C200 Programming Assignment № 9

Dr. M.M. Dalkilic
Computer Science
School of Informatics, Computing, and Engineering

Indiana University, Bloomington, IN, USA

December 2, 2023

Introduction

In this homework, you'll work on translating critical thinking to programming. This homework is a bit less intense than the last one! :) Since this is the last homework, you'll be reading some on your own and experimenting.

Make sure to commit and push your project and modules by **11:05PM EDT, Saturday December 9, 2023.**

Problem 1: Random Walk

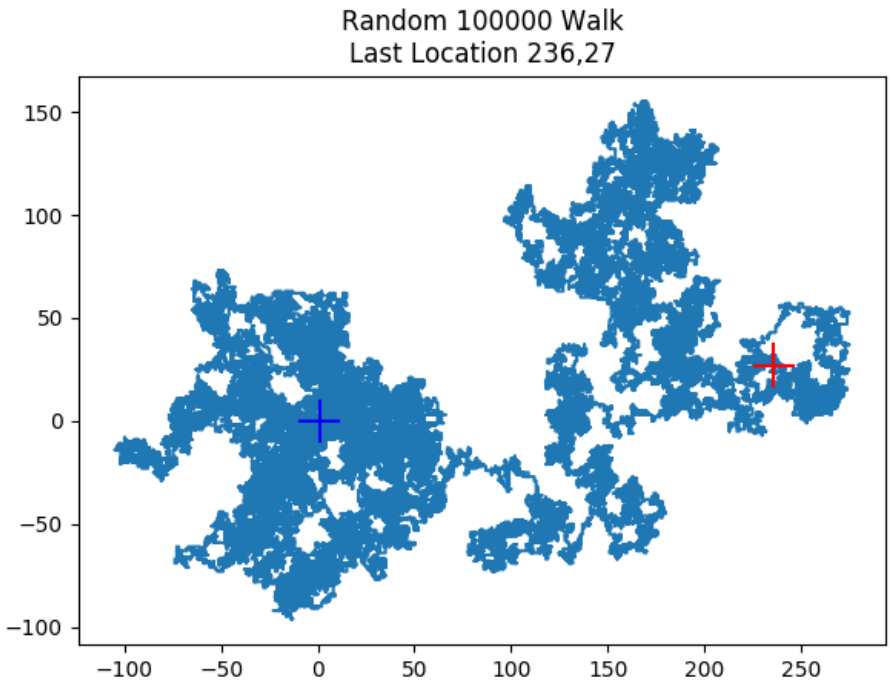

Figure 1: The blue cross is where we started and the red cross is where we ended.

A random walk is a *stochastic process*. A stochastic process is a series of values that are not determined functionally, but probabilistically. The random walk is supposed to describe an inebriated person who, starting from the bar, intends to walk home, but because of intoxication instead randomly takes single steps either forward or backward, left or right. The person has no memory of any steps taken, so theoretically, the person shouldn't move too far from where he or she starts. Random walks are used to model many phenomena, like the size of the web or changes in financial instruments. We will model a 2D random walk with two arrays x and y where x represents moving left or right and y represents forward or backward. The index i will represent the step and x[i],y[i] will represent the location at step i. So, for i=0, we have x[0],y[0] (starting place). Using random we choose from the list [1,2,3,4]. If the value is one then we move right from the previous x position:

```
1  x[i] = x[i-1] + 1
2  y[i] = y[i-1]
```

If the value is two, then we move left from the previous x position:

```
1  x[i] = x[i-1] - 1
2  y[i] = y[i-1]
```

If the value is three, we move up from the previous y position:

```
x[i] = x[i-1]
y[i] = y[i-1] + 1
```

And when the value is four, we move down from the previous y position:

```
x[i] = x[i-1]
y[i] = y[i-1] - 1
```

