نشر ریاضی، سال ۷، شمارهٔ ۲، صص ۱۵ـ۲۴

دو مسألهٔ الگوریتمی•

برونو پوازا*
ترجمهٔ یوسف امیرارجمند

۱. یک جمعِ بسیار ساده

مبحث الگوریتم، علمِ محاسبه است؛ کلمهٔ «الگوریتم» از نام خوارزمی ریاضیدان مشهور گرفته شده است. موضوع این علم یافتن روشهای کارآمد برای محاسبه است: کسی که از آن استفاده می‌کند باید بتواند به کمک این علم بین دو روش که به نتیجهٔ واحدی منتهی می‌شوند، روشی را انتخاب کند که به وقت و تلاش کمتری نیاز دارد. بنابراین، وظیفهٔ این علم در وهلهٔ اول، ارزیابی زمان و امکانات لازم برای به نتیجه رسیدن یک محاسبه است؛ در مرحلهٔ بعد باید، درصورت امکان، محاسبه‌ای را جانشین این محاسبه کند که کارآمدتر باشد، یعنی همان کار را انجام دهد اما با هزینهٔ کمتر.

باید بین دو نوع [مدت] زمان تمایز قائل شد: زمانِ موازی و زمان متوالی. زمان متوالی، مدتی است که صرف یک کار می‌شود اگر آن کار را به تنهایی انجام دهیم و زمان موازی، مدت لازم برای یک کار است اگر آن را با همکاری دوستان انجام دهیم.

بنا به تجربه با قرارداد، الگوریتمی را قابل اجرا به‌حساب می‌آوریم که، وقتی داده‌ای به طول n به‌آن داده شود، جواب را در یک زمان متوالی به‌دست دهد که به‌صورت یک چندجمله‌ای برحسب n باشد، یعنی زمانی‌که کران بالای آن An^A، به‌ازای عدد ثابتی مانند A، است. و برای اینکه ارزش آن را داشته باشد که از رفقا کمک بگیریم، می‌پذیریم که این الگوریتم واقعاً رضایتبخش نخواهد بود مگر آنکه زمان موازیِ لازم برای اجرای آن به‌صورت لگاریتمی و دارای کران بالای $B \log n$ باشد، که در آن B یک عدد ثابت است.

واضح است که مدت زمان اجرای الگوریتم، چه متوالی باشد چه موازی، یک مفهوم ریاضیِ خوش‌تعریف نیست؛ بنابراین باید دقیقتر بود.

اولاً باید معلوم کنیم که محاسبات در مورد چیست. واقعگرایانه‌ترین دیدگاه این است که محاسبات در مورد کلماتی است که با یک الفبای متناهی [مرکب از ارقام] نوشته می‌شوند. در مدرسه به‌سختی با الفبای ۰ ۱ ۲ ۳ ۴ ۵ ۶ ۷ ۸ ۹ عادت کرده‌ایم. اکنون که در عصر کامپیوتر هستیم، کار را آسان می‌کنیم و از میان این ارقام، تنها دو رقم اول ۰ و ۱ را نگه می‌داریم. حتماً می‌دانید که می‌توان اعداد را برمبنای دو (یا هر مبنای دیگری!) نوشت. کلمهٔ $\varepsilon_n\varepsilon_{n-1}\dots\varepsilon_1\varepsilon_0$ نمایشگر عدد صحیح $\varepsilon_n \times 2^n + \varepsilon_{n-1} \times 2^{n-1} + \dots + \varepsilon_1 \times 2 + \varepsilon_0$ است؛ مثلاً ۱۰۱۰۱ در مبنای دو نمایشگر عددی است که در مبنای ده به‌صورت ۲۱ نوشته می‌شود.

جمع کردن دو عدد که در مبنای دو نوشته شده باشند چندان مشکل نیست، زیرا جدول جمع را می‌توان زود یادگرفت؛ مثال شکل ۱ را نگاه کنید، در این شکل ارقامِ نگه داشته شده با حروف ایرانیک نوشته شده‌اند: یک به‌علاوهٔ یک، می‌شود ده؛ می‌نویسیم صفر، و یک را نگه می‌داریم؛ یک به‌علاوهٔ یک به‌علاوهٔ یک می‌شود یازده؛ می‌نویسیم یک، و یک را نگه می‌داریم؛ و همین‌طور الی آخر ...

بنابراین می‌بینیم که، در یک محاسبه، دادهٔ اولیه یک کلمهٔ دودویی است، یعنی رشته‌ای متناهی از ۰ ها و ۱ هاست، و حاصلْ هم به نَبع یک کلمهٔ دودویی است. مثلاً، در مورد جمع، دو کلمهٔ دودویی داریم که باید نمایشگر

```
 1 1   1 1 1
   1 0 1 1 1
+  1 1 0 0 1 1
-------------
 1 0 0 1 0 1 0
```

شکل۱

اعداد صحیح x و y باشند، و می‌خواهیم نمایش دودویی حاصلجمع آنها $x+y$ را بیابیم. طول داده‌ها، برطبق تعریف، تعداد کل نمادهای ۰ و ۱ ای است که در آن داده وجود دارد؛ بدین ترتیب، وقتی دو عدد n رقمی را با یکدیگر جمع می‌کنیم، طول داده $2n$ است. طول آن کلمهٔ دودویی که نمایشگر عدد x است، تقریباً برابر است با لگاریتم آن عدد در مبنای دو که با $\log x = \mathrm{Log}_{10} x / \mathrm{Log}_{10} 2$ نشان داده می‌شود.

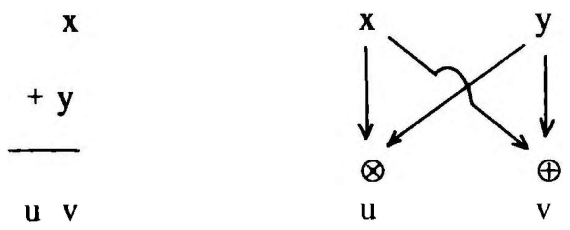
زمان محاسبه برحسب طول داده، یعنی n، حساب می‌شود. تقریباً واضح است که زمانِ لازم برای جمع کردن دو عدد n رقمی با روش دبستانی، تابعی است خطی از n به شکل $An + B$، که متناظر است با تعداد عملیاتی که باید روی نمادهای ۰ و ۱ انجام شود تا حاصل به‌دست آید. این هم واضح است که روشی که اجرای آن گام به گام پیش می‌رود الزاماً روشی است متوالی (که بسیار مناسبِ دبستان است: در دبستان خانم معلم نوشتن از روی دست همکلاسیهایتان را قدغن کرده است!) زیرا، برای دانستن رقم iام سمت راست در عدد حاصل، کافی نیست که ارقام iام دو عددی را که به هم اضافه می‌کنیم بدانیم؛ باید بدانیم آیا باقیمانده‌ای وجود دارد که از سمت راست بیاید یا خیر.

آخیقار حکیم آشوری این الگوریتم را به‌صورت زیر اصلاح کرد تا یک زمان موازی لگاریتمی به‌دست آورد: فرض کنید می‌خواهیم دو عدد x و y را که به‌وسیلهٔ دو رشتهٔ دودویی با طولی کمتر از 2^{m+1} نمایش داده می‌شوند با یکدیگر جمع کنیم. این دو را به دو قسمت می‌کنیم، و می‌نویسیم $x = a \times 2^m + b$ و $y = c \times 2^m + d$، که در آن اعداد a، b، c، و d با رشته‌هایی دودویی نمایش داده می‌شوند که طول هر یک از 2^m کمتر است. سپس، در حینی که آخیقار دو عدد b و d را با یکدیگر جمع می‌کند، همسر اول او شامیران، a و c را با این فرض که باقیماندهٔ نهایی جمع آخیقار صفر است، جمع می‌کند؛ و همسر دوم او ایشتار، همین جمع را با فرض اینکه باقیمانده ۱ خواهد بود انجام می‌دهد. هر سهٔ آنها همزمان کار می‌کنند، و بعد از اینکه کارشان را تمام کردند معلوم می‌شود که کدامیک، شامیران یا ایشتار، کارش بیهوده نبوده است! نتیجه اکنون در دسترس است. اما نباید وقت را برای نوشتن آن تلف کرد: این‌کار به‌دست $2^m + 1$ زنی که در حرمسرای آخیقار هستند سپرده خواهد شد. زن iام باید رقم iام حاصل را بنویسد؛ او برحسب باقیمانده‌ای که آخیقار حساب کرده است، یا رقم iام شامیران و یا رقم iام ایشتار را می‌نویسد.

