Programming Interview

Name:	Yao Jiaoyang
Date:	May 18, 2020
Job applied for:	Openings in Artificial Intelligence (AI) for Cancer research
Tick the appropriate box:	o Citizen
	o Permanent Resident
	● Student Pass
	o Others, please specify:

Question 1

Assumption: As shown in the example, we assume that each entry of the matrix contains it's row number (start with 1). For situations with multiple solutions, only one solution is required to be given.

Analysis: Since only Right and Down operations are allowed, a fixed number of steps $m + n - 2$ are needed to go across the corner of a $m \times n$ matrix. The $m - 1$ Down operations guarantee that every row number has at least one contribution to the total sum, and $n - 1$ Right operations contribute to the other part of the total sum.

Suppose the target sum is T and the amount of Right operations performed on i-th row is represented by c_i. Then the problem target becomes acquiring the solution of the following linear system:

$$\begin{cases} \sum_{i=1}^{m} i + \sum_{i=1}^{n} c_i \cdot i = T \\ \sum_{i=1}^{n} c_i \qquad = n - 1 \end{cases}$$

Only 2 linear constraints are given to find n unknown variable, thus there might be multiple solution when $n \geq 3$. For example, in the given example where $m = n = 4, sum = 16$, the solution can be either RDRDRD or DRRRDD. In my program, only one possible solution is given as output.

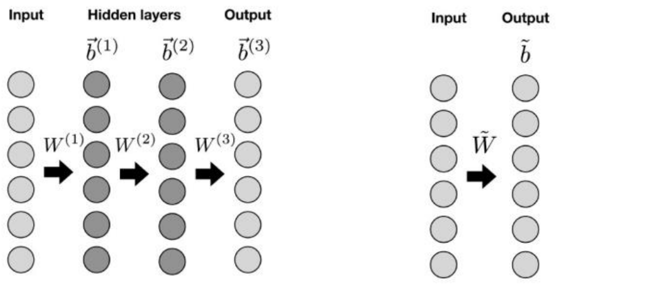
The sum value of the operation sequence is limited in the range of:

$$\sum_{i=1}^{m} i + (n-1) \sim \sum_{i=1}^{m} i + (n-1)m$$

For m=9, n=9 situation, the range is [53, 117]. All the required sum values are achievable.

For m=90000, n=100000 situation, the range is [234999, 9000045000]. 87127231192 is unachievable (thus it is not involved in the code) and 5994891682 is achievable.

Question 2



(a) Network 1: MLP with multiple hidden layers. (b) Network 2: MLP with no hidden layers.

From

$$\vec{a}^{(1)} = W^{(1)}\vec{a}^{(0)} + \vec{b}^{(1)}$$

We have

$$\begin{aligned}\vec{a}^{(2)} &= W^{(2)}\vec{a}^{(1)} + \vec{b}^{(2)}\\ &= W^{(2)}\left(W^{(1)}\vec{a}^{(0)} + \vec{b}^{(1)}\right) + \vec{b}^{(2)}\\ &= W^{(2)}W^{(1)}\vec{a}^{(0)} + W^{(2)}\vec{b}^{(1)} + \vec{b}^{(2)}\end{aligned}$$

And

$$\begin{aligned}\vec{a}^{(3)} &= W^{(3)}\vec{a}^{(2)} + \vec{b}^{(3)}\\ &= W^{(3)}\left(W^{(2)}W^{(1)}\vec{a}^{(0)} + W^{(2)}\vec{b}^{(1)} + \vec{b}^{(2)}\right) + \vec{b}^{(3)}\\ &= W^{(3)}W^{(2)}W^{(1)}\vec{a}^{(0)} + W^{(3)}W^{(2)}\vec{b}^{(1)} + W^{(3)}\vec{b}^{(2)} + \vec{b}^{(3)}\end{aligned}$$

Thus we have

$$\vec{a}^{(3)} = \widetilde{W}\vec{a}^{(0)} + \tilde{b}$$

where $\widetilde{W} = W^{(3)}W^{(2)}W^{(1)}$, $\tilde{b} = W^{(3)}W^{(2)}\vec{b}^{(1)} + W^{(3)}\vec{b}^{(2)} + \vec{b}^{(3)}$

Question 3

The model with 2 hidden layers (3-64-14-1) is implemented with PyTorch. The training procedure involves MSE loss function and SGD optimizer. For more implementation details please refer to the source code.

Question 4

4-connectivity is implemented for this problem.

To solve this problem, a **recursive** function “find_cnct” is defined in my solution. This function firstly determines whether input location in the given matrix is a “1” position. If it is “1”, the position will be labeled by the given label and the 4-connected positions are passed to “find_cnct” function recursively. If it is “0”, the function returns directly without any modification on the given matrix.