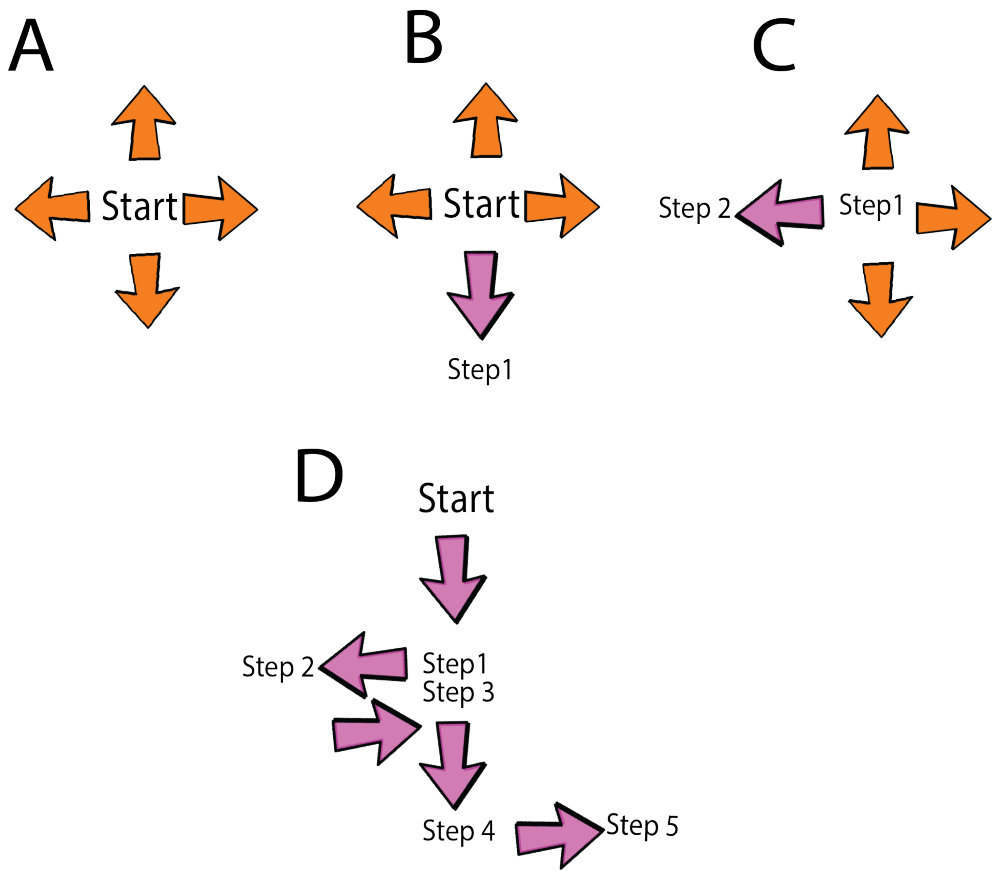

Figure 2: The first few steps of a random walk.

Here is another way to describe this:

$$step(0) = 0 \tag{1}$$

$$step_i(n) = \begin{cases} \text{left} & step(n-1),\ i = 1 \\ \text{right} & step(n-1),\ i = 2 \\ \text{up} & step(n-1),\ i = 3 \\ \text{down} & step(n-1),\ i = 4 \end{cases} \tag{2}$$

Session Output

```
Number of Steps: 100000
Figure similar to Figure 1
```

Deliverable for Programming Problem 1: Random Walk

- Complete the `step` function. (Template code has been provided)
- These are random walks, meaning the outputs will be different.
- Experiment with different numbers of steps. Do you see any number of steps that seems to always move significantly away from the start? (Put this answer in a comment at the top of the code)
- Put your code into a new module named **randomwalk.py**

