پردازش اطلاعات کوانتومی

پاییز ۱۴۰۱

ارائه ۹

۱ مقدمه

در این ارائه به الگوریتم Simon می‌پردازیم. در این مسئله با یک «جعبه سیاه» سروکار داریم که تابع $f : \{0,1\}^n \to \{0,1\}^n$ را پیاده سازی می‌کند. دقت کنید که این تابع یک ورودی n بیتی را به یک خروجی n بیتی نگاشت می‌کند. ما از قبل می‌دانیم که یک رشته ثابت و مخفی n بیتی وجود دارد، یعنی $s \in \{0,1\}^n$، که به ازای آن تساوی زیر همواره برقرار است:

$$f(x) = f(x \oplus s) \tag{۱}$$

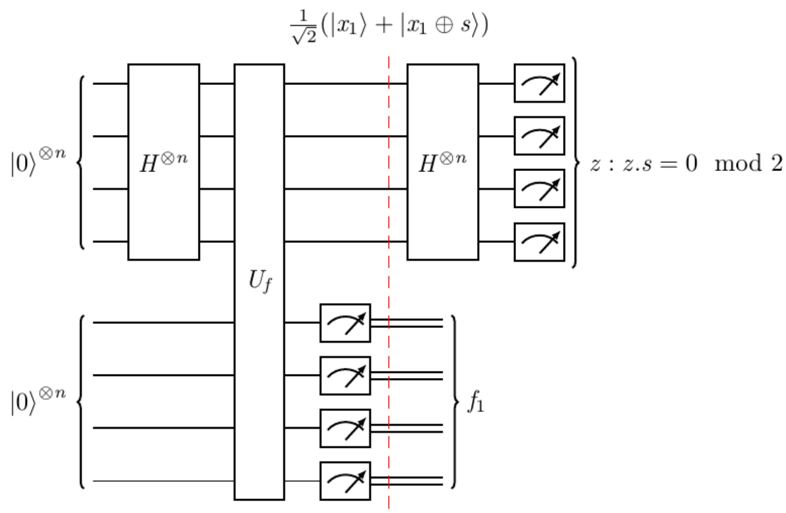
که در آن $\oplus$ عملیات بیتی XOR است. به توابعی که این خاصیت را برآورده می‌کنند توابع ۲-۱ (دو به یک) می‌گویند چرا که دقیقاً هر دو ورودی را به یک مقدار خروجی نگاشت می‌کنند.

مثال: فرض کنید که f تابعی است که رشته‌های دو بیتی را به عنوان ورودی و خروجی دریافت می‌کند. اگر $s = 01$ باشد، آنگاه تحت شرایط زیر این تابع ۲-۱ است:

ورودی	حاصل XOR	خروجی
00	01	00
01	00	00
10	11	11
11	10	11

می‌بینیم که چون $00 \oplus s = 01$ است، حاصل $f(00)$ و $f(01)$ نیز برابر است. به طور مشابه $10 \oplus s = 11$ و در نتیجه $f(10) = f(11)$ است. $\square$

هدف این مسئله یافتن رشته مخفی s با حداقل فراخوانی تابع f است. در شرایط کلاسیک ما در بدترین شرایط ممکن است نیاز داشته باشیم تابع f را $\frac{2^n}{2} + 1$ مرتبه فراخوانی کنیم. توجه کنید که تابع را باید آنقدر فراخوانی کنیم که دو خروجی یکسان را مشاهده کنیم و در بدترین شرایط ممکن است $\frac{2^n}{2}$ خروجی متفاوت را مشاهده کنیم. البته با استفاده از $\sqrt{2^n} = 2^{n/2}$ فراخوانی می‌توان با **احتمال خوبی** دو خروجی متفاوت را پیدا کرد که البته همچنان به نسبت اندازه ورودی نمایی است. اگر تعداد فراخوانی‌ها به صورت قابل توجه از آستانه $2^{n/2}$ کمتر باشد احتمال موفقیت



شکل ۱: مدار

خیلی کم می‌شود. بنابراین به نظر می‌رسد بهترین مقداری که به صورت کلاسیک می‌توان به آن دست پیدا کرد همین آستانه باشد. در ادامه به راه حل کوانتومی این مسئله می‌پردازیم.

۲ الگوریتم Simon

اجرای این الگوریتم بسیار ساده است. مدار این الگوریتم در شکل ۱ آمده است. گام‌های این الگوریتم به شرح زیر هستند:

۱. متناظر با ورودی n سیم کوانتومی را در ایجاد کنید و آنها را با مقدار $|0\rangle^{\otimes n}$ مقداردهی کنید.

۲. متناظر با خروجی n سیم کوانتومی ایجاد کنید و و آنها را با مقدار $|0\rangle^{\otimes n}$ مقداردهی کنید.

