به نام خدا

دانشگاه تهران - دانشگاه صنعتی خواجه نصیرالدین طوسی تهران

دانشکده مهندسی برق و کامپیوتر

شبکهٔ عصبی مک کلاچ-پیتز

نام و نام خانوادگی	محمدجواد احمدی
شمارهٔ دانشجویی	۴۰۱۰۰۰۸۶

فهرست مطالب

فهرست تصاویر

فهرست جداول

پرسش۱. شبکهٔ عصبی Mcculloch-Pitts

۱ ماشین متناهی قطعی (DFA)

توضیح پوشهٔ کدهای Mcculloch-Pitts

کدهای مربوط به این قسمت، علاوه بر پوشهٔ محلی کدها در این لینک آورده شده است.

۱.۱ پاسخ قسمت الف

جدول انتقال حالتی که مطابق توضیحات سوال و متناسب با شبکهٔ نورون‌ها به صورت زیر است:

جدول ۱: نمایش جدید جدول انتقال حالت

IN			OUT		
A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	0	1	0
1	1	0	1	1	1
1	1	1	1	1	1

۲.۱ پاسخ قسمت ب

در این قسمت براساس نام‌گذاری انجام‌شده در جدول ۱، به دست می‌آوریم که هر خروجی معادل چه ترکیبی از ورودی‌هاست. در جدول کارنو، روش SOP و روش POS دو روش مختلف برای تبدیل یک عبارت منطقی به یک عبارت استاندارد می‌باشند. در روش SOP (Sum of Products)، عبارت منطقی به صورت جمع بیشترین (OR) حاصل ضرب کوچکترین (AND) های ممکن تشکیل می‌شود. به عبارت دیگر، ابتدا برای هر حالت ممکن مقادیر ورودی، یک خروجی (AND) از متغیرهای ورودی ایجاد می‌شود. سپس تمام این خروجی‌ها با هم جمع می‌شوند. در روش POS (Product of Sums)، عبارت منطقی به صورت ضرب کوچکترین (AND) حاصل جمع بیشترین (OR) های ممکن تشکیل می‌شود. به عبارت دیگر، ابتدا برای هر حالت ممکن مقادیر ورودی، یک جمع (OR) از متغیرهای ورودی ایجاد می‌شود. سپس تمام این جمع‌ها با هم ضرب می‌شوند. استفاده از هر

دو روش، به دلیل وجود یک تعداد محدودی ورودی، به راحتی قابل انجام است. با ارائهٔ این توضیحات، ابتدا به سراغ خروجی D می‌رویم. جداول و محاسبات مربوط به این خروجی به شرح زیر است (منطق SOP):

	$\bar{C}$	C		$\bar{C}$	C
$\bar{A}\cdot\bar{B}$	0	0	$\bar{A}\cdot\bar{B}$	0	1
$\bar{A}\cdot B$	1	0	$\bar{A}\cdot$B	2	3
A.B	1	1	A.B	6	7
A.B	1	0	A.$\bar{B}$	4	5

$(2,6)$	B.$\bar{C}$
$(4,6)$	A.$\bar{C}$
$(6,7)$	A.B

(۱)

با منطق POS اما داریم:

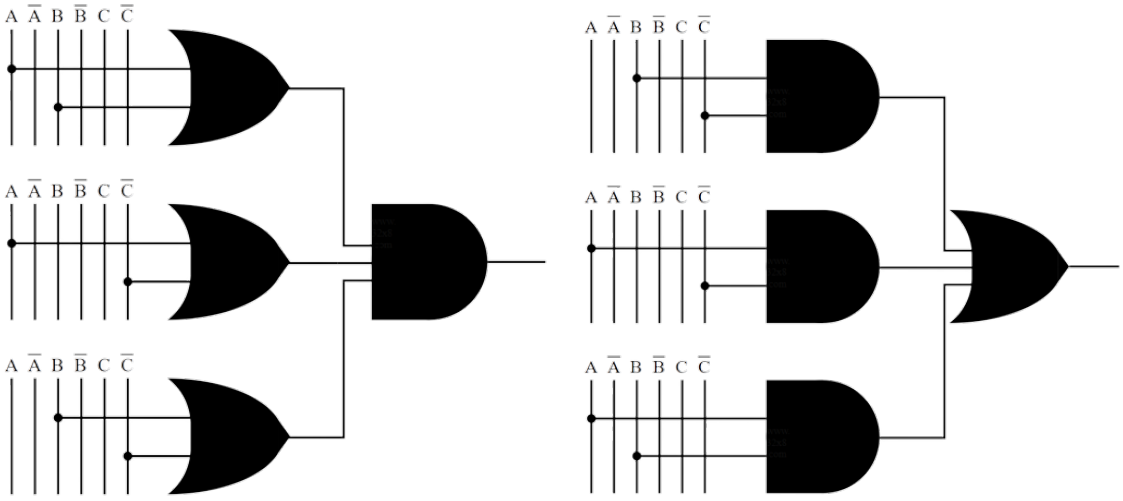
	$\bar{C}$	C		$\overline{\text{C}}$	C
$\bar{A}\cdot\bar{B}$	0	0	$\overline{\text{A}}\cdot\overline{\text{B}}$	0	1
$\bar{A}\cdot B$	1	0	$\overline{\text{A}}\cdot$B	2	3
A.B	1	1	A.B	6	7
A.$\bar{B}$	1	0	A.B	4	5