Problem 2: Imaginary Numbers & Fractals

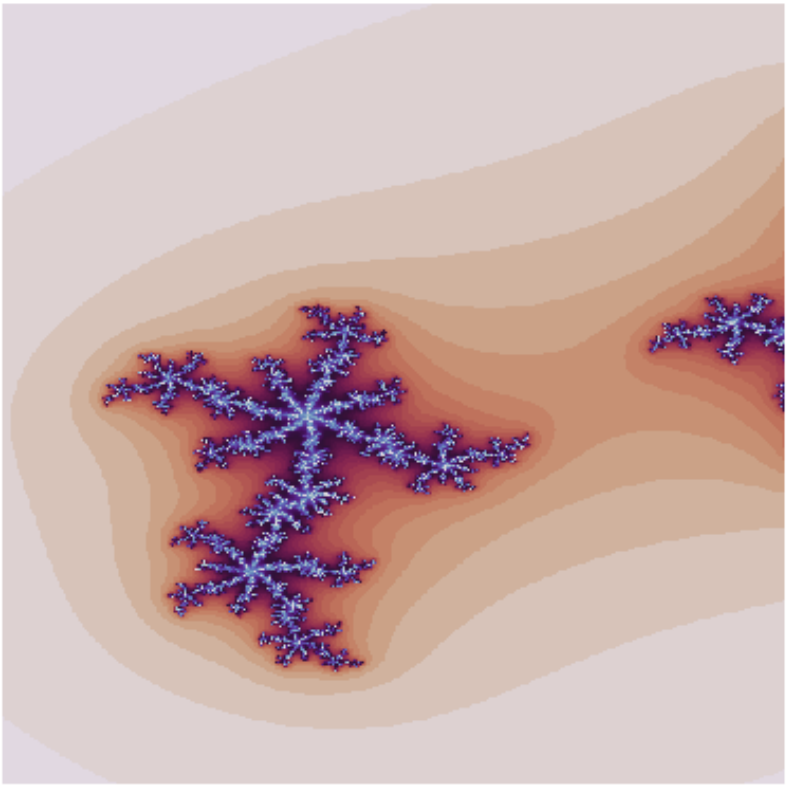

Figure 3: Erie silver snowflakes on an alien planet of sand.

In class we discussed how complex numbers are a pair of real numbers, $a, b \in \mathbb{R}$ and i is the solution to $i^2 = -1$. We called a the *real* part and b the imaginary part. We found that Python has a complex class too, but engineers chose j for i. In this problem, we're extending our complex number class called Complex_Number to include a few more operations:

- modulus overloading __abs__. The modulus of a complex number $x \pm yj$ is $\sqrt{x^2 + y^2}$
- power overload __pow__ that returns the power of a number (see `https://docs.python.org/3/library/operator.html?highlight=__pow__`). For example,

```
1 >>> x = complex(1,2)
2 >>> for i in range(6):
3 ...     x**i
4 ...
5 (1+0j)
6 (1+2j)
```

```
 7  (-3+4j)
 8  (-11-2j)
 9  (-7-24j)
10  (41-38j)
11  >>>
```

We have included +,* as shown in class. When the class is complete, you'll have a complex number class that you'll use for what I created that looks like snowflakes for this code:

```
1  def mandelbrot(z,MAX_ITER):
2      n = 0
3      lz = Complex_Number(0,0)
4      c = Complex_Number(-0.1, 0.65)
5      while abs(z) <= 10 and n < MAX_ITER:
6          z = z**2 + lz + c
7          lz = z
8          n += 1
9      return n/MAX_ITER
```

As you can see from the code, we need modulus, addition, and power. When run, this code:

```
1   x1 = Complex_Number(1,2)
2   y1 = complex(1,2)
3   print(x1,y1)
4   print(abs(x1),abs(y1))
5      for i in range(6):
6          print(x1**i,y1**i)
```

produces

```
1  (1+2j) (1+2j)
2  2.23606797749979 2.23606797749979
3  (1+0j) (1+0j)
4  (1+2j) (1+2j)
5  (-3+4j) (-3+4j)
6  (-11-2j) (-11-2j)
7  (-7-24j) (-7-24j)
8  (41-38j) (41-38j)
```

and the fractal image. As you can see, I modified the __str__ to mimic Python's print.

Deliverable for Problem 2: Complex Numbers

- Complete the Complex_Number class
- Do *NOT* use any built-in packages for complex numbers. You are making your own `Complex` class. (including cmath, complex())

Problem 3: SQL

In class we were introduced to SQL and the relational model. You will have a great deal of freedom with this problem. Create a table called `Weather` with attributes `City`, `State`, `High`, `Low` and populate it with the data shown in Table 1.

Weather			
City	State	High	Low
Phoenix	Arizona	105	90
Tucson	Arizona	101	92
Flag Staff	Arizona	105	90
San Diego	California	77	60
Albuquerque	New Mexico	80	72
Nome	Alaska	64	-54

Table 1: Relation Weather and tuples

.