۳. دریچه $H^{\otimes n}$ را به سیم‌های متناظر ورودی اعمال کنید. یعنی آنها را در حالت برهم‌نهاده قرار دهید.

۴. جعبه سیاه را اعمال کنید.

۵. سیم‌های متناظر با خروجی را اندازه‌گیری کنید. این اندازه‌گیری باعث می‌شود که خروجی به یکی از $\frac{2^n}{2}$ حالت ممکن فروشکسته [1] شود. این حالت را با f_1 نام‌گذاری کنید. هر حالت خروجی از طریق دو ورودی متفاوت ممکن است تولید شوند. بنابراین این اندازه‌گیری حالت سیم‌های متناظر ورودی را نیز تغییر می‌دهد و آنها را

[1] Collapse

به حالت $\frac{1}{\sqrt{2}}|x_1\rangle + \frac{1}{\sqrt{2}}|x_1 \oplus s\rangle$ می‌برد که x_1 یک رشته دلخواه ورودی است و تساوی زیر برقرار است:

$$f|x_1\rangle = f|x_1 \oplus s\rangle = f_1. \tag{۲}$$

۶. بر روی سیم‌های متناظر ورودی دریچه $H^{\otimes n}$ را اعمال کنید. از فرمول پر استفاده‌ای که در ارائه‌های قبلی نیز آن را به کار برده بودیم بدست خواهیم آورد:

$$\frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} \frac{1}{\sqrt{2}}((-1)^{z.x_1} + (-1)^{z.(x_1\oplus s)})|z\rangle \tag{۳}$$

حال توجه کنید که احتمال حالت‌هایی که $(-1)^{z.x_1} \neq (-1)^{z.(x_1\oplus s)}$ باشد برابر صفر است. پس فقط حالت‌هایی باقی می‌مانند که $(-1)^{z.x_1} = (-1)^{z.(x_1\oplus s)}$ باشد. این تساوی زمانی برقرار است که هر دو توان زوج یا هر دو توان فرد باشند. یعنی باقیمانده آنها به پیمانه ۲ باید برابر باشد:

$$z.x_1 = z.(x_1 \oplus s) \mod 2 \tag{۴}$$

$$z.x_1 = z.x_1 \oplus z.s \mod 2 \tag{۵}$$

$$z.s = 0 \mod 2 \tag{۶}$$

۷. سیم‌های متناظر با ورودی را در پایه‌های محاسباتی اندازه‌گیری کنید. با توجه به توضیحات فوق یک z را بدست خواهیم آورد که می‌دانیم در مورد آن عبارت زیر صحیح است:

$$z.s = 0 \mod 2. \tag{۷}$$

حال اگر فرایند فوق را تقریباً n مرتبه تکرار کنیم (از گام اول) آنگاه تعداد کافی معادله به فرم $z.s = 0 \mod 2$ بدست می‌آوریم که از طریق آنها می‌توانیم n مجهول که همان بیت‌های رشته مخفی s هستند را محاسبه کنیم:

$$z_1^1 s_1 + \cdots + z_n^1 s_n = 0 \mod 2 \tag{۸}$$

$$z_1^2 s_1 + \cdots + z_n^2 s_n = 0 \mod 2 \tag{۹}$$

$$\vdots \tag{۱۰}$$

$$z_1^{n-1} s_1 + \cdots + z_n^{n-1} s_n = 0 \mod 2 \tag{۱۱}$$

که در اینجا z_j^i نشان‌دهنده حاصل اندازه‌گیری کیوبیت j-ام در تکرار i-ام گام‌های الگوریتم است. توجه کنید که یک پاسخ بدیهی این سیستم معادلات $s = 0$ است که ما به دنبال آن نیستیم. همچنین توجه کنید که اگر $z = 0$ را در خروجی مشاهده کنیم باز هم اطلاع بیشتری در مورد رشته مخفی به ما نمی‌دهد چونکه به ازای هر رشته n بیتی مخفی، معادله $z.s = 0 \mod 2$ برقرار است.

۱.۲ مثال

تابعی که در مثال ابتدای ارائه بررسی شد را در نظر بگیرید و فرض کنید که جعبه سیاه کوانتومی آن به ما داده شده است. فرض کنید که یک بار الگوریتم فوق را اجرا کرده‌ایم. از میان تمام حالت‌های ممکن z ابتدا باقیمانده به پیمانه ۲ حاصلضرب بیتی آنها در $s = 01$ را بدست می‌آوریم:

$$z = 00 \rightarrow 0.0 + 0.1 = 0 \mod 2 \tag{۱۲}$$

$$z = 01 \rightarrow 0.0 + 1.1 = 1 \mod 2 \tag{۱۳}$$

$$z = 10 \rightarrow 1.0 + 0.1 = 0 \mod 2 \tag{۱۴}$$

$$z = 11 \rightarrow 1.0 + 1.1 = 1 \mod 2 \tag{۱۵}$$

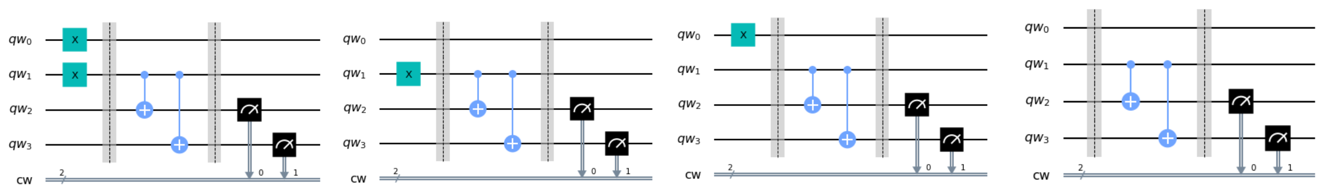
بنابراین فقط احتمال مشاهده $z = 00$ و یا $z = 10$ وجود دارد. اگر بعد از دو بار فراخوانی ($n = 2$) هر دوی این رشته‌ها را در خروجی اندازه‌گیری کنیم، آنگاه متوجه می‌شویم که تنها رشته‌هایی که می‌توانند $z.s = 0 \mod 2$ را ارضا کنند $s = 00$ و $s = 01$ هستند. همانطور که گفته شد $s = 00$ پاسخ بدیهی است و بنابراین $s = 01$ به عنوان رشته مخفی پیدا می‌شود. توجه کنید که ممکن است بدشانسی بیاوریم و در چند اجرای متوالی $z = 00$ را مشاهده کنیم که هیچ کمکی نمی‌کند. اما احتمال این رخداد با تکرار کم می‌شود ($1/2^{(n-1)}$). همچنین با افزایش طول رشته‌ها، تعداد خروجی‌های ممکن افزایش پیدا می‌کند و احتمال مشاهده $z = 0$ کم می‌شود. از طرفی برای اینکه به تعداد کافی معادله مستقل پیدا کنیم هر بار که از خروجی یک نمونه را دریافت می‌کنیم (با اندازه‌گیری) اگر یکی از نتایج قبلی را دوباره مشاهده کنیم نمی‌توان با آن اطلاعات خود درباره رشته مخفی را افزایش داد. به این ترتیب این مسئله هم یکی از فاکتورهایی است که کارایی الگوریتم سیمون را تحت تأثیر قرار می‌دهد.

۲.۲ شبیه‌سازی

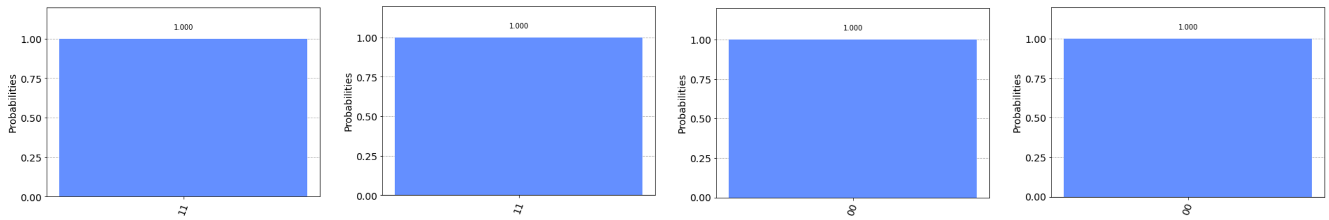
در این قسمت مثال قبلی را از طریق شبیه‌ساز بررسی می‌کنیم. به این منظور ابتدا نیاز داریم که جعبه سیاه تابع مذکور به ازای رشته مخفی $s = 01$ را پیاده‌سازی کنیم. توجه کنید که مدار نشان داده‌شده در شکل ۲ این کار را انجام می‌دهد. به صورت خاص مشخص است که به ازای ورودی‌های 00 و 01 خروجی برابر 00 است و به ازای ورودی‌های 10 و 11 خروجی 11 است.