فرض می‌کنیم t_m زمان موازی لازم برای به‌دست آوردن حاصلجمع دو عدد 2^m رقمی به‌علاوهٔ یک عدد یک‌رقمی (باقیمانده) باشد. بنابراین داریم $t_{m+1} \leqslant t_m + A$، که در آن A یک ثابت است و متناظر است با زمانی که زنهای حرمسرا برای انتخاب خود لازم دارند؛ از آنجا داریم $t_m \leqslant A \times m$. بنابراین اگر $2^{m-1} < n \leqslant 2^m$، برای جمع کردن دو عدد n رقمی با این روش، زمان موازی لازم بیشتر از $A \times m \leqslant A(\log n + 1)$ نخواهد بود.

۲. مدارهای بولی

در اثبات کوچکی که در بالا آمد، مسامحه‌ای صورت گرفت: در این اثبات برای تعریف زمان، چه زمان متوالی و چه زمان موازی، شهود خواننده به کمک طلبیده شده است. در این بخش وسیلهٔ جبران این بی‌دقتی را فراهم می‌آوریم.



شکل۲

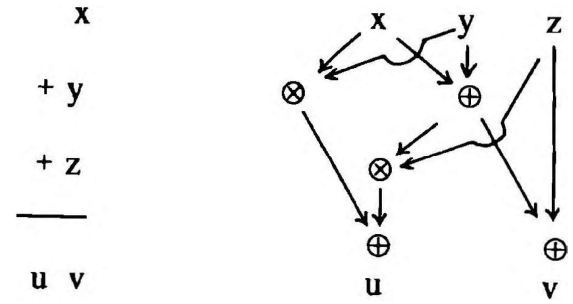
تصویر شکل ۲یک مدار بولی بسیار ساده را نشان می‌دهد و تصویر شکل ۳مدار دیگری را نشان می‌دهد که کمی پیچیده‌تر است.

همان‌طور که می‌بینیم، مدار عبارت است از یک نمودار جهتدار، یعنی ساختاری مرکب از تعدادی متناهی نقطه، که در اینجا به آنها «دَر» می‌گوییم، و تعدادی متناهی پیکان. هر پیکانی از یک دَر به دَرِ دیگر می‌رود. سه نوع دَر وجود دارد: اول دَرِ ورودی که هیچ پیکانی به‌آن منتهی نمی‌شود، و با یک نماد متغیر x، y، z ... (یک متغیر واحد ممکن است چند در ورودی را مشخص کند)، و یا با یکی از دو نماد ثابت ۰ یا ۱ مشخص می‌گردد؛ دوم درهای جمعی، که با نماد $\oplus$ نشان داده می‌شوند و بالاخره درهای ضربی که با نماد $\otimes$ مشخص می‌شوند و دو پیکان به آنها منتهی می‌شود. دری را که پیکانی از آن خارج نمی‌شود دَرِ خروجی گویند. به‌علاوه، لازم است نمودار شامل هیچ دورِ جهتدار نباشد، یعنی در آن هیچ راه دوری، که به نقطهٔ اولش بازگردد و تمام پیکانهایش در یک جهت واحد باشند، توان یافت.

دو نماد $\oplus$ و $\otimes$ نشان‌دهندهٔ عملیات اصلیی هستند که فقط روی دو نماد ۰ و ۱ اجرا می‌شوند و به‌ترتیب جمع و ضرب به پیمانهٔ دو هستند که مقادیر آنها برابر است با $0 \oplus 0 = 0$، $0 \oplus 1 = 1$، $1 \oplus 0 = 1$، $1 \oplus 1 = 0$، $0 \otimes 0 = 0$، $0 \otimes 1 = 0$، $1 \otimes 0 = 0$، $1 \otimes 1 = 1$. این دو عملِ بسیار ساده ویژگیهای معمولِ جمع و ضرب را دارند، و تحت آنها مجموعهٔ $\{0, 1\}$ هیأتی می‌سازد که آن را با $\mathbf{F}_2$ نشان می‌دهیم.

به مدار C با m خروجی، که ورودیهای آن با n متغیر x_1، ... ، x_n مشخص شده‌اند، تابع f_C از $\{0,1\}^n$ در $\{0,1\}^m$ نظیر می‌شود که به طریق زیر محاسبه می‌گردد: رشتهٔ دودویی ε_1، ... ، ε_n مرکب از ε_i ها را درنظر می‌گیریم که ε_i یا ۰ است یا ۱؛ به‌جای هر x_i مقدار ε_iی متناظر را قرار می‌دهیم؛ سپس این مقادیر را در مدار پراکنده می‌سازیم، پیکانها را دنبال می‌کنیم، و هر بار که از دری عبور می‌کنیم عملی را که متناظر با آن است انجام می‌دهیم: هر دَرِ $\oplus$، حاصلجمع دو مقداری را که دریافت می‌کند به پیمانهٔ دو تحویل می‌دهد. هر دَرِ $\otimes$، حاصلضرب دو مقداری را که دریافت می‌کند به پیمانهٔ دو تحویل می‌دهد. از آنجا که در نمودارْ دور وجود ندارد، بالاخره این عمل به هر یک از m هایی که از مدار خارج می‌شوند، بدون **هیچ‌گونه** ابهامْ مقداری می‌دهد. طبق تعریف، m تایی متناظر، مقدار $f_C(\varepsilon_1, \ldots, \varepsilon_n)$ است.

مثلاً، در مورد مدار شکل ۲، اگر به دو ورودی x و y مقدار ۱ را نسبت دهیم، در خروجی u عدد ۱ و در خروجی v عدد ۰ به‌دست می‌آید. در مدار شکل ۳، اگر مقدار ۱ را به ورودیهای x و z و مقدار ۰ را به ورودی y نسبت دهیم، باز هم در خروجی u عدد ۱ و در خروجی v عدد ۰ به‌دست

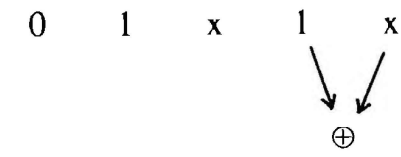


شکل۳

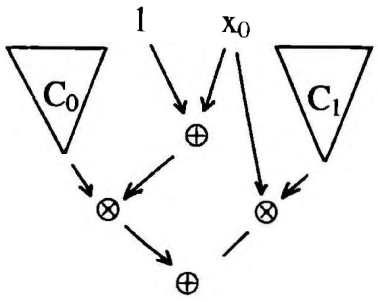
می‌آید. خواننده می‌تواند، اگر روش بهتری به‌نظرش نمی‌رسد، با انجام دادن تمام آزمایشها، تحقیق کند که این مدار آخر در v حاصلجمع x، y و z را به پیمانهٔ دو به‌دست می‌دهد، در حالی‌که u مقداری را نمایش می‌دهد که در میان مقادیر x، y و z از همه بیشتر ظاهر می‌شود (اگر در میان مقادیر x، y و z دو تا ۱ باشد داریم $u = 1$، در غیر این‌صورت $u = 0$).

هر تابع دلخواهی از $\{0,1\}^n$ به $\{0,1\}^m$ را می‌توان با یک مدار مناسب نمایش داد (اما، طبیعتاً ممکن است دو مدار مختلف تابع واحدی را نشان دهند؛ در این‌صورت می‌گوییم که این دو مدار هم‌ارز هستند). در واقع، اولاً چون تابعی از $\{0,1\}^n$ به $\{0,1\}^m$ چیزی جز یک نوع گروهبندی m مختصهٔ آن، که توابعی از $\{0,1\}^n$ به $\{0,1\}$ هستند، نیست، می‌توان فرض کرد که $m = 1$ (علت این هم که غالباً در بررسی مدارها به مدارهایی باز می‌گردیم که تنها یک خروجی دارند همین است)؛ سپس از استقرا روی n استفاده می‌کنیم. به‌ازای $n = 1$، چهار تابع ممکن عبارت‌اند از 0، 1، x، $x \oplus 1$، که توسط مدارهایی که در شکل ۴ نشان داده شده‌اند محاسبه می‌گردند. اکنون باید ثابت کنیم که اگر این امر در مورد توابع n متغیره صادق باشد، در مورد توابعی که به $n+1$ متغیر بستگی دارند نیز صادق است. فرض می‌کنیم $f(x_1, x_2, \dots x_n)$ یک چنین تابعی باشد، که دو تابع n متغیرهٔ $f_0(x_2, \dots x_n) = f(0, x_2, \dots x_n)$ و $f_1(x_2, \dots x_n) = f(1, x_2, \dots x_n)$ به‌آن وابسته است؛ اگر دو تابع اخیر توسط مدارهای C_0 و C_1 محاسبه شوند، تابع $f = x_1 \otimes f_1 \oplus (1 \oplus x_1) \otimes f_0$ توسط مداری محاسبه می‌شود که در شکل ۵ نمایش داده شده است، و از آنجا نتیجهٔ مطلوب به‌دست می‌آید.