I've made equivalent list comprehension on this iterable:

```
data = [('Phoenix', 'Arizona', 105, 90),
            ('Tucson', 'Arizona', 101, 92),
            ('Flag Staff', 'Arizona', 105, 90),
            ('San Diego', 'California', 77, 60),
            ('Albuquerque', 'New Mexico', 80, 72),
            ('Nome', 'Alaska', 64 ,-54)]
```

You should read about these SQL functions (**NOTE**: SQL functions, not Python functions): `count()`, `sum()`, `min()`, `max()` as well as "group by" and "in". We've given the answer to query 1 (i.e. MYSQL query). The results of these queries are shown in the output. The following section shows the results obtained by using python and list comprehension–you are required to implement the MYSQL operations in python to implement these queries.

1. Select all the tuples (Query 1) This is an example–observe where the SQL is and replicate this for the remainder of the queries

```
print("Query 1")
def query1(db_cursor):
    temp = []
    for i in db_cursor.execute("SELECT * FROM Weather"):
        temp.append(i)
    return temp
print(query1(my_cursor))
print("List Comprehension: ", data)
```

2. Select all the tuples where the High temperature is less than 80 (Query 2)

```
print("Query 2")
#def query2(db_cursor):
#    temp = []
#    for i in db_cursor.execute("Some fantastic SQL"):
#        temp.append(i)
#    return temp
#print(query1(my_cursor))
print("List Comprehension: ", [d for d in data if d[2] < 80 ])
```

3. Select All the cities where the low temperature is strictly greater than the Low of Albuquerque – you cannot use the number 72.0 in the query (Query 3)

```
print("Query 3")
#def query3(db_cursor):
#    temp = []
#    for i in db_cursor.execute("Some fantastic SQL"):
#        temp.append(i)
#    return temp
#print(query3(my_cursor))
x =[d[0] for d in data if d[3] > [d[3] for d in data if d[0] == 'Albuquerque'][0]]
print("List Comprehension: ",x)
```

4. Select the city and temperature with the smallest low temperature (Query 4)

```
print("Query 4")
#def query4(db_cursor):
#    temp = []
#    for i in db_cursor.execute("Some fantastic SQL"):
#        temp.append(i)
#    return temp
#print(query4(my_cursor))
print("List Comprehension: ",[(d[0],d[3]) for d in data if d[3] in (sorted(data, key = lambda x:x[3])[0])])
```

5. Select the city temperature with the largest high temperature–since there are two, both cities should be returned. (Query 5)

```
print("Query 5")
#def query5(db_cursor):
#    temp = []
```

```
#     for i in db_cursor.execute("Some fantastic SQL"):
#         temp.append(i)
#     return temp
#print(query5(my_cursor))
print("List Comprehension: ",[(d[0],d[2]) for d in data if d[2] in
    (sorted(data, key = lambda x:x[2],reverse=True)[0])])
```

6. Display the *average* High and Low temperatures–you are not allowed to use Avg() (Query 6)

```
print("Query 6")
#def query6(db_cursor):
#     temp = []
#     for i in db_cursor.execute("Some fantastic SQL"):
#         temp.append(i)
#     return temp
#print(query6(my_cursor))

print("List Comprehension: ", [(sum([d[2] for d in data])/len(data
    ),sum([d[3] for d in data])/len(data))])
```

7. Give the counts of cities by their Low temperatures (Query 7)

```
print("Query 7")
#def query7(db_cursor):
#     temp = []
#     for i in db_cursor.execute("Some fantastic SQL"):
#         temp.append(i)
#     return temp
#print(query7(my_cursor))

print([(i,list(map((lambda x: x[3]),data)).count(i)) for i in set(
    map((lambda x: x[3]),data))])
```

Output

```
Query 1
('Phoenix', 'Arizona', 105.0, 90.0)
('Tucson', 'Arizona', 101.0, 92.0)
('Flag Staff', 'Arizona', 105.0, 90.0)
('San Diego', 'California', 77.0, 60.0)
('Albuquerque', 'New Mexico', 80.0, 72.0)
('Nome', 'Alaska', 64.0, -54.0)
List Comprehension:
[('Phoenix', 'Arizona', 105, 90),
 ('Tucson', 'Arizona', 101, 92),
 ('Flag Staff', 'Arizona', 105, 90),
 ('San Diego', 'California', 77, 60),
 ('Albuquerque', 'New Mexico', 80, 72),
 ('Nome', 'Alaska', 64, -54)]

Query 2
('San Diego', 'California', 77.0, 60.0)
('Nome', 'Alaska', 64.0, -54.0)
List Comprehension:
[('San Diego', 'California', 77, 60),
 ('Nome', 'Alaska', 64, -54)]

Query 3
('Phoenix',)
('Tucson',)
('Flag Staff',)
List Comprehension:  ['Phoenix', 'Tucson', 'Flag Staff']
```

