C200 Programming Assignment №5

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

Indiana University, Bloomington, IN, USA

March 24, 2022

Introduction

Please, please begin this homework immediately. **Due Date: 10:59 PM, Thursday, March, 31, 2022**. Submit your work before the deadline and remember to add, commit and push often.

Problem 1: Entropy

In this problem, we will be calculating what's called *Entropy*. Entropy is used in ML, AI, vision, finance, *etc.*

For this problem, we are only working with finite probabilities. We can think of probabilities as a list of numbers $p_0, p_1, \ldots, p_n$ such that

$$\begin{aligned} p_i &\geq 0, \quad i = 0, 1, \ldots, n && (1)\\ 1 &= p_0 + p_1 + p_2 + \cdots + p_n && (2)\\ &= \sum_{i=0}^{n} p_i && (3) \end{aligned}$$

We can make a list (we'll assume the items are of the same type and immutable) into a probability. For example, consider a list `x = ["a","b","a","c","c","a"]`.

1. gather the items uniquely (*hint: dictionary*)
2. count each time the item occurs
3. create a new list of the `count/len(dictionary)`

The list of probabilities would be y = [3/6,1/6,2/6] (a's probability is 3/6, b's probability is 1/6 and c's probability is 2/6). Observe that if you add up these numbers, they will sum to one.

We still need to show you *how* to calculate entropy:

$$\begin{aligned} entropy &= -(p_0 \log_2(p_0) + p_1 \log_2(p_1) + \cdots + p_n \log_2(p_n)) && (4)\\ entropy(y) &= -(\frac{3}{6}\log_2(3/6) + \frac{1}{6}\log(1/6) + \frac{2}{6}\log_2(2/6)) && (5)\\ &= -(.5(-1.0) + 0.17(-2.58) + .33(-1.58)) && (6)\\ &= 1.46 && (7) \end{aligned}$$

Because of continuity arguments we treat $\log_2(0) = 0$. Python's math module will correctly state this math error, so you'll **have to simply add 0 if the probability is 0**.

Deliverables Problem 1

- Take a non-empty list of objects and calculate the entropy.
- You **cannot** use the `count` function.
- Complete the functions for this problem.
- You must round to two decimal places.

Problem 2: Run of 1s

In this problem, you'll take a list of 0s and 1s as an input. Your program will output the *longest* sequential run of 1s (no 0s encountered).

Output function.py

```
L([1, 0, 1, 0, 1, 0])
1

L([0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0])
2

L([1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1])
4
```

Deliverables Problem 2

- Complete the program.

Problem 3: Nine of Cattails

It's known that if you add the digits of any number, and that it eventually equals nine, that the original number is divisible by nine. Here is an example:

$$39789 \quad -> \quad 3+9+7+8+9=36 \tag{8}$$

Add the remaining digits

$$36 \quad -> \quad 3+6=9 \tag{9}$$

We are left with only one digit which is 9 and hence the number 39789 is divisible by 9 (Infact, 9 * 4421 = 39789).

So we can see that the number 39789 is indeed divisible by 9 by following our method. Write a program that determines if a number is by divisible by nine by using this method. You *cannot* simply divide by 9 and return True! You must continually add the numbers until a single digit is left; that digit will either be 9 or something else.

Caution: Remember that 'sum' is actually a built-in function in Python so we suggest that you store the results of addition in a variable named other than 'sum' to prevent errors or failures during testing.

sum = a + b + c (this is not a good practise because 'sum' is a reserved keyword in Python), rather
my_sum = a + b + c (this is safe to do)

Deliverables Problem 3

- Complete the functions **following** the algorithm described above.
- You can neither use the % sign in the program nor divide directly by 9.