این مطلب را می‌توان به‌نحو دیگری نیز مطرح کرد، و آن اینکه هر تابعی از $\{0,1\}^n$ به $\{0,1\}$ را می‌توان به‌صورت یک چندجمله‌ای در هیأت $\mathbf{F}_2$ بیان کرد؛ تقریباً واضح است که این چندجمله‌ایها هم توابعی هستند که توسط مدارها محاسبه می‌شوند؛ مثلاً در مورد مدار شکل ۳،



شکل۴



شکل۵

در حالی که

$$v = (x \oplus y) \oplus z = x \oplus y \oplus z$$

همان‌طور که همه می‌دانند، چندجمله‌ای طبق تعریف برابر است با مجموعی از تک‌جمله‌ایها، چون با هیأتی سر و کار داریم که هر عنصر آن در رابطهٔ $x^2 = x$ (منظورم البته $x \otimes x = x$ است!) صدق می‌کند، فقط مجهولات درجهٔ اول را دخالت می‌دهیم؛ با این وصف می‌توانیم، به‌محض اینکه n مجهول را در دست داشته باشیم، 2^n تک‌جمله‌ای بسازیم. وقتی می‌خواهیم مقدار چندجمله‌ای $P(x_1, \dots x_n) = (1 \oplus x_1) \otimes \dots \otimes (1 \oplus x_n)$ را در یک نقطه حساب کنیم، بهتر است که با بسط آن برحسب 2^n تک‌جمله‌ای‌اش شروع نکنیم، چون مدت زمان لازم برای این کار، تابعی نمایی از n است؛ بنابراین، هریک از $1 \oplus x_i$ها را محاسبه می‌کنیم، و سپس حاصلضرب آنها را درنظر می‌گیریم. برای این کار تنها به تعدادی خطی عملیات احتیاج داریم (درواقع، $P(x_1, \dots x_n) = 1$ اگر تمام x_iها صفر باشند، در غیر این‌صورت $P(x_1, \dots x_n) = 0$)؛ این طرز محاسبه به‌صورت یک مدار درمی‌آید که بدون کم و کاست نمایشگر عبارت $P(x_1, \dots x_n) = (1 \oplus x_1) \otimes \dots \otimes (1 \oplus x_n)$ است، و در مورد چهار متغیر در شکل ۶ ترسیم شده است.

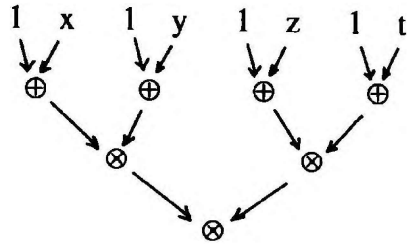
بنابراین، برای کارآمد بودن شیوهٔ محاسبه، بهتر است از نوشتن عبارات چندجمله‌ای به‌صورت متعارف اجتناب کنیم، و این چیزی است که تمام دانشجویان سال اول دانشگاهها می‌دانند. اما منظورم از «عبارت چندجمله‌ای» دقیقاً چیست، و تفاوت آن با مدارهای ما کدام است؟ برای جواب دادن به‌این سؤال، به‌این نکته توجه می‌کنیم که درهای این مدارها، که درصورت ورودی نبودن باید به‌سبب دوتایی بودنِ عملیات مربوطه دو پیکان دریافت کنند، این مکان را دارند که هر تعداد پیکان صادر کنند. می‌گوییم ظرفیت (خارجی) آنها نامحدود است. این توانایی به‌این شکل متجلی می‌شود که می‌توان از نتیجهٔ محاسبه‌ای که یک‌بار انجام شده است دوبار استفاده کرد و تکرار آن لزومی ندارد.

در میان مدارها، آنهایی را که تنها دارای یک خروجی هستند و ظرفیت آنها یک است، یعنی مدارهایی را که درهایشان فقط می‌توانند یک پیکان واحد صادر کنند، درنظر می‌گیریم. به آنها عبارتهای بولی خواهیم گفت، و خواننده خود می‌تواند دریابد که این عبارتها در واقع متناظرند با عبارات چندجمله‌ای (که به‌صورت مجموعی از تک‌جمله‌یها درنیامده‌اند) که به شکل سطری (پشت سرهم) نوشته شده‌اند. در این طرز نوشتن اگر دو زیرعبارتِ یکسان

$$P(x,y,z,t) =$$

$$((1\oplus x)\otimes(1\oplus y))\otimes((1\oplus z)\otimes(1\oplus t)) =$$

$$1 \oplus x \oplus y \oplus z \oplus t \oplus x\otimes y \oplus x\otimes z \oplus x\otimes t \oplus y\otimes z \oplus y\otimes t \oplus z\otimes t \oplus x\otimes y\otimes z \oplus x\otimes y\otimes t \oplus x\otimes z\otimes t \oplus y\otimes z\otimes t \oplus x\otimes y\otimes z\otimes t$$



شکل۶

در دو محل مختلف ظاهر شوند، فرمالیسم ریاضی ما را وادار می‌کند که آنها را تکرار کنیم. به شکل ۶ نگاه کنید، در این شکل ترجمهٔ کلمه به کلامه سبب شده است که به‌جای یک در با ظرفیت چهار، چهار در با نشانهٔ ۱ بیاوریم.

این را هم همین الآن بگوییم: نمی‌دانیم که آیا در حقیقت می‌توان به کمک مدارها توابع را به‌صورتی واقعاً کوتاهتر از عبارات نشان داد یا نه؛ این مسأله، مسألهٔ حل‌نشده‌ای در مبحث الگوریتم است، که در پاراگراف بعد مورد بررسی قرار خواهد گرفت. برای اینکه منظورمان را از کلمهٔ «کوتاهتر»، دقیقتر بیان کنیم، باید دو پارامتر عددی وابسته به مدار را تعریف کنیم: اولی اندازهٔ آن است، که عبارت است از تعداد درهایی که ورودی نیستند؛ دومی عمق آن است، که عبارت است از درازای (= تعداد پیکانهای) حداکثر راهی که از یک ورودی به یک خروجی می‌رود. مثلاً، اندازهٔ مدار شکل ۳ برابر است با ۵ و عمق آن ۳ است.

هر مداری، مانند C، که تنها یک خروجی دارد، آشکارا به یک عبارت هم‌ارز T به‌صورت زیر تبدیل می‌شود: با استقرا روی عمق C، یعنی p، شروع می‌کنیم: اگر $p = 0$، C یک ورودی می‌شود و قرار می‌دهیم $T = C$؛ اگر $p = q + 1$، خروجی C دو پیکان خود را از درهای π_0 و π_1 دریافت می‌کند (که ممکن است یکی باشند!): آن قسمت از C را که بالای در اول قرار دارد با C_0، و آن قسمت از C را که بالای در دوم است با C_1 نشان می‌دهیم؛ از آنجا که یک در C می‌تواند چند پیکان صادر کند، C_0 و C_1 مجزا نیستند. اکنون این دو مدار را، که عمق آنها حداکثر q است، به دو جملهٔ هم‌ارز T_0 و T_1 تبدیل می‌کنیم. این کار به ما اجازه می‌دهد که $T_0 \oplus T_1$ یا $T_0 \otimes T_1$ را جایگزین C کنیم، بسته به اینکه C جمعی باشد یا ضربی (نگاه کنید به شکل ۷).