Output

```
Query 4
('Nome', -54.0)
List Comprehension:  [('Nome', -54)]

Query 5
('Phoenix', 105.0)
('Flag Staff', 105.0)
List Comprehension:  [('Phoenix', 105), ('Flag Staff', 105)]

Query 6
(88.66666666666667, 58.333333333333336)
List Comprehension:  [(88.66666666666667, 58.333333333333336)]

Query 7
(-54.0, 1)
(60.0, 1)
(72.0, 1)
(90.0, 2)
(92.0, 1)
List Comprehension:  [(72, 1), (-54, 1), (60, 1), (90, 2), (92, 1)]
```

Delivearble for Problem 3: SQL

- Write the MYSQL code using python for queries 1-7.
- Although this is never done, for the purposes of making the problem easier, we will make sure to remove the table and repopulate it everytime so we don't have to comment out portions.

Programming partners

wgurley@iu.edu, kvpriede@iu.edu
davgourl@iu.edu, maklsmit@iu.edu
mohiambu@iu.edu, dyashwar@iu.edu
leokurtz@iu.edu, apathma@iu.edu
schinitz@iu.edu, scotbray@iu.edu
ameydesh@iu.edu, gepearcy@iu.edu
escolber@iu.edu, aselki@iu.edu
mrcoons@iu.edu, ajtse@iu.edu
jc168@iu.edu, patedev@iu.edu
cgkabedi@iu.edu, liwitte@iu.edu
spgreenf@iu.edu, ragmahaj@iu.edu
coopjose@iu.edu, rnschroe@iu.edu

makinap@iu.edu, etprince@iu.edu
phjhess@iu.edu, cmarcuka@iu.edu
dblackme@iu.edu, madymcsh@iu.edu
ceub@iu.edu, woodsky@iu.edu
ridbhan@iu.edu, cnyarko@iu.edu
vkommar@iu.edu, rosenbbj@iu.edu
brhint@iu.edu, wlyzun@iu.edu
egoldsto@iu.edu, ap79@iu.edu
zacbutle@iu.edu, mszczas@iu.edu
maudomin@iu.edu, tarturnm@iu.edu
althart@iu.edu, aranjit@iu.edu
bencho@iu.edu, thnewm@iu.edu
mrfehr@iu.edu, patel89@iu.edu
jakchap@iu.edu, tpandey@iu.edu
grafe@iu.edu, reedkier@iu.edu
sg40@iu.edu, fmahamat@iu.edu
dkkosim@iu.edu, rtrammel@iu.edu
jwcase@iu.edu, ruska@iu.edu
mkames@iu.edu, jcn1@iu.edu
jdemirci@iu.edu, dwo@iu.edu
leegain@iu.edu, rorymurp@iu.edu
nfarhat@iu.edu, ksadiq@iu.edu
mkleinke@iu.edu, krbpatel@iu.edu
daminteh@iu.edu, ltmckinn@iu.edu
skunduru@iu.edu, iperine@iu.edu
greenpat@iu.edu, awsaunde@iu.edu
aakindel@iu.edu, aptheria@iu.edu
jabbarke@iu.edu, blswing@iu.edu
anrkram@iu.edu, jpochyly@iu.edu
nmcastan@iu.edu, myeralli@iu.edu
bellcol@iu.edu, utwade@iu.edu
brownset@iu.edu, wtatoole@iu.edu
tfreson@iu.edu, mnimmala@iu.edu
apbabu@iu.edu, snyderjk@iu.edu
oakinsey@iu.edu, mz24@iu.edu
delkumar@iu.edu, emisimps@iu.edu
colrkram@iu.edu, wtubbs@iu.edu
aberkun@iu.edu, pp31@iu.edu
kaneai@iu.edu, coenthom@iu.edu
agrevel@iu.edu, sasayini@iu.edu
fkeele@iu.edu, chrimanu@iu.edu