Problem 4: Recursion, Tail Recursion, Generators

In this problem, you'll implement the recursive function, tail recursive function, and generator. The tail recursive and generator function names have _t() and _g() appended to their names. (Hint: that a generator just yields without storing anything).

$$
\begin{aligned}
p(0) &= 10000 && (10)\\
p(n) &= p(n-1) + 0.02p(n-1) && (11)\\
p\_t(n, v=0) & \quad \text{tail recursive} && (12)\\
p\_g() & \quad \text{generator} && (13)
\end{aligned}
$$

$$
\begin{aligned}
c(1) &= 9 && (14)\\
c(n) &= 9c(n-1) + 10^{n-1} - c(n-1) && (15)\\
c\_t(n, v=0) & \quad \text{tail recursive} && (16)\\
c\_g() & \quad \text{generator} && (17)
\end{aligned}
$$

$$
\begin{aligned}
d(0) &= 1 && (18)\\
d(n) &= 3d(n-1) + 1 && (19)\\
d\_t(n, v=1) & \quad \text{tail recursive} && (20)\\
d\_g() & \quad \text{generator} && (21)
\end{aligned}
$$

Programming Problem 4: Recursion

- Complete each of the functions.

Problem 5: Potato-Smith Sort

Potato-Smith sort (or “potato”) for short is an improved insertion sort. The algorithm works by sending insertation sort larger sublists. We’ll work through an example. Assume that a list is given x = [9,8,7,6,5,4,3,2,1,0] and initially size = 2.

We use insertion sort (hereby referred to as is) on sublists of size then we double the size.

```
is(x[0:2]) + is(x[2:4]) + is(x[4:6]) + is(x[6:8]) + is(x[8:])
[8, 9, 6, 7, 4, 5, 2, 3, 0, 1]
size = size * 2
is(x[0:4]) + is(4:8) + is(x[8:])
[6, 7, 8, 9, 2, 3, 4, 5, 0, 1]
size = size * 2
is(x[0:8]) + is(x[8:])
[2, 3, 4, 5, 6, 7, 8, 9, 0, 1]
size = size * 2
is(x[0:])
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

You should be able to figure out the stopping critera using size and length of the list.

Programming Problem 5: Potato

- Complete the Potato function.
- You must write the insertion sort from lecture.
- You obviously cannot use anything else to sort.

Problem 6: Maximal Value

In this problem, you're given a list of numbers. The function should return a list of three values (in this same order) namely, the interval start, interval stop and the value of the greatest sum. For example if the list is: x = [7, -9, 5, 10, -9, 6, 9, 3, 3, 9] then the greatest interval is from x[2:10] giving 36:

$$sum(x[2, 10]) \quad = \quad 5 + 10 - 9 + 6 + 9 + 3 + 3 + 9 = 36 \tag{22}$$

The function should return a list of values i.e., [2, 10, 36]

For this problem you are allowed to use sum(lst) which returns the sum of a non-empty list of numbers.

Note: Remember that 'sum' is a built-in function and you should not use it as a normal variable, for example:

sum = a + b + c (this is not a good practise) because this example store the result of the addition in a variable called sum but Python has already reserved the word 'sum' for a function.

Programming Problem 6: Maximal Value

- Complete the function
- You are allowed to use the sum function

Problem 7: Coins

Sometimes problems are a lot simpler than you initially imagine. Write a function that takes an amount tot in dollars and returns a list:

$$[q, d, n, p]$$

where q is the number of quarters, d is the number of dimes, n is the number of nickels, and p is the number of pennies and