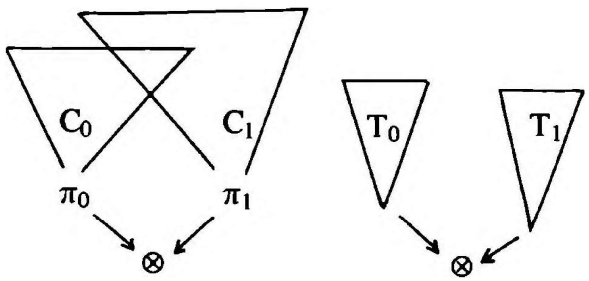
این تبدیل عمق را تغییر نمی‌دهد، اما اثر فاجعه‌باری بر اندازه دارد. در واقع این تبدیل عبارت است از اینکه در هر مرحله به‌جای C_0 و C_1 دو نسخهٔ متناظر و مجزا از هم را قرار دهیم، و این کار، اندازه را هر بار تقریباً دو برابر می‌کند، و در آخر اندازهٔ T نسبت به اندازهٔ C نمایی خواهد شد!

مسألهٔ حل‌نشده‌ای که اکنون مطرح شد، این است که آیا تبدیل دیگری وجود دارد که ظریفتر باشد و همواره یک مدار را، بدون افزایش خارق‌العادهٔ اندازه، به یک عبارت هم‌ارز تبدیل کند؟ اکثر متخصصین برآن‌اند که جواب منفی است، اما نمی‌دانند چگونه آن را ثابت کنند.

این بخش را با دو مثال از تبدیلهایی که در آنها اندازه و عمق فقط در اعداد ثابتی ضرب می‌شوند خاتمه می‌دهم.

اولین تبدیل بسیار ساده است و مربوط می‌شود به عملیات پایه بر روی دو نماد ۰ و ۱. در تعریف مدارهای بولی، بیشتر مرسوم است که «عملگرهای بولی»، یعنی $x \wedge y = x \otimes y$، $x \vee y = x \oplus y \oplus x \otimes y$، $\neg x = 1 \oplus x$ را در نظر بگیریم. از آنجا که هر یک از آنها برحسب جمع و ضرب به پیمانهٔ ۲ بیان می‌شود، می‌توان هر مداری را که دارای ورودیهای $\neg$، $\vee$، و $\wedge$ باشد، به‌سادگی، با قرار دادن یک مدار کوچک به‌جای هر ورودی بولی، به یک مدار با ورودیهای $\oplus$ و $\otimes$ تبدیل کرد. این کار را می‌توان در جهت عکس نیز انجام داد زیرا $x \otimes y = x \wedge y$ و $x \oplus y = (\neg x \wedge y) \vee (x \wedge \neg y)$. ما ترجیح می‌دهیم که مدارها را از طریق عملیاتی معرفی کنیم که ماهیت حسابی دارند و ریاضیدانان با ویژگیهای آنها آشناترند تا با ویژگیهای عملیات بولی؛ اما خواهیم دید که با این کار تغییر چندانی پدید نمی‌آید.

مثال دوم بسیار ظریفتر است. این مثال را هوور[1]، کلاو[2]، و پی‌پنجر[3] ابداع کرده‌اند (نگاه کنید به اثباتی که در [۹]، قضیهٔ ۶.۲، آمده است) و هر مداری مانند C با اندازهٔ t و عمق p را به مدار هم‌ارزی مانند C^* تبدیل می‌کند که ظرفیت آن دو است (از یک ورودی فقط صفر، یک، یا دو پیکان می‌تواند خارج شود)، اندازهٔ آن t^* است که از $3t$ کوچکتر است و عمق آن p^* است که از $2p$ کوچکتر است!



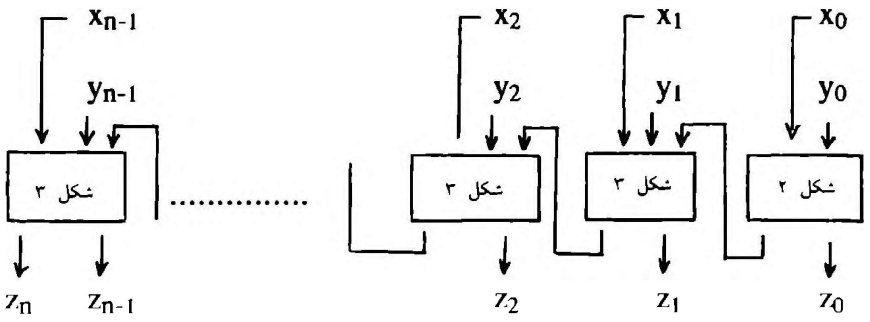
شکل۷

۳. الگوریتم و رشتهٔ مدارها

نمایش مداریِ تابع بولی نحوهٔ محاسبهٔ آن را بهتر از نمایش سطریش نشان می‌دهد. اندازهٔ مدار، که عبارت است از تعداد کل عملیات اصلیی که باید انجام شود، همان زمان متوالی لازم برای محاسبه است. اما زمان موازی، عمق مدار است زیرا، اگر در کنار هر دری یک نگهبان بگُماریم، او نمی‌تواند قبل از رفقایش که بالا هستند عمل کند.

از اینجا تعریفی به‌دست می‌آید، که شاید برخی جزئیات آن (که تأثیری

1. Hoover 2. Klawe 3. Pippenger



شکل۸

بر مفهوم «محاسبهٔ قابل اجرا در یک زمان چندجمله‌ای» ندارد) محل بحث باشد، اما به هر حال تعریفی است مُتقن از مفهوم زمانِ اختصاص داده شده به یک الگوریتم. در اینجا آنچه را که در بخش اول در مورد زمان لازم برای جمع دو عدد گفتیم ثابت خواهیم کرد.

از این‌رو فرض می‌کنیم که می‌خواهیم اعداد $x_0 \dots x_{n-1}$ و $y_0 \dots y_{n-1}$ را که توسط دو رشتهٔ دوتایی با اندازهٔ n نمایش داده می‌شوند با یکدیگر جمع کنیم. مداری که جمع را انجام می‌دهد باید $2n$ ورودی متناظر با تمام این ارقام داشته باشد؛ و نیز باید $n+1$ خروجی برای نوشتن $z_0 \dots z_n$، یعنی حاصلجمع وجود داشته باشد.

اکنون مداری را که متناظر با جمع دبستانی است، توصیف می‌کنیم. ابتدا x_0 و y_0 را با یکدیگر جمع می‌کنیم که در نتیجه آخرین رقم حاصلجمع و یک باقیماندهٔ r_1 به‌دست می‌آید: این دقیقاً همان کاری است که مدار شکل ۲ انجام می‌دهد. سپس، باید x_1، y_1 و r_1 را با یکدیگر جمع کنیم تا z_1 و ماندهٔ r_2 به‌دست آید: این کاری است که مدار شکل ۳ انجام می‌دهد. حالا باید دوباره شروع کنیم، و برای این کار باید نسخه‌هایی از مدار شکل ۳ را در کنار هم قرار دهیم. نگاه کنید به طرح سوارکردن مدار در شکل ۸.

پیدا کردن اندازهٔ مدار ساده است: یک نسخه از شکل ۲ را که دارای دو دَر عملیاتی است در کنار $n-1$ نسخه از شکل ۳ که اندازهٔ آن ۴ است قرار می‌دهیم. بنابراین اندازهٔ کل می‌شود $4n - 4 + 2 = 4(n-1)$؛ از این‌رو زمان متوالی خطی است و کران بالای آن $4n$ است.

پیدا کردن عمق هم خیلی پیچیده‌تر از این نیست: یک نسخه از مدار شکل ۲، به عمق ۱، و $n-1$ نسخه از مدار شکل ۳، به عمق ۳، را بر روی هم قرار داده‌ایم (هر چند که طرح شکل ۸ افقی است!)، به‌طوری‌که کران بالای عمق کل $3(n-1)+1$ است، اما، اگر با دقت بیشتری نگاه کنیم (این طرح را برای، مثلاً، $n=4$ رسم کنید)، می‌بینیم که «مسیر بُحرانی» متعلق به مانده‌هاست، و این راه از هر شکل ۳ تنها از طریق دو پیکان عبور می‌کند. به‌عبارت دیگر، یکی از طولانی‌ترین راههای مدار، راهی است که x_0 را به z_n وصل می‌کند، و اندازهٔ آن $2(n-1)+1 = 2n-1$ است.

