst):
    last=lst[0]
    for i in lst[1:]:
        yield last,i
10    last = i

def xtile(lst,lo=0,hi=0.001, width = 50,
        chops=[0.1 ,0.3,0.5,0.7,0.9],
        marks=["-"," "," "," ","-"," "],
15    bar="|",star="*",show="%3s"):
    """The function _xtile_ takes a list of (possibly)
    unsorted numbers and presents them as a horizontal
    xtile chart (in ascii format). The default is a
    contracted _quintile_ that shows the
20    10,30,50,70,90 breaks in the data (but this can be
    changed- see the optional flags of the function).
    """
    # ordered_list = sorted(lst) # Dr.Menzies tricks
    # lo = min(lo, ordered_list[0])
    # hi = max(hi, ordered_list[-1])
    # showNumbers = [ ordered_list[int(percent * len(lst))] for percent in chops]
    # # print showNumbers
    # showMarks = [" "] * width
    # def find_index (x):
    #     return int(width*float((x-lo))/(hi-lo))
    # markIndex = [find_index(i) for i in showNumbers]
    # for i in range(width):
    #     if i in range(markIndex[0],markIndex[1]+1) or i in range(markIndex[-2],mar
kIndex[-1]+1):
    #         showMarks[i] = "-"
35    # #print showMarks
    # showMarks[int(width * 0.5)] = "/"
    # showMarks[find_index(ordered_list[int(len(lst)*0.5))]] = "*"
    # return " ".join(showMarks) + ", ".join([show %str(round(i,3)) for i in showN
umbers])
    def pos(p) : return ordered[int(len(lst)*p)]
    def place(x) :
        return int(width*float((x - lo))/(hi - lo+0.00001))
    def pretty(lst) :
        return ','.join([show % x for x in lst])
    ordered = sorted(lst)
    lo = min(lo,ordered[0])
    hi = max(hi,ordered[-1])
    what = [pos(p) for p in chops]
    where = [place(n) for n in what]
    out = [" "] * width
50    for one,two in pairs(where):
        for i in range(one,two):
            out[i] = marks[0]
            marks = marks[1:]
    out[int(width/2)] = bar
55    out[place(pos(0.5))] = star
    return ''.join(out) + ", " + pretty(what)

60 def Demo() :
    import random
    random.seed(1)
    # nums = [random.random()*2 for _ in range(100)]
    #nums = [0.011,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01]
    nums = [0,0.1,0.1,0.6,0.4,0.1,0.9,0.1,0.1, 3]
65    line = ''*26+'='*23
    print ('%29s,%3s,%3s,%3s,%3s' % ('10%', '30%', '50%', '70%', '90%'))+'\n'+line
    print xtile(nums,lo=0,hi=1.0,width=25,)

70
if __name__ == "__main__": Demo()

```


Sep 23, 14 11:37

csc791sbse:hw4:Fu

Page 1/1

```

from __future__ import division
import sys
sys.dont_write_bytecode = True
from base import *

5
def _ab12():
    def a12slow(lst1, lst2):
        more = same = 0.0
        for x in sorted(lst1):
            for y in sorted(lst2):
10                 if x==y :
                     same += 1
                     elif x > y :
                         more += 1
15         return (more + 0.5*same) / (len(lst1)*len(lst2))
        random.seed(1)
        l1 = [random.random() for x in range(5000)]
        more = [random.random()*2 for x in range(5000)]
        l2 = [random.random() for x in range(5000)]
20        less = [random.random()/2.0 for x in range(5000)]
        for tag, one, two in [("lless", l1, more),
                              ("lmore", more, less), ("same", l1, l2)]:
            t1 = msecs(lambda : a12(l1, less))
            t2 = msecs(lambda : a12slow(l1, less))
25        print "\n", tag, "\n", t1, a12(one, two)
            print t2, a12slow(one, two)

def a12(lst1, lst2):
    " " "how often is lst1 often more than y in lst2?
    assumes lst1 nums are meant to be greater than lst2" " "
30    def loop(t, t1, t2):
        while t1.m < t1.n ^ t2.m < t2.n:
            h1 = t1.l[t1.m]
            h2 = t2.l[t2.m]
            h3 = t2.l[t2.m+1] if t2.m+1 < t2.n else None
35            if h1 > h2:
                t1.m += 1; t1.gt += t2.n - t2.m
            elif h1 == h2:
                #if h3 and gt(h1, h3) < 0: original bugs
                if h3 ^ h1 > h3:
40                    t1.gt += t2.n - t2.m - 1
                    t1.m += 1; t1.eq += 1; t2.eq += 1
            else:
                t2, t1 = t1, t2
45        return t.gt*1.0, t.eq*1.0
    #-----
    lst1 = sorted(lst1, reverse=True)
    lst2 = sorted(lst2, reverse=True)
    n1 = len(lst1)
    n2 = len(lst2)
50    t1 = Options(l=lst1, m=0, eq=0, gt=0, n=n1)
    t2 = Options(l=lst2, m=0, eq=0, gt=0, n=n2)
    gt, eq = loop(t1, t1, t2)
    return gt/(n1*n2) + eq/2/(n1*n2)
55
if __name__ == "__main__": eval(cmd())

```