$(0,1)$	$\overline{\text{A}}\cdot\overline{\text{B}}$
$(1,3)$	$\overline{\text{A}}\cdot$C
$(1,5)$	$\overline{\text{B}}\cdot$C

(۲)

در نتیجه داریم (شکل ۱):

$$D = \text{B}\bar{\text{C}} + \text{A}\bar{\text{C}} + \text{AB}, \qquad D = (A+B)(A+\bar{C})(B+\bar{C}) \tag{۳}$$



(ب) POS (ا) SOP

شکل ۱: نحوهٔ ترکیب ورودی‌ها برای تولید D.

در ادامه به سراغ خروجی E می‌رویم. جداول و محاسبات مربوط به این خروجی به شرح زیر است (منطق SOP):

	$\bar{C}$	C
$\bar{A}\cdot\bar{B}$	0	1
$\bar{A}\cdot B$	0	1
A.B	1	1
A.$\bar{B}$	1	1

	$\bar{C}$	C
$\bar{A}\cdot\bar{B}$	0	1
$\bar{A}\cdot B$	2	3
A.B	6	7
A.$\bar{B}$	4	5

$(1,3,5,7)$	C
$(4,5,6,7)$	A

(۴)

با منطق POS اما داریم:

	$\bar{C}$	$\bar{C}$
$\bar{A}\cdot\bar{B}$	0	1
$\bar{A}\cdot B$	0	1
A.B	1	1
A.$\bar{B}$	1	1

	$\overline{C}$	C
$\overline{AB}$	0	1
$\overline{A}\cdot B$	2	3
A · B	6	7
A · $\overline{B}$	4	5

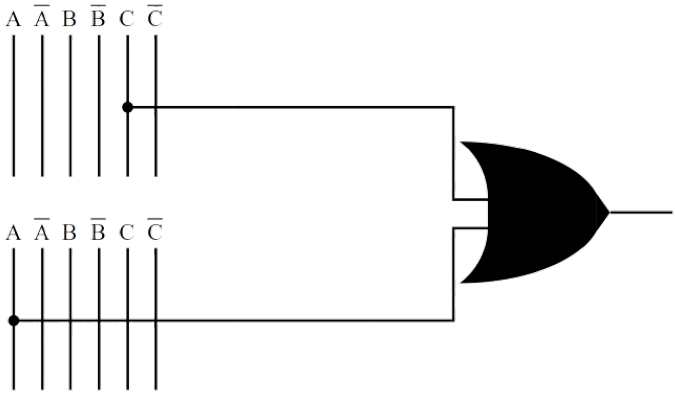
$(0,2)$	$\bar{A}\cdot\bar{C}$

(۵)

در نتیجه داریم (شکل ۲):

$$E = C + A \tag{۶}$$

در پایان هم به سراغ خروجی F می‌رویم. جداول و محاسبات مربوط به این خروجی به شرح زیر است (منطق SOP):



شکل ۲: نحوهٔ ترکیب ورودی‌ها برای تولید E.

	$\bar{C}$	C
$\bar{A}\cdot\bar{B}$	0	0
$\bar{A}\cdot B$	0	0
A.B	1	1
A.$\bar{B}$	1	0

	$\bar{C}$	C
$\overline{A}\,\bar{B}$	0	1
A.B	2	3
A.B	6	7
A.$\bar{B}$	4	5

$(4,6)$	A.$\bar{C}$
$(6,7)$	A.B

(۷)

با منطق POS اما داریم:

	$\bar{C}$	C
$\bar{A}\cdot\bar{B}$	0	0
$\bar{A}\cdot B$	0	0
A.B	1	1
$A\cdot\bar{B}$	1	0