بنابراین، این الگوریتم که یک الگوریتم متوالی نمونه است، با زمان موازی خطی کار می‌کند، و برای کسی که عجله دارد رضایت‌بخش نیست. برای توصیف مدار اخیقار، باید ابتدا به خوانندهٔ (ناآشنا) توضیح دهم که زنان حرمسرا به چه کار می‌آیند. گفتیم که اگر ماندهٔ اخیقار ۰ باشد، آنها عدد شامیران را انتخاب می‌کنند، و اگر مانده ۱ باشد، عدد ایشتار را انتخاب می‌کنند. به‌عبارت دیگر آنها تابع $s(x,y,z)$ را که به‌صورت $s(x,y,0)=x$ و $s(x,y,1)=y$ تعریف می‌شود حساب می‌کنند. این تابع طبیعتاً گزیننده نامیده می‌شود، و به‌وسیلهٔ مدار شکل ۹ نمایش داده شده است.

در ضمن این را هم مشاهده می‌کنیم که به کمک گزیننده می‌توان جمع و ضرب را به پیمانهٔ دو بیان کرد، زیرا: $x+y = s(x, s(1,0,x), y)$ و $x \oplus y = s(0,x,y)$. از این‌رو می‌توان گزیننده را به‌عنوان تنها عمل پایه در مدارها درنظر گرفت. بنابراین لازم است که هر دَرکه ورودی نیست سه پیکان دریافت کند، و این باید به‌ترتیب باشد زیرا عمل دیگر جابه‌جایی نیست. این امر زیاد هم غیرطبیعی نیست، زیرا محاسبهٔ چیزی جز یک رشته انتخاب نیست: برحسب اینکه یک ۰ و یا یک ۱ پیش‌رو باشد، این یا آن عمل انجام می‌شود. اما ما دو عمل $+$ و $\oplus$ را به‌آن ترجیح داده‌ایم زیرا این‌دو عمل ویژگیهای جبری جالب توجه‌تری دارند. همان‌طورکه گفتیم این انتخاب اهمیت حیاتی ندارد، زیرا با تغییر پایهٔ عملیات اصلی تنها کاری که انجام می‌شود این است که اندازه و عمق مدارها در یک، عدد ثابت ضرب می‌شود.

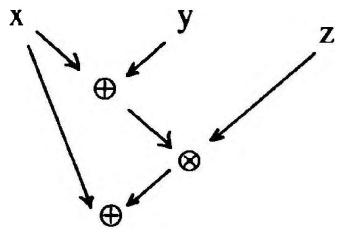
بنابراین فرض می‌کنیم A_m مداری باشد که روش اخیقار برای جمع دو عدد 2^m رقمی را توصیف‌می‌کند. عمق مدار را با p_m نشان‌می‌دهیم. به‌ازای $m=0$، داریم $2^m = 1$، و مدار شکل ۳ مناسب است: $p_0 = 3$. شکل ۱۰ [زنان] حرمسرای اخیقار را نشان می‌دهد، که مشغول جمع زدن دو عدد 2^{m+1} رقمی هستند. در سمت راست، خود اخیقار مشغول جمع زدن دو نیمهٔ سمت راست است (دو عدد 2^m رقمی و یک ماندهٔ ۰)؛ او 2^m رقم اول حاصلجمع به‌علاوهٔ یک، مانده را پیدا می‌کند؛ در سمت چپ شامیران و ایشتار را مشاهده می‌کنیم، که مشغول جمع زدن دو نیمهٔ سمت چپ هستند و هر یک با فرض متفاوتی در مورد ماندهٔ اخیقار کار می‌کند. وقتی هر سه کارشان را تمام کنند، $2^m + 1$ زن حرمسرا (که فقط یکی را نشان داده‌ایم) انتخابشان را می‌کنند، و $2^m + 1$ رقم سمت چپ حاصلجمع را می‌نویسند.

چون عمق s برابر ۳ است، می‌بینیم که $p_{m+1} = p_m + 3$، یا $p_m = 3(m+1)$. عمق در واقع لگاریتمی است، زیرا اخیقار برای جمع زدن دو عدد n رقمی، با قرار دادن چند صفر در سمت چپ آنها را کامل خواهد کرد تا کاملاتی با طول 2^m به‌دست آورد، که m کوچکترین عدد ممکن است؛ بنابراین $m \leqslant 1 + \log n$؛ و این کار را در زمانی (موازی) که کران بالای آن $3 \log n + 6$ است انجام می‌دهد.

$$s(x,y,0) = x$$

$$s(x,y,1) = y$$

$$s(x,y,z) = (x \oplus y) \otimes z \oplus x$$



شکل۹

در این میان، بیایید اندازهٔ مدار A_m یعنی t_m را محاسبه کنیم. داریم $t_0 = 4$. از آنجا که هر گزیننده متناظر است با یک مدار با اندازهٔ ۳ (نگاه کنید به شکل ۹)، داریم $t_{m+1} = 3t_m + 3(1 + 2^m)$؛ پس رابطهٔ بازگشتی زیرا را به‌دست قرار می‌دهیم $u_m = 3^m t_m$؛ می‌آوریم: $u_0 = 4$، $u_{m+1} = u_m + (1/3)^m + (2/3)^m$؛ و از آنجا $u_m = 2 + 1 + 1/3 + \cdots + (1/3)^m + 1 + 2/3 + \cdots + (2/3)^m$. برای محاسبهٔ مقدار دقیق u_m کافی است فرمول حاصلجمع یک تصاعد هندسی را به‌یاد بیاوریم، و توجه داشته باشیم که این مقدار همواره از حد خود، یعنی $2 + (1 - 1/3)^{-1} + (1 - 2/3)^{-1} = 6{,}5$، کوچکتر خواهد بود. بنابراین $t_m < (6{,}5)3^m$. به‌جای m قرار می‌دهیم $\log n + 1$، و می‌بینیم که اندازهٔ مداری که دو عدد با طول n در عمق لگاریتمی را جمع می‌کند از $(6{,}5)3^1 + \log n = (3 \times 6{,}5)2^{\log n \times \log 3} < 20n^{\log 3} < 20n^2$ بیشتر نیست (چون $4 < 3 < 2$، پس $2 > \log 3 > 1$؛ فراموش نکنید که لگاریتم در مبنای دو است!). برای اینکه یک زمان موازی لگاریتمی داشته باشیم، مجبور شدیم مقداری انرژی هدر دهیم، و در نتیجه دیگر اندازهٔ مدار برحسب n خطی نیست؛ اما بازهم چندجمله‌ای باقی می‌ماند. این نهایت حکمت اخیقار را می‌رساند که برای اینکه جمعهایش را انجام دهد به تعدادی نمایی زن حرمسرا احتیاج ندارد.

۴. مسأله موازی بودن

نکته‌ای هست که دربارهٔ آن زیاد بحث نمی‌کنیم: هر الگوریتمی را می‌توان به‌طور طبیعی با یک مدار نمایش داد، که اندازه و عمق آن تقریبهای معقولی هستند از زمان متوالی و زمان موازیِ لازم برای اجرای آن الگوریتم. امیدوارم که خواننده از طریق مثال جمع به اندازهٔ کافی در این مورد متقاعد شده باشد.

دقیقتر بگوییم: هر الگوریتمْ رشته‌ای نامتناهی از مدارهای C_1، ...، C_n، ... به‌دست می‌دهد، که جملهٔ nام آن متناظر است با محاسبه‌ای که روی داده‌های با طول n انجام می‌شود (بنابراین C_n دارای n متغیر ورودی است). در واقع، الگوریتم جمع تنها دو عدد ده رقمی یا دو عدد ده هزار رقمی را با یکدیگر جمع نمی‌کند بلکه، دو عدد n رقمی را با یکدیگر جمع می‌کند، که n دلخواه است.

برای اینکه بیان مطلب را ساده کنیم، از این به بعد فرض می‌کنیم که مدارها فقط یک خروجی دارند، یعنی جواب الگوریتم همیشه یا ۱ است و یا ۰، یعنی یا «آری» و یا «خیر». به‌عبارت دیگر، الگوریتمْ تعلق دادهٔ مفروض μ را، که یک کلمهٔ دوتایی دلخواه است، به یک مجموعهٔ معین X می‌آزماید (اگر جواب در مورد دادهٔ μ، ۱ باشد، می‌گوییم الگوریتم μ را می‌پذیرد، و اگر جواب ۰ باشد می‌گوییم μ را طرد می‌کند). تحت این شرایط، به مجموعهٔ X می‌گوییم مسأله (به‌عبارت دیگر، معنی «مسأله» چیزی جز «مجموعهٔ کلمات» نیست!). مثال یک مسأله: x_1، ...، x_n را داریم، سومین رقم اعشار عددی را که از جمع نیمهٔ اول این کلمه با نیمهٔ دوم آن به‌دست می‌آید پیدا کنید!