```
tot = q*.25 + d*.10 + n*.05 + p*.01
q+d+n+p is a minimum
```

For example, $2.24 = [8, 2, 0, 1] = [.25 \times 8 + .10 \times 2 + .05 \times 0 + .01 \times 4]$. Thus, your list should contain the fewest total coins possible that equals the dollar amount. You must only return non-negative integers!

Deliverables Programming Problem 7

- Complete the function

Paired Programming Partners

mabdayem@iu.edu, egmorley@iu.edu
ahnabrah@iu.edu, mbarrant@iu.edu
adamsjf@iu.edu, gabradle@iu.edu
dadeyeye@iu.edu, jhlazar@iu.edu
cmaguila@iu.edu, shadoshi@iu.edu
ahmedrr@iu.edu, samuwagn@iu.edu
malshama@iu.edu, akaushal@iu.edu
olalbert@iu.edu, owinston@iu.edu
nalemanm@iu.edu, yudsingh@iu.edu
faysalza@iu.edu, ehallor@iu.edu
rnameen@iu.edu, kbburnet@iu.edu
svamin@iu.edu, sothor@iu.edu
jaygul@iu.edu, rbajaj@iu.edu
bbacso@iu.edu, cl101@iu.edu
cbalbuen@iu.edu, cbylciw@iu.edu
ikbanist@iu.edu, lancswar@iu.edu
zsbanks@iu.edu, mathchen@iu.edu
tymbarre@iu.edu, cadwilco@iu.edu
dcblakle@iu.edu, jwrohn@iu.edu
timbogun@iu.edu, ddahodu@iu.edu
mlboukal@iu.edu, sskauvei@iu.edu
logbrads@iu.edu, mooralec@iu.edu
lburrola@iu.edu, pheile@iu.edu
aidcarli@iu.edu, apoellab@iu.edu
dcaspers@iu.edu, mlumbant@iu.edu
joecool@iu.edu, rpdeady@iu.edu
ccoriag@iu.edu, chlzhang@iu.edu
gcruzcor@iu.edu, eliserr@iu.edu
jacuau@iu.edu, jpenrigh@iu.edu
cadelaga@iu.edu, aramo@iu.edu
emdelph@iu.edu, mdtanner@iu.edu
edepke@iu.edu, eluthra@iu.edu
shrdesai@iu.edu, kjwalapu@iu.edu
eeconomo@iu.edu, ibnash@iu.edu
ereilar@iu.edu, msronan@iu.edu
kamdelmo@iu.edu, ndvanbur@iu.edu
jaespin@iu.edu, nps1@iu.edu
mfanous@iu.edu, rwan@iu.edu
nfarhat@iu.edu, kevko@iu.edu

jayfish@iu.edu, vvictori@iu.edu
sydfoste@iu.edu, astrouf@iu.edu
ethfrago@iu.edu, divpatel@iu.edu
fraustom@iu.edu, nfrische@iu.edu
gaoxinl@iu.edu, mppan@iu.edu
gillenj@iu.edu, tclady@iu.edu
ggivan@iu.edu, johnslia@iu.edu
bgloor@iu.edu, jsm13@iu.edu
noahgrah@iu.edu, cy30@iu.edu
halejd@iu.edu, yangyuc@iu.edu
ejharms@iu.edu, dylomall@iu.edu
jchobbs@iu.edu, camitong@iu.edu
binyhu@iu.edu, ssalama@iu.edu
jthurd@iu.edu, jk130@iu.edu
silmudee@iu.edu, evewalsh@iu.edu
srimmadi@iu.edu, njindra@iu.edu
aj110@iu.edu, sasaluja@iu.edu
yjan@iu.edu, sj110@iu.edu
gjarrold@iu.edu, powelchr@iu.edu
mjerrell@iu.edu, rlmcdani@iu.edu
yuljiao@iu.edu, gkarnuta@iu.edu
cjohanns@iu.edu, yl181@iu.edu
fkanmogn@iu.edu, hdwatter@iu.edu
phklein@iu.edu, ryou@iu.edu
jtkrug@iu.edu, remarche@iu.edu
wlegear@iu.edu, rosavy@iu.edu
jleverty@iu.edu, mmansoo@iu.edu
linweix@iu.edu, hnasar@iu.edu
lopezis@iu.edu, weidzhen@iu.edu
vimadhav@iu.edu, apapaioa@iu.edu
gmanisca@iu.edu, luilmill@iu.edu
pmanolis@iu.edu, sowvemul@iu.edu
tymath@iu.edu, srpothir@iu.edu
sahmir@iu.edu, owenaj@iu.edu
jamoya@iu.edu, ashankwi@iu.edu
bolabanj@iu.edu, gkyoung@iu.edu
daparent@iu.edu, lzinn@iu.edu
perkcaan@iu.edu, gtutton@iu.edu
shevphil@iu.edu, anniye@iu.edu
sprabhak@iu.edu, evtomak@iu.edu
raia@iu.edu, mvincen@iu.edu

vramkum@iu.edu, chsand@iu.edu
mattroac@iu.edu, lzinn@iu.edu
asaokho@iu.edu, amystaff@iu.edu
mschauss@iu.edu, jjwelp@iu.edu
ansiva@iu.edu, actoney@iu.edu
grtalley@iu.edu, dazamora@iu.edu