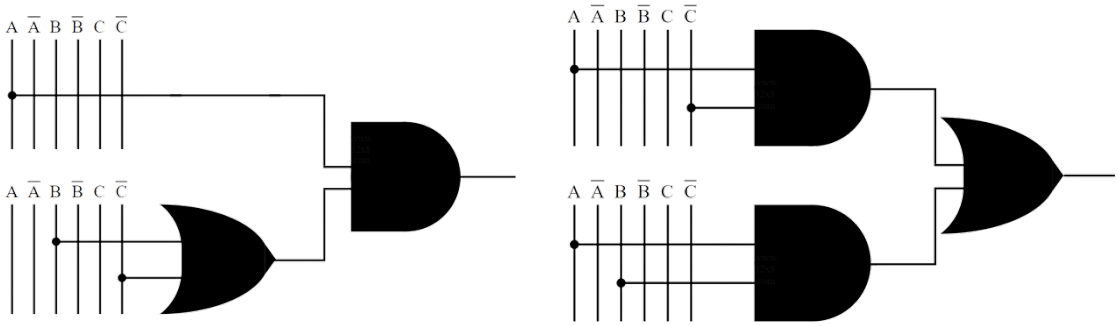
	$\overline{\mathrm{C}}$	C
$\overline{\mathrm{A}}\cdot\overline{\mathrm{B}}$	0	1
$\overline{\mathrm{A}}\cdot\mathrm{B}$	2	3
A.B	6	7
$\mathrm{A}.\bar{B}$	4	5

$(0,1,2,3)$	$\overline{\mathrm{A}}$
$(1,5)$	$\overline{\mathrm{B}}\cdot\mathrm{C}$

(۸)

در نتیجه داریم (شکل ۳):

$$F = A\bar{C} + AB \qquad F = (A)\left(B + \bar{C}\right) = (A)\overline{(C\bar{B})} \tag{۹}$$



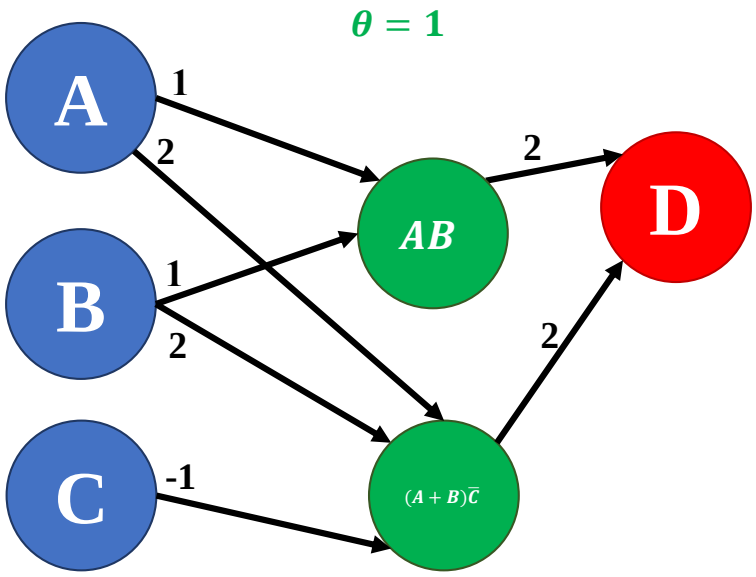
POS (ب) SOP (آ)

شکل ۳: نحوهٔ ترکیب ورودی‌ها برای تولید F.

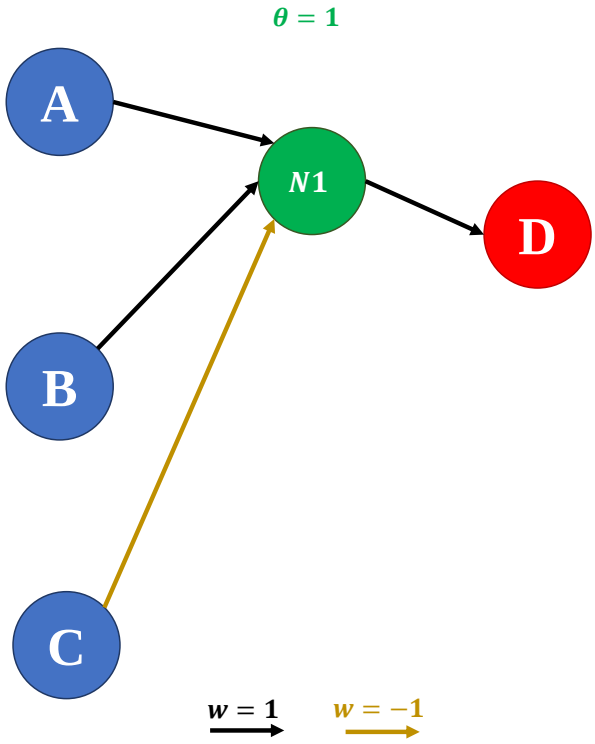
حال شبکهٔ مربوط به هر خروجی را تشکیل می‌دهیم. ابتدا برای خروجی D و بر اساس رابطهٔ ۳ شبکهٔ آورده‌شده در شکل ۴ را تشکیل می‌دهیم. می‌توانستیم حالت دیگر را نیز در نظر بگیریم اما در آن صورت به تعداد نورون بیش‌تری نیاز بود. هم‌چنین می‌توان حالت بهینهٔ نشان‌داده‌شده در شکل ۵ را برای خروجی D در نظر گرفت. هرچند بهینگی مدنظر این قسمت از سوال نیست. در ادامه برای خروجی E و بر اساس رابطهٔ ۶ شبکهٔ آورده‌شده در شکل ۶ را تشکیل می‌دهیم. در پایان، برای خروجی F و بر اساس رابطهٔ ۹ شبکهٔ آورده‌شده در شکل ۷ را تشکیل می‌دهیم. می‌توانستیم حالت دیگر را نیز در نظر بگیریم اما در آن صورت به تعداد نورون بیش‌تری نیاز بود. برای جلوگیری از تولید عدد دو برای F ناچاریم که شبکه را به شکل ۸ تغییر دهیم.