الگوریتمی که شایستهٔ نام الگوریتم باشد، روی داده‌های با طول دلخواه محاسبات را یکسان انجام می‌دهد، به‌این معنی که رشتهٔ $\{C_n\}$ از مدارها رفتار نسبتاً منظمی از خود نشان می‌دهند. در اینجا نمی‌خواهیم به تحلیل این موضوع بپردازیم زیرا اولاً چنین تحلیلی آن‌قدرها هم ساده نیست، و ثانیاً این تحلیل اثری بر روی دو مسألهٔ مهمی که در اینجا مورد بحث ما هستند و به مقدار امکانات لازم برای اجرای یک محاسبه مربوط می‌شوند، ندارد. بنابراین رشته‌های دلخواه C_n را می‌پذیریم، با این شرایط که اندازه و یا عمق آنها از توابع مشخصی از n تجاوز نکند و این رشتهٔ تعریف‌کننده چیزی است که متخصصین امر، آن را «الگوریتمهای نایکنواخت» می‌گویند.

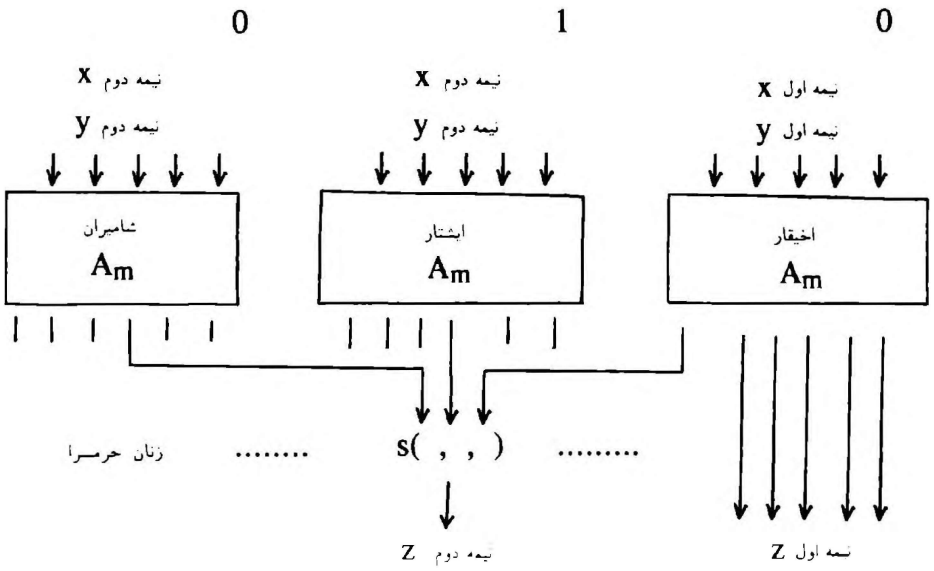
بدین‌ترتیب می‌گوییم مسألهٔ X از نوع $\mathbb{P}$ است هرگاه توسط رشته‌ای چون $\{C_n\}$ از مدارها حل شود که اندازهٔ آن، t_n، از یک چندجمله‌ای برحسب n تجاوز نکند. این ردهٔ $\mathbb{P}$ نوع نایکنواخت ردهٔ Pِ مسائلی است که در یک زمان چندجمله‌ای حل می‌شوند. این رده بیشتر به‌نام P/poly معروف است و در اینجا بیش از این درباره‌اش بحث نمی‌کنیم. همین‌طور، نوع نایکنواختِ ردهٔ نیک، به روایت کوک* را درنظر خواهیم گرفت: مسأله را $\mathbb{NC}$ گوییم هرگاه توسط رشته‌ای از مدارها حل شود که عمق آن p_n از تابعی خطی از $\log n$ تجاوز نکند.

تقریباً واضح است (نگاه کنید به حرمسرای اخیقار) که ردهٔ $\mathbb{NC}$ مشمول ردهٔ $\mathbb{P}$ است. در واقع، از آنجا که هر دری تنها دو پیکان دریافت می‌کند، مداری با عمق p بیش از 2^p در نخواهد داشت (از پایین شروع کنید)، و در نتیجه اندازهٔ مداری به عمق $A \log n$ کوچکتر از n^A است.

چیزی که مشکل ایجاد می‌کند، عکس قضیه است: نمی‌دانیم که آیا $\mathbb{P} = \mathbb{NC}$ یا نه، یعنی نمی‌دانیم که آیا می‌توان هر مسأله‌ای را که قابل حل در یک زمان متوالی چندجمله‌ای است، در یک زمان موازی لگاریتمی نیز حل کرد یا نه. سؤال این است که آیا $\mathbb{P} = \mathbb{NC}$؟ این همان مسألهٔ موازی بودن است.

احتمالاً تساوی برقرار نیست، ما نمی‌توان این موضوع را ثابت کرد. از طرف دیگر متخصصان مبحث الگوریتم برای پیدا کردن الگوریتمهای با عمق

* Nick's Classe de Cook



شکل ۱۰

لگاریتمی برای اکثر مسائل جاری تلاش فراوانی به‌خرج داده‌اند. این را در مورد جمع دو عدد مشاهده کردیم؛ در مورد ضرب، مسأله کمی پیچیده‌تر است [۹].

برای فهم بهتر مطلب، دو حکم را اثبات می‌کنیم.

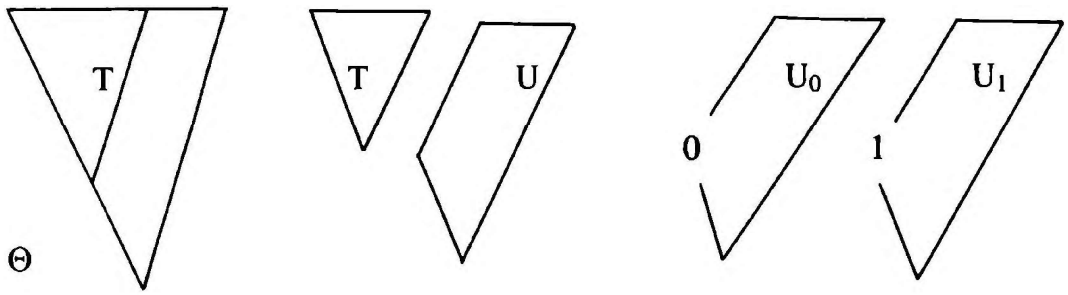
لِم قطری کردن. $\mathbb{P} = \mathbb{NC}$ اگر و تنها اگر عدد ثابتی مانند A وجود داشته باشد به‌طوری‌که هر مداری با اندازهٔ t، که دارای تنها یک خروجی است، هم‌ارز باشد با یک مدار (یا حتی یک عبارت!) که عمق آن از $A \log t$ تجاوز نکند.

اثبات. وجود A ایجاب می‌کنند که $\mathbb{P} = \mathbb{NC}$. زیرا $\log(n^B) = B \log n$. برعکس، فرض می‌کنیم A وجود ندارد یعنی به‌ازای هر عدد صحیح m، مداری مانند C_m داریم با اندازهٔ t_m، که هم‌ارزی با عمق کوچکتر از $m \log(t_m)$ ندارد. می‌توانیم، شاید با اغماض از برخی C_m ها، فرض کنیم که رشتهٔ $\{t_m\}$ از اندازه‌ها اکیداً صعودی است. تعدادی درهای ورودی C_m از $t_m + 1$ کوچکتر است (این را می‌توان از طریق استقرا بر روی اندازه و با حذف یک ورودی ثابت کرد)، و تعداد متغیرهای ورودی v_m نیز همین‌طور. اکنون مسألهٔ X زیر را درنظر می‌گیریم که فقط شامل کلماتی است که اندازهٔ آنها به‌صورت $n = 1 + t_m$ است. یک چنین کلمه‌ای، μ، در X است اگر و فقط اگر C_m، کلمه‌ای را که از v_m حرف اول آن تشکیل می‌شود بپذیرد. در این‌صورت X در یک زمان متوالی خطی حل می‌شود و نه در یک زمان موازی لگاریتمی. **فهوالمطلوب**

قضیهٔ اسپایرا. هر عبارت بولی با اندازهٔ t هم‌ارز است با یک مدار (یا یک عبارت!) با عمقی کوچکتر از $4 \log t + 1$.