در گام بعدی، مطابق الگوریتم Simon ورودی‌های برهم‌نهاده را به مدار اعمال می‌کنیم و سپس خروجی را اندازه‌گیری می‌کنیم. به این منظور ابتدا مدار به شکل زیر ایجاد می‌کنیم:

```
s = "01"
n = len(s)
qw = QuantumRegister(n*2, name="qw")
```

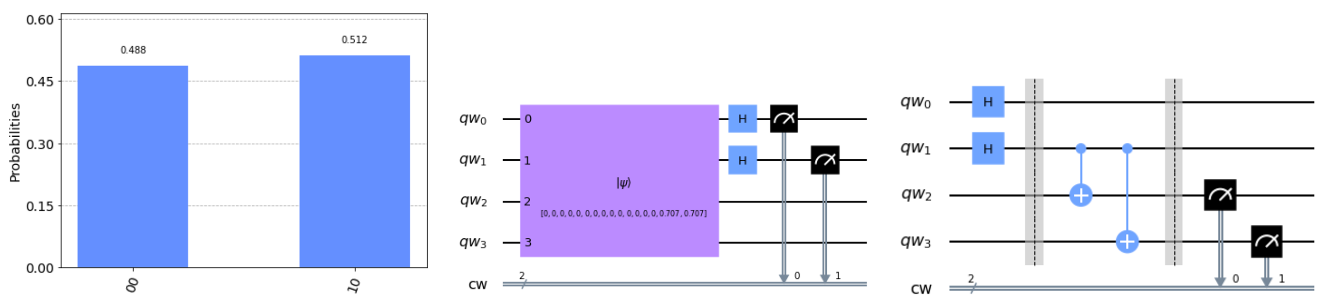


(آ) ورودی 00 (ب) ورودی 10 (ج) ورودی 01 (د) ورودی 11



(ه) خروجی 00 (و) خروجی 10 (ز) خروجی 01 (ح) خروجی 11

شکل ۲: جعبه سیاه مثال



(آ) بخش اول مدار سیمون (ب) بخش دوم مدار سیمون (ج) حاصل اندازه‌گیری مدار سیمون

شکل ۳: بررسی مدار سیمون

```
cw = ClassicalRegister(n, name="cw")
simon_circuit = QuantumCircuit(qw, cw)
for i in range(n):
    simon_circuit.h(i)
simon_circuit.barrier()
simon_circuit.cx(1, 2)
simon_circuit.cx(1, 3)
simon_circuit.barrier()
for i in range(n):
    simon_circuit.measure(i+n, i)
```

مدار حاصل در شکل ۳آ آمده است. سپس، با استفاده از کد زیر حالت سیستم را بررسی می‌کنیم:

```
backend = Aer.get_backend("statevector_simulator")
```

```
result = execute(simon_circuit, backend=backend, shots=1).result()
print('State after first measurement:', result.get_statevector())
```

اگر این شبیه‌سازی را انجام دهیم، متوجه می‌شویم که سیستم در یکی از دو حالت زیر قرار می‌گیرد:

$$\langle\psi_1| = \frac{1}{\sqrt{2}}[1100, 0000, 0000, 0000] \quad (۱۶)$$

$$= \frac{1}{\sqrt{2}}([1000, 0000, 0000, 0000] + [0100, 0000, 0000, 0000]) \quad (۱۷)$$

$$\langle\psi_2| = \frac{1}{\sqrt{2}}[0000, 0000, 0000, 0011] \quad (۱۸)$$

$$= \frac{1}{\sqrt{2}}([0000, 0000, 0000, 0010] + [0000, 0000, 0000, 0001]) \quad (۱۹)$$

که در نهایت به صورت زیر قابل نوشته شدن است:

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1000\rangle) \quad (۲۰)$$

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0111\rangle + |1111\rangle) \quad (۲۱)$$

می‌بینیم که به ازای ورودی‌های 11 و 01 خروجی برابر 11 است و برای ورودی‌های 00 و 10 خروجی 00 است که مورد انتظار است. به صورت دقیق‌تر، هرگاه که خروجی 11 اندازه گرفته می‌شود حالت سیم‌های متناظر ورودی 11 و 01 هستند و هرگاه که خروجی 00 اندازه گرفته می‌شود حالت سیم‌های متناظر ورودی 00 و 10 است. حال یک مدار جدید می‌سازیم و آن را با حالت خروجی از مدار اول مقداردهی می‌کنیم:

```
qw = QuantumRegister(n*2, name="qw")
cw = ClassicalRegister(n, name="cw")
simon_circuit = QuantumCircuit(qw, cw)
simon_circuit.initialize(result.get_statevector(), qw)
for i in range(n):
    simon_circuit.h(i)
for i in range(n):
    simon_circuit.measure(i, i)
```

حاصل در شکل ۳ب نمایش داده شده است. سپس این مدار را به شکل زیر شبیه‌سازی می‌کنیم:

```
backend = BasicAer.get_backend('qasm_simulator')
shots = 1000
results = execute(simon_circuit, backend=backend, shots=shots).result()
counts = results.get_counts()
plot_histogram(counts)
```

حاصل شبیه‌سازی در شکل ۳ج آمده است. می‌بینم که مطابق انتظار فقط خروجی‌های $z = 00$ و $z = 10$ دیده می‌شوند، چرا که فقط حاصلضرب بیتی این دو مقدار در $s = 01$ به پیمانه ۲ صفر می‌شود.