۳.۱ پاسخ قسمت ج

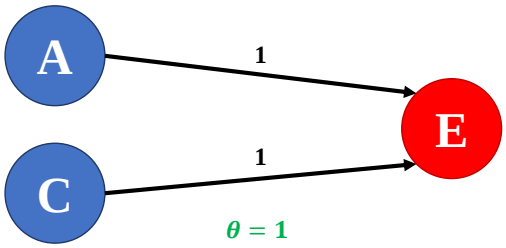
با ادغام شبکه‌های بهینهٔ درنظرگرفته‌شده در شکل ۴، شکل ۶ و شکل ۷، شبکهٔ نمایش‌داده‌شده در شکل ۹ را به دست می‌آوریم. با این وجود حالت بهینه و پاسخگوی این ادغام با سعی‌وخطا در پاسخ قسمت د به دست آمده است که آن را در شکل ۱۰ نمایش داده‌ایم (با فرض اینکه خروجی دارای تابع فعال‌ساز است و یا جنس صفر و یک آن از پیش تعریف شده و به محض دریافت یک، یک می‌شود). با این حال اگر اتصال مستقیم ورودی و خروجی را مجاز ندانیم، یک نورون میانی برای خروجی E هم در نظر می‌گیریم و نتیجه به صورتی خواهد بود که در شکل ۱۱ نمایش داده شده است. هرچند در کد می‌توان این محدودیت را دور زد؛ اما به لحاظ منطق تئوری نیاز به استفاده از ۴ نورون داریم تا خروجی بیش‌تر از یک نشود (آستانه در تمام حالات یک است).



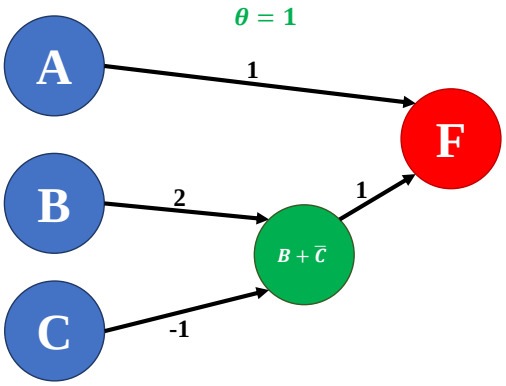
شکل ۴: شبکه‌ای که می‌تواند برای خروجی D در نظر گرفته شود.



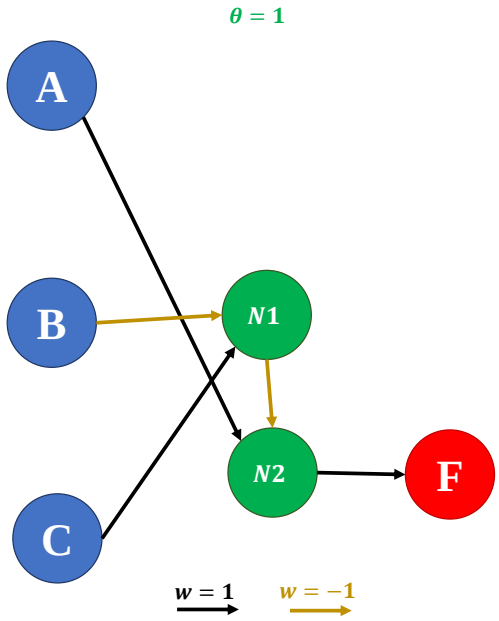
شکل ۵: شبکه‌ای بهینه که می‌تواند برای خروجی D در نظر گرفته شود.