اثبات (نگاه کنید به شکل ۱۱). اثبات از طریق استقرا بر روی t است. گزاره به‌ازای $t = 1$، که عمق عبارت ۱ است، درست است. بنابراین عبارت Θ با اندازهٔ t را که مقدار آن اقلاً ۲ باشد درنظر می‌گیریم. یک زیرعبارت مینیمال با اندازه‌ای اکیداً بزرگتر از $t/2$ استخراج می‌کنیم. خروجی آن دو پیکانش را از یک یا دو زیر عبارت که اندازهٔ آنها کوچکتر از و یا مساوی با $t/2$ است دریافت می‌کند، که آن‌هم اکیداً کوچکتر است از t. برطبق فرض استقرا، T هم‌ارز است با عبارتی مانند T^* با اندازهٔ $1 + 4 \log t - 3 = 1 + 4 \log(t/2) + 1$. اگر T کل عبارت Θ باشد، مطلوب حاصل است؛ در غیر این‌صورت، عبارتی را که از جایگذاری فقط یک درِ ورودی به‌جایِ T در Θ حاصل می‌شوند با U_0 و U_1 نشان می‌دهیم؛ U_0 عبارتی است که در آن درِ ورودیِ جایگزینِ T با ۰ نشانگذاری شده است، و به همین نحو U_1. اندازهٔ این دو جمله از $t/2 + 1$ (در واقع از $(\frac{1}{2}(t/2 + 1)!$) کوچکتر است. آنها را مانند T به یک یا دو زیر عبارت با اندازهٔ کوچکتر از $t/2$ تجزیه می‌کنیم و در مورد آنها از فرض استقرا استفاده می‌کنیم. بنابراین U_0 و U_1 را هم می‌توان با دو عبارت U_0^* و U_1^* به عمق $4 \log(t/2) + 2$ تعویض کرد. عبارت Θ هم‌ارز عبارت $\Theta^* = s(U_0^*, U_1^*, T^*)$ است، که در آن s گزینده‌ای است که از طریق مداری به عمق ۳ بیان می‌شود. بنابراین عمق Θ^* (حداکثر) برابر است با $4 \log t + 1 = 4 \log(t/2) + 2 + 3$. **فهوالمطلوب**

نتیجه. $\mathbb{P} = \mathbb{NC}$ به‌این معنا هم هست که می‌توان مداری مانند C را به یک عبارت هم‌ارز T تبدیل کرد؛ اندازهٔ T هم کوچکتر از یک چندجمله‌ای برحسب اندازهٔ C است. در واقع، از طرفی دیگر اگر $\mathbb{P} = \mathbb{NC}$، مدار C با اندازهٔ t تبدیل می‌شود به یک مدار، یا حتی یک عبارت T، به عمق $A \log t$، که اندازهٔ آن کوچکتر است از $2^{A \log t} = t^A$. از طرف دیگر، اگر جملهٔ T با اندازهٔ t^A با C هم‌ارز باشد، طبق قضیهٔ اسپایرا به یک جملهٔ هم‌ارز T^* به عمق $4A \log t + 1 = 4 \log(t^A) + 1$ تبدیل می‌شود. بنابراین مسألهٔ



شکل۱۱

موازی بودن هم‌ارز است با مسألهٔ قدرتِ توصیف‌کنندگی عبارتها و مدارها، در مقایسه با یکدیگر.

۵. مسألهٔ مهم دیگر

در ابتدای این مقاله شما را به دبستان برگرداندم و از شما خواستم که جمع بزنید. اکنون به دبیرستان می‌برمتان، و دستگاههای معادلات خطی را حل خواهیم کرد.

یک داده برای مسألهٔ LIN که اکنون بررسی خواهیم کرد عبارت است از دستگاهی مرکب از n معادلهٔ خطی n مجهولی که ضرایب آن در $\mathbf{F}_2$ هستند؛ این دستگاه به‌صورت زیر است:

$$a_{11} \odot x_1 \oplus \ldots \oplus a_{1n} \odot x_n = b_1$$

$$a_{n1} \odot x_1 \oplus \ldots \oplus a_{nn} \odot x_n = b_n$$

که در آن a_{ij} ها و b_i ها همگی ۰ هستند یا ۱!

عدد صحیح n شاخص معقولی از اندازهٔ دستگاه است زیرا که اندازهٔ دستگاه متناظر است با $n^2 + n$ ضریب دوتایی داده شده.

مسألهٔ LIN این است که بدانیم آیا این دستگاه جوابی در $\mathbf{F}_2$ دارد یا ندارد، یعنی جوابی که از x_iهایی تشکیل شده باشد که مقدار آنها همگی ۰ و یا ۱ باشد.

روش اول: فقط تعدادی متناهی n تایی بولی وجود دارند که می‌توانند جواب دستگاه باشند. همگی آنها را یکی یکی می‌آزماییم و مشاهده می‌کنیم که آیا جوابی در میان آنها وجود دارد یا خیر! اینکه آیا این روش شایستهٔ نام الگوریتم هست یا نه، محل بحث است. نقص این روش در این است که به تعدادی نمایی آزمایش احتیاج دارد.

می‌توانیم دترمینانها را حساب کنیم که هم در $\mathbf{F}_2$ و هم در هر هیأت دیگری معتبر است؛ در روش کرامر فرض بر این است که می‌توان یک زیردترمینان ماکسیمال غیر صفر از دستگاه [معادلات] استخراج کرد. از این نکته بگذریم و توجه خود را به محاسبهٔ یک دترمینان از مرتبهٔ n معطوف داریم.

دترمینان دستگاه ما دارای فرمول زیر است:

$$D = \Sigma a_{1,\sigma 1} \ldots a_{n,\sigma n} = \Sigma a_{\sigma 1,1} \ldots a_{\sigma n,n}$$

که جمعبندی روی تمام جایگشتهای σی $\{1, \ldots, n\}$ صورت می‌گیرد. لازم نیست نگران علامت جایگشتها باشیم زیرا $1- = 1$. ضرب کردن n عامل که مقدار هر یک ۰ و یا ۱ است، بیشتر از n واحد زمان وقت نمی‌گیرد. کاری که زیاد وقت می‌گیرد انجام دادن $n!$ ضرب، و سپس جمع کردن حاصلضربها با هم است. زمان لازم برای این کار تابعی است که از تابع نمایی هم بزرگتر است.

باقی می‌ماند روش حذفی گاوس. این روش عبارت از این است که به‌جای دستگاه [معادلات]، دستگاه هم‌ارزی به‌دست آوریم که مثلث‌شکل است و جواب را فوری به‌دست می‌دهد.

شروع کار بدین‌گونه است: اولین مجهول را درنظر می‌گیریم؛ اگر در هیچ یک از معادلات ظاهر نشده باشد آن را یک پارامتر محسوب می‌کنیم، در غیر این‌صورت یکی از معادلاتی را که شامل آن مجهول می‌شود در سطر اول، که سطر محور است، قرار می‌دهیم؛ سپس این سطر را به تمام سطور دیگری که شامل این مجهول می‌شود اضافه و یا از آنها کم می‌کنیم. به‌این ترتیب دستگاهی به‌دست می‌آوریم که تشکیل شده از یک معادله که مجهول اول را برحسب مجهولات دیگر بیان می‌کند، و $n-1$ معادلهٔ دیگر با $n-1$ مجهول که مجهول اول از آنها حذف شده است. این دستگاه با دستگاه قبل هم‌ارز است زیرا از ترکیب سطرهای دستگاه قبل به‌دست آمده است. ممکن است این حالت پیش بیاید که اولین عضو یکی از این معادلات جدید ناپدید شود؛ اگر در مورد دومین عضو نیز چنین باشد، شرط $0 = 0$ به‌دست می‌آید، و می‌توان سطر مورد نظر را حذف کرد؛ اگر دومین عضو ناپدید نشود، شرط $1 = 0$ به‌دست می‌آید و دستگاه غیرممکن [ممتنع] است. همین اعمال را برای حذف دومین، و سپس سومین مجهول و غیره تکرار می‌کنیم. اگر به حالت غیرممکنی برنخوریم، بالاخره یک دستگاه مثلثی به‌دست خواهد آمد، که در آن برخی از مجهولات، که به‌صورت پارامتر درنظر گرفته شده بودند، به عضو دوم، منتقل شده‌اند. اگر به شرط $1 = 0$ برنخوریم، دستگاه جواب دارد.