saecohen@iu.edu, ansakrah@iu.edu
deombeas@iu.edu, gsilingh@iu.edu
gmhowell@iu.edu, jactrayl@iu.edu
sakalwa@iu.edu, aveluru@iu.edu
alscarr@iu.edu, lmadiraj@iu.edu
spgerst@iu.edu, evataylo@iu.edu
ek37@iu.edu, vmungara@iu.edu
laburkle@iu.edu, apavlako@iu.edu
hawkjod@iu.edu, annaum@iu.edu
sydecook@iu.edu, asultano@iu.edu
twfine@iu.edu, vyeruba@iu.edu
seangarc@iu.edu, megapaul@iu.edu
jtbland@iu.edu, btasa@iu.edu
nihanas@iu.edu, qshamsid@iu.edu
coopelki@iu.edu, mzagotta@iu.edu
clearle@iu.edu, abiparri@iu.edu
adhuria@iu.edu, smremmer@iu.edu
sfuneno@iu.edu, anajmal@iu.edu
milhavil@iu.edu, sahaan@iu.edu
lawmat@iu.edu, arirowe@iu.edu
khannni@iu.edu, ammulc@iu.edu
aaamoako@iu.edu, avraya@iu.edu
fkanmogn@iu.edu, ijvelmur@iu.edu
kjj6@iu.edu, orrostew@iu.edu
cuizek@iu.edu, rvinzant@iu.edu
kdembla@iu.edu, wardjohn@iu.edu
cfampo@iu.edu, patekek@iu.edu
rl29@iu.edu, masmatth@iu.edu
jhar@iu.edu, anemlunc@iu.edu
gandhira@iu.edu, leolin@iu.edu
josespos@iu.edu, lvansyck@iu.edu
amkhatri@iu.edu, linjaso@iu.edu
nolakim@iu.edu, asaokho@iu.edu
jacobben@iu.edu, lpelaez@iu.edu
nfelici@iu.edu, surapapp@iu.edu
migriswo@iu.edu, bcmarret@iu.edu
aroraarn@iu.edu, dernguye@iu.edu
quecox@iu.edu, drsnid@iu.edu
liansia@iu.edu, aditpate@iu.edu
cpkerns@iu.edu, nsatti@iu.edu
edfran@iu.edu, jneblett@iu.edu

oeichenb@iu.edu, ysanghi@iu.edu
alelefeb@iu.edu, cltran@iu.edu
lflenoy@iu.edu, ism1@iu.edu
loggreen@iu.edu, muyusuf@iu.edu
blacount@iu.edu, rpoludas@iu.edu
matgarey@iu.edu, jarlmint@iu.edu
garcied@iu.edu, audtravi@iu.edu
wilcusic@iu.edu, clscheum@iu.edu
eakanle@iu.edu, joshroc@iu.edu
ajeeju@iu.edu, jaslnu@iu.edu
jacklapp@iu.edu, vpolu@iu.edu
simadams@iu.edu, ryarram@iu.edu
anlego@iu.edu, samyuan@iu.edu
bkante@iu.edu, fshamrin@iu.edu
achordi@iu.edu, cstancom@iu.edu
jdc6@iu.edu, mmarotti@iu.edu
ahavlin@iu.edu, mehtriya@iu.edu
wanjiang@iu.edu, amanocha@iu.edu
ethbrock@iu.edu, aamathew@iu.edu
maxklei@iu.edu, dukthang@iu.edu
ckdiallo@iu.edu, benprohm@iu.edu
ryanbren@iu.edu, vrradia@iu.edu
tychid@iu.edu, wilsori@iu.edu
tcconnol@iu.edu, lukastef@iu.edu
hk120@iu.edu, sahimann@iu.edu
ejhaas@iu.edu, impofujr@iu.edu
nokebark@iu.edu, ntuhl@iu.edu
howamatt@iu.edu, justyou@iu.edu
skatiyar@iu.edu, jnzheng@iu.edu
sgaladim@iu.edu, gavsteve@iu.edu
lpfritsc@iu.edu, clmcevil@iu.edu
adiyer@iu.edu, tolatinw@iu.edu
bencalex@iu.edu, patelsak@iu.edu
flynncj@iu.edu, mjroelle@iu.edu
arnadutt@iu.edu, sahishah@iu.edu
aketcha@iu.edu, rwan@iu.edu
ohostet@iu.edu, ntorpoco@iu.edu
saganna@iu.edu, brayrump@iu.edu
huhasan@iu.edu, ao9@iu.edu
ajgrego@iu.edu, jwmullis@iu.edu
kekchoe@iu.edu, tzuyyen@iu.edu