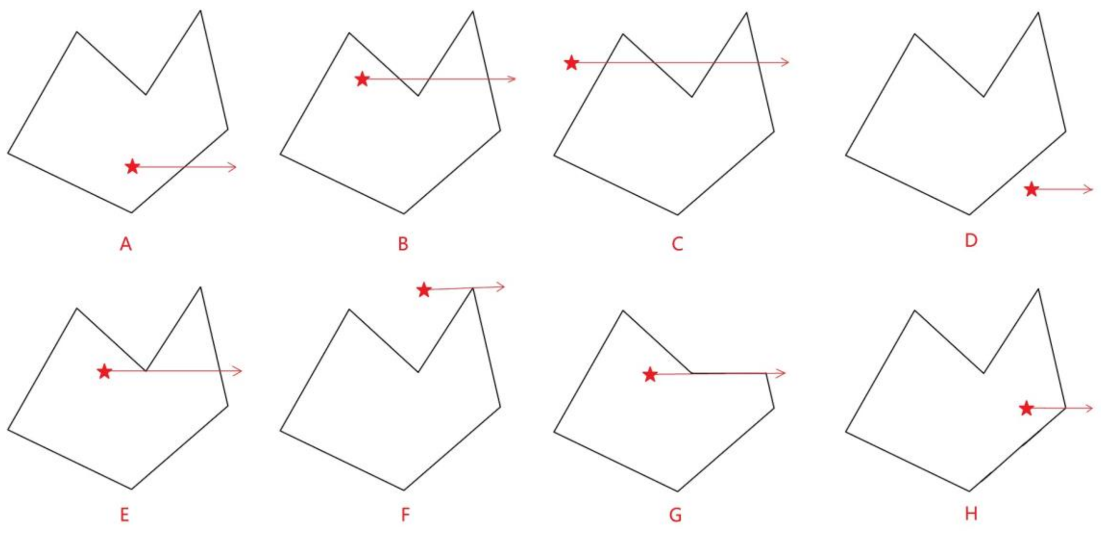
Since the “1”s in the input and “1”s in the output have different meanings, the input “1”s are converted to “-1” to avoid ambiguity. For each position, it is labeled only when it is “-1”, otherwise it is already colored or invalid.

Question 5

Not Answered

Question 6

One lemma state that if we create a half-line based on Q, the amount of intersection point between the half-line and polygon will indicate whether it is in or out of the polygon. As shown below where we use the right direction half-line for judgement, when **even number of intersections** are observed, the point is out of the polygon (e.g. C, D), while when **odd number of intersections** are observed, the point is in the polygon (e.g. A, B).



Some special situations are discussed in example EFGH. When the half-line intersects the polygon on a corner, the intersection number should be count as 0 or 2 if the two sides of the corner are at the same side of the half-line (e.g. E, F). Otherwise the intersection number should be count as 1 (e.g. H). If the half-line coincides with one side of the polygon, the intersection number count as 1.

A point on the edge is considered “in” the polygon. Other implementation details please refer to the source code.

Question 7

2-dimansion:

The coordinate-to-index conversion:

$$I = x_1 + x_2 L_1$$

The index-to-coordinate conversion:

$$\begin{cases} x_1 = I \; mod \; L_1 \\ x_2 = I \mid L_1 \end{cases}$$

where “mod” is the modulo operation and “|” is the integer division operation.

d-dimension:

For the d-dimension situations, The matrix with shape $(L_1, L_2, L_3 \dots L_d)$ have coordinates represented by $(x_1, x_2, x_3 \dots x_d)$. Since the indexing priority for each dimension is in a sequenced order, the first n dimensions can be indexed separately without affecting later dimensions. For example, the 3-dimension coordinate-to-index conversion can be derived by:

$$\begin{aligned} I &= x_{1\sim2} + x_3 L_{1\sim2} \\ &= I_{1\sim2} + x_3 L_1 L_2 \\ &= x_1 + x_2 L_1 + x_3 L_1 L_2 \end{aligned}$$

where $x_{1\sim2}$, $I_{1\sim2}$, $L_{1\sim2}$ represent the corresponding parameters when considering dimension 1 and 2 as one dimension. Note that $x_{1\sim2} \equiv I_{1\sim2}$

And the 3-dimension index-to-coordinate conversion:

$$\begin{cases} x_{1\sim2} = I \; mod \; L_{1\sim2} \\ x_3 = I \mid L_{1\sim2} \end{cases} \xrightarrow{L_{1\sim2}=L_1L_2} \begin{cases} x_{1\sim2} = I \; mod \; (L_1L_2) \\ x_3 = I \mid (L_1L_2) \end{cases}$$

then

$$\begin{cases} x_1 = x_{1\sim2} \; mod \; L_1 \\ x_2 = x_{1\sim2} \mid L_1 \\ x_3 = I \mid (L_1L_2) \end{cases} \xrightarrow{x_{1\sim2}=I \; mod \; (L_1L_2)} \begin{cases} x_1 = I \; mod \; (L_1L_2) mod \; L_1 \\ x_2 = I \; mod \; (L_1L_2) \mid L_1 \\ x_3 = I \mid (L_1L_2) \end{cases}$$

$$\xrightarrow{properties \; of \; mod \; \& \mid operation} \begin{cases} x_1 = I \; mod \; L_1 \mid 1 \\ x_2 = I \; mod \; (L_1L_2) \mid L_1 \\ x_3 = I \; mod \; (L_1L_2L_3) \mid (L_1L_2) \end{cases}$$

Thus we can find the expression pattern for d-dimension case. To make the expression more consistent, we assume $L_0 = 1$.

The coordinate-to-index conversion is:

$$\begin{aligned} I &= x_1 + x_2 L_1 + x_3(L_1 L_2) + \cdots + x_d(L_1 L_2 \ldots L_{d-1}) \\ &= \sum_{i=1}^{d}\left(x_i \prod_{j=0}^{i-1} L_j\right) \end{aligned}$$

And the index-to-coordinate conversion:

$$x_n = I\ mod\ \left(\prod_{i=1}^{n} L_i\right) | \left(\prod_{j=0}^{n-1} L_j\right)$$

Some special cases for index-to-coordinate conversion need to be clarified. When $n = 1$, $x_1 = I\ mod\ L_1 \mid L_0 = I\ mod\ L_1$. When $n = d$, $x_d = I\ mod\ \left(\prod_{i=1}^{d} L_i\right) \middle| \left(\prod_{j=0}^{d-1} L_j\right) = I \mid \left(\prod_{j=0}^{d-1} L_j\right)$.