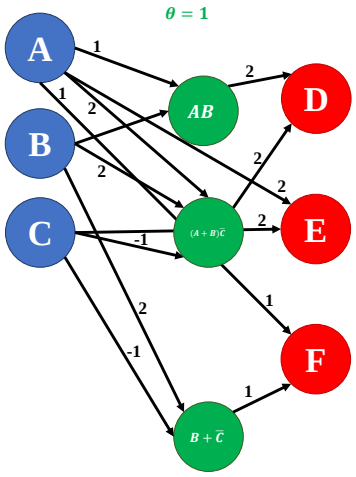
شکل ۶: شبکه‌ای که می‌تواند برای خروجی E در نظر گرفته شود.



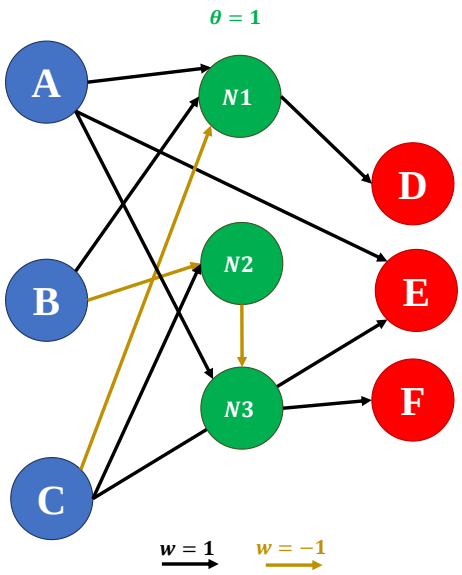
شکل ۷: شبکه‌ای که می‌تواند برای خروجی F در نظر گرفته شود (دارای ایراد جزئی. درستش در شکل ۸).



شکل ۸: شبکه‌ای که می‌تواند برای خروجی F در نظر گرفته شود.



شکل ۹: شبکهٔ ادغام‌شده (دارای ایراد جزئی. درستش در شکل ۱۰).

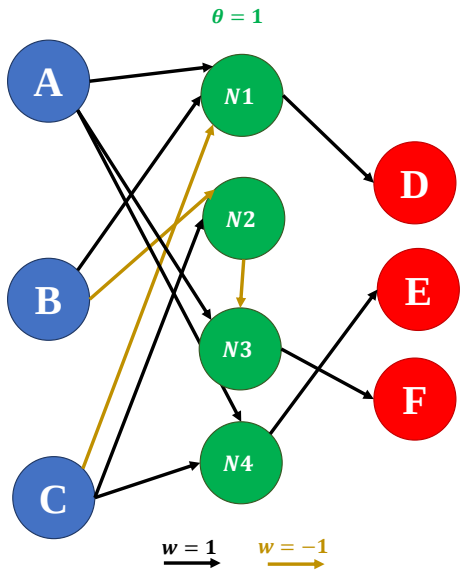


شکل ۱۰: شبکهٔ ادغام‌شدهٔ بهینه (حالت ایده‌آل با ملاحظه).

۴.۱ پاسخ قسمت د

برای این قسمت از سوال، ضمن انجام یکسری بهینه‌سازی نهایی روی شبکه دستورات زیر را می‌نویسیم:

```
class McCullochPittsNeuron:
    def __init__(self, weights, threshold, bias=0):
        self.weights = weights
        self.bias = bias
        self.threshold = threshold
```



شکل ۱۱: شبکهٔ ادغام‌شدهٔ بهینه (حالتی دیگر).

```
    def fire(self, inputs):
        weighted_sum = sum(w * x for w, x in zip(self.weights, inputs)) + self.bias
        if weighted_sum >= self.threshold:
            return 1
        else:
            return 0

class ThreeInputThreeOutputNetwork:
    def __init__(self):
        self.neuron_D = McCullochPittsNeuron([1, 1, -1], 1)
        self.neuron_F1 = McCullochPittsNeuron([1, -1], 1)
        self.neuron_F = McCullochPittsNeuron([1, 1], 1)

    def predict(self, inputs):
        A, B, C = inputs
        D = self.neuron_D.fire([A, B, C])
        E_inputs = [A, C]
        E = self.neuron_D.fire(E_inputs)
        F1 = self.neuron_F1.fire([B, C])
        F_inputs = [A, F1]
        F = self.neuron_F.fire(F_inputs)
        return D, E, F

# Test the network for all possible input states
```

```
network = ThreeInputThreeOutputNetwork()

for A in range(2):
    for B in range(2):
        for C in range(2):
            D, E, F = network.predict([A, B, C])
            print(f"Input: ({A}, {B}, {C}) --> Output: ({D}, {E}, {F})")
```