zguising@iu.edu, ir1@iu.edu
daxbills@iu.edu, jtsuter@iu.edu
ethickma@iu.edu, pricemo@iu.edu
laharden@iu.edu, nmr1@iu.edu
nharkins@iu.edu, jwember@iu.edu
rcaswel@iu.edu, zshamo@iu.edu
swconley@iu.edu, nniranj@iu.edu
jkielcz@iu.edu, pravulap@iu.edu
stkimani@iu.edu, savebhat@iu.edu
dja1@iu.edu, voram@iu.edu
mdonato@iu.edu, wwtang@iu.edu
tchapell@iu.edu, erschaef@iu.edu
hermbrar@iu.edu, kamaharj@iu.edu
hh35@iu.edu, lmeldgin@iu.edu
spdamani@iu.edu, cialugo@iu.edu
austdeck@iu.edu, thomps16@iu.edu
efritch@iu.edu, aledminc@iu.edu
hamac@iu.edu, tangtom@iu.edu
avulas@iu.edu, isaramir@iu.edu
arklonow@iu.edu, majtorm@iu.edu
johnguen@iu.edu, cmvanhov@iu.edu
kaihara@iu.edu, antando@iu.edu
cannan@iu.edu, perezand@iu.edu
evacoll@iu.edu, mmunaf@iu.edu
ovadeley@iu.edu, jsadiq@iu.edu
bcdutka@iu.edu, nlippman@iu.edu
allencla@iu.edu, nichojop@iu.edu
bjdahl@iu.edu, rt11@iu.edu
aaragga@iu.edu, jlzhao@iu.edu
jwdrew@iu.edu, rafir@iu.edu
mbeigie@iu.edu, aidnschi@iu.edu
earuland@iu.edu, schwajaw@iu.edu
maladwa@iu.edu, jomeaghe@iu.edu
nbernot@iu.edu, lqadan@iu.edu
diebarro@iu.edu, epautsch@iu.edu
howardbw@iu.edu, emgward@iu.edu
kapgupta@iu.edu, dsummit@iu.edu
alchatz@iu.edu, jwetherb@iu.edu
apchavis@iu.edu, keasandl@iu.edu
mbrockey@iu.edu, reddyrr@iu.edu
adwadash@iu.edu, cjwaller@iu.edu

ruqchen@iu.edu, emluplet@iu.edu
dce@iu.edu, gszopin@iu.edu
joehawl@iu.edu, mveltri@iu.edu
gcopus@iu.edu, giomayo@iu.edu
ag69@iu.edu, bmpool@iu.edu
abellah@iu.edu, deturne@iu.edu
micahand@iu.edu, gavilleg@iu.edu
masharre@iu.edu, owysmit@iu.edu
krisgupt@iu.edu, wtrucker@iu.edu
jhhudgin@iu.edu, sezinnkr@iu.edu
mdiazrey@iu.edu, gs29@iu.edu
mdoxsee@iu.edu, adsize@iu.edu
ddrotts@iu.edu, samrile@iu.edu
fdonfrio@iu.edu, jwu6@iu.edu
mwclawso@iu.edu, asteini@iu.edu
caegrah@iu.edu, pateishi@iu.edu
marganey@iu.edu, skp2@iu.edu
fu7@iu.edu, gmpierce@iu.edu
keswar@iu.edu, bdyiga@iu.edu
ayoajayi@iu.edu, pw18@iu.edu
lcoveney@iu.edu, hasiddiq@iu.edu
jdgonzal@iu.edu, asidda@iu.edu
kraus@iu.edu, nrizvi@iu.edu
agawrys@iu.edu, crmoll@iu.edu