انواع ترکیب‌ها و روش‌های دیگری در کدهای پیاده‌سازی آورده‌شده در این لینک آزمایش شده است که براساس روش خواسته‌شده در صورت سوال، روش کدنویسی‌ای که به‌صورت صریح از گیت‌های منطقی استفاده نکرده است و کاملاً نورون‌پایه است صحیح است. با اجرای دستورات، نتایج به شرح زیر خواهد بود و مشاهده می‌شود که خواستهٔ سوال کاملاً برآورده شده است:

```
Input: (0, 0, 0) --> Output: (0, 0, 0)
Input: (0, 0, 1) --> Output: (0, 1, 0)
Input: (0, 1, 0) --> Output: (1, 0, 0)
Input: (0, 1, 1) --> Output: (0, 1, 0)
Input: (1, 0, 0) --> Output: (1, 1, 1)
Input: (1, 0, 1) --> Output: (0, 1, 0)
Input: (1, 1, 0) --> Output: (1, 1, 1)
Input: (1, 1, 1) --> Output: (1, 1, 1)
```

هم‌چنین دستورات زیر را نوشته‌ایم که بر اساس شبکهٔ طراحی‌شده و منطق گفته‌شده و با دریافت حالت کنونی و ورودی، مراحل پذیرش با رویت ۱۰۰ را نشان دهد:

```
# Test the network with initial state A=0, B=0
network = ThreeInputThreeOutputNetwork()

# Define the initial state
A, B = 0, 0

while True:
    # Get the input C
    C = int(input("Enter input C (0 or 1): "))

    # Get the next state and acceptance
    D, E, F = network.predict([A, B, C])

    # Print the output
    print(f"Current state: ({A}, {B}), Input: {C} --> Next state: ({D}, {E}), Acceptance: {F}")

    # Update the current state
    A, B = D, E

    # Check if the acceptance is 1, and break the loop if it is
    if F == 1:
        print("Acceptance reached!")
```

```
        break
```

نتیجهٔ یک نمونه از پیاده‌سازی به‌صورت زیر است که مشاهده می‌شود کاملاً منطبق با خواست سوال است:

```
Enter input C (0 or 1): 0
Current state: (0, 0), Input: 0 --> Next state: (0, 0), Acceptance: 0
Enter input C (0 or 1): 1
Current state: (0, 0), Input: 1 --> Next state: (0, 1), Acceptance: 0
Enter input C (0 or 1): 1
Current state: (0, 1), Input: 1 --> Next state: (0, 1), Acceptance: 0
Enter input C (0 or 1): 0
Current state: (0, 1), Input: 0 --> Next state: (1, 0), Acceptance: 0
Enter input C (0 or 1): 0
Current state: (1, 0), Input: 0 --> Next state: (1, 1), Acceptance: 1
Acceptance reached!
```

این دستور را به یک صورت دیگر نیز پیاده‌سازی کرده‌ایم:

```
# Take input from user
current_state = [int(x) for x in input("Enter current state A,B (separated by comma): ").split(',')]
C = int(input("Enter input C: "))
next_state = current_state.copy()

# Test the network for the given input
network = ThreeInputThreeOutputNetwork()

while True:
    D, E, F = network.predict([current_state[0], current_state[1], C])
    next_state[0], next_state[1] = D, E
    if F == 1:
        break
    current_state = next_state.copy()
    C = int(input("Enter input C: "))

# Print the final output
print(f"Current State: ({current_state[0]}, {current_state[1]}) --> Input: {C} --> Next State: ({next_state[0]}, {next_state[1]}) --> Acceptance: {F}")


# Take input from user
current_state = [int(x) for x in input("Enter current state A,B (separated by comma): ").split(',')]
C = int(input("Enter input C: "))
next_state = current_state.copy()

# Test the network for the given input
network = ThreeInputThreeOutputNetwork()
```

```
28
29 # Flag variable to track if F=1 has occurred previously or not
30 f_occurred = False
31
32 while True:
33     D, E, F = network.predict([current_state[0], current_state[1], C])
34     next_state[0], next_state[1] = D, E
35     if F == 1 and not f_occurred:
36         f_occurred = True
37     else:
38         current_state = next_state.copy()
39         C = int(input("Enter input C: "))
40
41     # Print output only if F=1 has occurred previously or F is not equal to 1
42     if f_occurred or F != 1:
43         print(f"Current State: ({current_state[0]}, {current_state[1]}) --> Input: {C} --> Next
       State: ({next_state[0]}, {next_state[1]}) --> Acceptance: {F}")
44
45     if f_occurred:
46         break
47
48 # Print the final output
49 print(f"Current State: ({current_state[0]}, {current_state[1]}) --> Input: {C} --> Next State: ({
       next_state[0]}, {next_state[1]}) --> Acceptance: {F}")
```

نتیجهٔ دو نمونه از پیاده‌سازی با استفاده از روش دیگر به‌صورت زیر است که با منطق خواسته‌شده سازگار است:

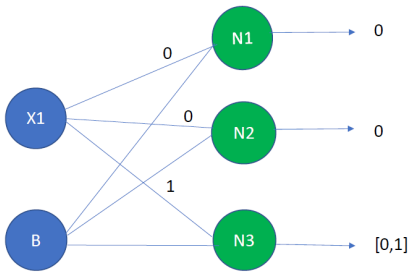
```
1  Enter current state A,B (separated by comma): 0,0
2  Enter input C: 0
3  Enter input C: 1
4  Enter input C: 1
5  Enter input C: 0
6  Enter input C: 0
7  Current State: (1, 0) --> Input: 0 --> Next State: (1, 1) --> Acceptance: 1
8  ------------------------------------------------
9
10 Enter current state A,B (separated by comma): 1,1
11 Enter input C: 0
12 Current State: (1, 1) --> Input: 0 --> Next State: (1, 1) --> Acceptance: 1
13 Current State: (1, 1) --> Input: 0 --> Next State: (1, 1) --> Acceptance: 1
```

۵.۱ راه‌حل و نگاه دوم

ما برای حل این چالش دو منطق متفاوت را پیش بردیم که در ادامه با شرح مختصری راجب این ماشین دیدگاه خود را مطرح خواهیم کرد. با توجه به سوال مطرح شده که در آن شبیه سازی یک ماشین متناهی قطعی خواسته شده است ما به مطالعه و

شناخت این رفتار این ماشین پرداختیم و با توجه به درکی که از این ماشین به دست آمد تلاش کردیم که در نخست رفتار هر یک از حالت‌های این ماشین را به صورت مجزا پیاده سازی کنیم که برای این کار از نورون Mcculloch-Pitts توسعه یافته بهره بردیم. در این سوال DFA به این صورت طراحی شده تا رشته 100 را دی یک دنباله از اعداد باینری را تشخیص دهد که در ادامه با توجه به ورودی‌های ممکن حرکات هر حالت از این ماشین را نشان می‌دهیم. در بهر گیری از این ماشین رشته ورودی ما در ایتدا بر روی یک ابزار ذخیره سازی ذخیره می‌شود و با استفاده ار یک حد خواندن از یک سمت به صورت بیت به بیت خوانده شده و به حالتی که در ان قرار داریم اعمال می‌شود و این ماشین با توجه به رفتاری که برای آن تعریف شده و ورودی اعمال شده تصمیم گیری می‌کند. ما سه حالت مدنظر را با اعداد باینری شماره گذاری کرده‌ایم به این صورت که حالت اول (00) ، حالت دوم (01) و حالت سوم (10).

حالت اول



شکل ۱۲: تصویر مربوط به حالت اول در نگاه دوم.

توابع حاصل شده از نورون‌ها برای شبکه اول

$$N1 = X1.W1 + B = 0$$

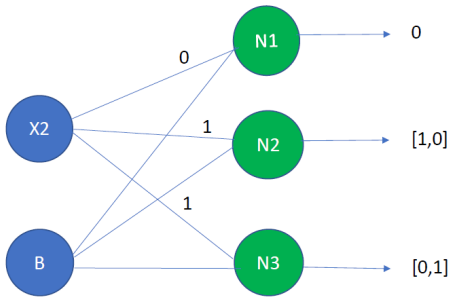
$$N2 = X1.W2 + B = 0$$

$$N3 = \begin{cases} 1 & if X1 = 1 \Rightarrow X1.W3 + B \\ 0 & else \end{cases}$$

در این حالت به ازای هر ورودی صفر و یک دو حرکت وجود دارد که البته حالت پذیزش و یا عدم پذیرش آن هم مدنظر می‌باشد.همانطور که گفته شد رشته 100 با یک شروع می‌شود پس تا لحظه دیدن یک در حالت اول می‌مانیم و تنها با دیدن یک به حالت بعدی می‌رویم(01).برای شبیه سازی این عملیات ما سه نورون در خروجی در نظر گرفته‌ایم که خروجی نورون $N1$ حالت پذیرش و یا عدم پذیرش را مشخص می‌کند به این صورت که خروجی یک به معنای پذیرش و صفر به معنای عدم پذیرش می‌باشد و دو نورون $N2$ و $N3$ هر کدام یک بیت از شماره حات بعد را نمایش می‌دهند.لازم به ذکر است که مقدار $bias$ و

$threshold$، صفر می‌باشد و همچنین روش مقایسه نورون ما $NET == 0$ می‌باشد.ما می‌دانیم که در حالت اول در هر صورت ما با عدم پذیرش روبه‌رو هستیم به این دلیل که در ابتدا تنها یک بیت از رشته مد نظر ما دیده شده است پس در هر صورت خروجی نورون اول ما باید صفر باشد پس وزن‌ها و بایاس را جوری درنظر گرفتیم که خروجی نورون اول صفر شود. نورون‌های $(N3, N2)$ قرار است با خروجی که تولید می‌کنند شماره حالت بعدی را نشان دهند.پس اگر بیت اول ورودی ما صفر بود که ما قرار است در همان حالت فعلی بمانیم(00) این به این معناست که نورون‌های $(N2, N3)$ باید صفر را تولید کنند.اما اگر بیت اول ورودی ما یک بود ما باید به حالت (01) برویم که در این صورت نورون $N2$ صفر می‌ماند و نورون $N3$ که بیت کم ارش را برای آن در نظر گرفته ایم یک می‌شود که در جدول و شکل نورون ها همه‌ی حالت های ممکن درج شده است.

حالت دوم



شکل ۱۳: تصویر مربوط به حالت دوم در نگاه دوم.

توابع حاصل از نورون‌ها برای شبکه دوم

$$N1 = X2.W1 + B = 0$$

$$N2 = \begin{cases} 1 & if X2 = 0 \Rightarrow X2.W2 + B \\ 0 & else \end{cases}$$

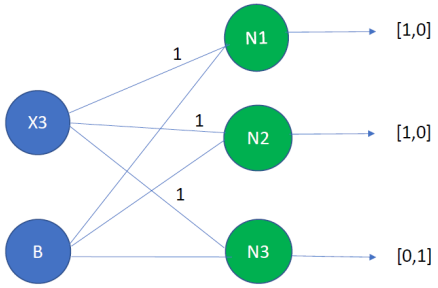
$$N3 = \begin{cases} 1 & if X2 = 1 \Rightarrow X2.W3 + B \\ 0 & else \end{cases}$$

توضیحات نورون $N1$ دقیقا مشابه حالت قبل است به این معنی که چون هنوز دنباله مورد نظر ما قطعا در این مرحله نمی‌تواند دیده شده باشد پس باید خروجی آن صفر در نظر گرفته شود.ما باید خود را کاملا در این حالت متصور باشیم به این معنی که به صورت منطقی ما اینگونه به این موضوع نگاه می‌کنیم که حتما یک را دیده ایم که اکنون در این حالت قرار داریم پس در صورت دیدن صفر که بیت دوم مورد نظر ما بعد از یک است باید به حالت بعد یعنی حالت سوم(10) برویم پس با توجه به این نکات اگر در این حالت صفر را دیدیم نورون $N2$ باید یک شود و نورون $N3$ باید صفر شود که نشان گر حالت سوم می‌باشد اما اگر ورودی

ما یک بود مه باید همچنان در حالت کنونی که (01) می‌باشد بمانیم که در این صورت نورون $N2$ صفر و نورون $N3$ باید یک شود که نمایان گر همان حالت فعلی است.(01).

لازم به ذکر است که این شبکه دارای $bias$ و $threshold$ یک است و روش مقایسه آن برای هر نورون متفاوت است.

حالت سوم



شکل ۱۴: تصویر مربوط به حالت سوم در نگاه دوم.

توابع حاصل از نورون‌ها برای شبکه سوم

$$N1 = \begin{cases} 1 & if X3 = 0 \Rightarrow X3.W1 + B \\ 0 & else \end{cases}$$

$$N2 = \begin{cases} 1 & if X3 = 0 \Rightarrow X3.W2 + B \\ 0 & else \end{cases}$$

$$for X3 = 0, 1 \Rightarrow X3.W3 + B = 1$$

نورون $N1$ در صورت دیده شدن صفر حالت پذیرش را یک می‌کند به این دلیل که با توجه به در این حالت قرار گرفتن می‌دانیم که حتما تا اینجای کار رشته (10) دیده شده است پس بادیدن یک صفر دیگر رشته مد نظر ما کاملا دیده شده.به همین ترتیب نورون های $N2, N3$ باید با دیدن صفر دیگر حالت نهایی را نمایش دهند که (11) می‌باشد پس باید هردو یک را تولید کنند و اما اگر بیت یک را دیدند هم حالت پذیرش باید صفر شود و هم حالت قبل را باید در خروجی نمایش دهند که در جدول دقیقا ذکر شده است.همچنین bias و threshold یک در نظر گرفته شده است.

حالت اول:

ورودی	N1	N2	N3
۰	۰	۰	۰
۱	۰	۰	۱

حالت دوم:

ورودی	N1	N2	N3
۰	۰	۱	۰
۱	۰	۰	۱

حالت سوم:

ورودی	N1	N2	N3
۰	۱	۱	۱
۱	۰	۰	۱