اکنون حداکثر زمان محاسبهٔ متوالی را، بدون دقت زیاد، پیدا می‌کنیم (و خوانندهٔ نابارومان را دعوت می‌کنیم که خود مدار متناظر با این الگوریتم را رسم کند!). برای جدا کردن سطر اول از بقیهٔ سطرها باید $n+1$ عمل انجام داد؛ $n-1$ سطر دیگر نیز وجود دارند؛ بنابراین برای حذف اولین مجهول،

$(n+1)(n-1) < n^2$ عمل لازم است. از آنجا که باید n مجهول حذف شوند، تعداد عملیات لازم برای مثلثی‌سازی از مرتبهٔ n^3 است، یعنی زمان لازم از درجهٔ سوم n است، و ما طبق قرارداد و یا بنا به تجربه این زمان را عملی می‌دانیم!

حال، پس از یادآوری مطالب بالا دستگاههایی درنظر می‌گیریم که از معادلات چندجمله‌ای با ضرایب متعلق به $\mathbf{F}_2$ تشکیل شده‌اند و دیگر به معادلات خطی اکتفا نمی‌کنیم. خانواده‌ای متشکل از n معادله به‌صورت $P(x_1, \ldots, x_n) = 0$ را، که در آن P یک چندجمله‌ای است که به‌صورت مجموع چند تک‌جمله‌ای نوشته می‌شود، یک دستگاه معادلات چندجمله‌ای با اندازهٔ (حداکثر) n می‌گوییم؛ تعداد کل مجهولات دستگاه حداکثر n تاست. عدد صحیح m، در مورد چندجمله‌ایها، محک واقعاً خوبی است برای اندازهٔ دستگاه.

مسئلهٔ POL عبارت از این است که بدانیم آیا چنین دستگاهی در $\mathbf{F}_2$ جواب دارد یا نه.

سؤال مهم دوم که به‌صورت «آیا P = NP؟» یا دستکم «آیا $\mathbb{P} = \mathbb{NP}$؟» مطرح می‌شود این است که بدانیم آیا الگوریتمی برای حل POL در یک زمان (متوالی) چندجمله‌ای وجود دارد یا نه.

البته می‌توان تمام مقادیر ممکن مجهولات را آزمود، اما این کار به زمانی نمایی احتیاج دارد. بیایید روش حذف را بیازماییم. قرار می‌دهیم $x = x_1$، $y = (x_2, \ldots, x_n)$. هر معادله‌ای به‌صورت $A(y) \odot x \oplus B(y) = 0$، نوشته می‌شود که A و B در $[y]$ $\mathbf{F}_2$ هستند. به‌ازای هر y که $A(y) = 1$، این معادله دارای جواب یکتای $x = B(y)$ است؛ در حالی که اگر $A(y) = 0$، مقدار x هرچه باشد، باید داشته باشیم $B(y) = 0$. به‌عبارت دیگر، معادله دارای جواب است اگر و تنها اگر معادلهٔ زیر دارای جواب باشد:

$$(1 \oplus A(y)) \odot B(y) = 0 \qquad (i)$$

هرگاه چند معادله داشته باشیم، باید این مطلب را که مقادیر به‌دست آمده برای x برابر هستند بیان کنیم، و این منجر می‌شود به

$$A(y) \odot A'(y) \odot (B(y) \oplus B'(y)) = 0 \qquad (ii)$$

بنابراین اگر تمام معادلاتی را که به شکل (i) و نیز تمام آنهایی را که به شکل (ii) هستند بنویسیم، دستگاهی به‌دست خواهیم آورد که متغیر x از آن حذف شده است، و دارای یک جواب است اگر و تنها اگر دستگاه اولیه جواب داشته باشد. با تکرار این عمل، دستگاهی به‌دست می‌آوریم بدون متغیر، که تنها درصورتی سازگار است که شامل $0 = 1$ باشد!

بله، اما این کار عملی نیست، زیرا زمان لازم به‌سرعت و با شدت افزایش می‌یابد، به دو دلیل. اولاً باید، برطبق قراردادهایمان، چندجمله‌ایها را به‌صورت مجموعی از تک‌جمله‌ایها بنویسیم، به‌عبارت دیگر باید ضربها را در معادلاتی به‌شکل (i) و (ii) انجام دهیم. تکرار این عمل مقداری وقت لازم دارد. ثانیاً، پس از اولین حذف، درحدود n^2 معادله از نوع (ii) و پس از حذف دوم تقریباً n^3 معادله خواهیم داشت ... و الی آخر؛ بنابراین خیلی زود با تعدادی نمایی معادله مواجه خواهیم شد! (برعکس حالت خطی، که در آن حذف گاوسی تعداد معادلات را افزایش نمی‌دهد.)

«P = NP?» شرارآمیزترین مسألهٔ حل‌نشدهٔ مبحث الگوریتم است. پانزده سالی است که این مسألهٔ مشغلهٔ فکری تمام متخصصین بوده است و جملگی، به استثنای چند آشوبگر، متفق‌القول‌اند که جواب منفی است. بنابراین بجاست که از خود بپرسیم اهمیت NP در چیست، و چرا حل POL این‌قدر مهم است.

سعی خواهم کرد در چند کلمه جوابتان را بدهم. فرق بین حذف مجهول بین دو معادلهٔ خطی $x \oplus L(y) = 0$ و $x \oplus L'(y) = 0$، و حذف x بین دو معادلهٔ چندجمله‌ای $A(y) \odot x \oplus B(y) = 0$ و $A'(y) \odot x \oplus B'(y) = 0$، در این است که در مورد اول انشعاب وجود ندارد، در حالی که در مورد دوم، باید، برحسب اینکه $A(y)$ صفر است یا نه، چندین امکان را بررسی کرد: به‌این علت است که الگوریتم حذف، معادلاتی به‌شکل (ii) به‌وجود می‌آورد، که بالا رفتن تعداد آنها فوق‌العاده زیاد می‌شود.

الگوریتم NP، الگوریتمی است که باید در هر مرحله ازکارکردش، چندین امکان را بررسی کند؛ هر شاخه ازکارکرد به یک زمان چندجمله‌ای احتیاج دارد، اما باید تعدادی نمایی از شاخه‌ها را بررسی کند؛ به محض اینکه یکی از شاخه‌ها اجازه دهد الگوریتم جواب مثبت می‌دهد. به‌عبارت دیگر اگر شاخهٔ درست را حدس بزنیم، جواب مسأله چندجمله‌ای می‌شود (در اینجا، در هر مرحله باید معادلهٔ «محوری»، مانند $A(y) = 0$، را که حذف x را ممکن می‌سازد حدس زد. خوب اگر این‌طور است، یکباره می‌توان جواب را حدس زد). گزارهٔ P = NP به‌این معناست که هر مسأله‌ای که به‌این ترتیب از طریق حدس زدن حل شود می‌تواند از طریق دیگر بدون حدس زدن نیز در یک زمان (متوالی) چندجمله‌ای حل شود. گزاره $\mathbb{P} = \mathbb{NP}$ نیز همان است اما به‌صورت نایکنواخت.

NP و سؤال بالا ازاین نظر اهمیت دارند که حاوی تعداد باور نکردنی مسألهٔ الگوریتمی جالب توجه‌اند، که خیلی دوست داریم برای آنها الگوریتمهای چندجمله‌ای داشته باشیم. اما چنین چیزی رؤیای زیبایی بیش نیست.

اینکه اهمیت مسألهٔ POL در ردهٔ NP تا این اندازه حیاتی است، به‌این علت است که تمام مسائل در ردهٔ اخیر توسط یک الگوریتم چندجمله‌ای به POL تبدیل می‌شود (می‌گوییم که POL، NP-تمام است؛ $\mathbb{NP}$-تمام نیز هست)؛ به‌عبارت دیگر درجهٔ پیچیدگی الگوریتمی POL حداکثر در ردهٔ NP است؛ اگر خودش P باشد، هر X دیگری در NP هم همین‌طور است. به‌جای اینکه به تفصیل توضیح دهم که یک تبدیل چندجمله‌ای چیست، مثالی خواهم آورد از تبدیل یک مسألهٔ POL به مسأله‌ای که ظاهراً ساده‌تر است، اما در واقع به همان اندازه پیچیده است.

مسألهٔ مورد نظر، مسألهٔ POL۳ است، که حالت خاصی است از POL که در آن هر معادله تحت این قید اضافی است که نباید شامل بیش از ۳ متغیر از n متغیر باشد.

قضیهٔ کوک. اگر POL۳ را بتوان در زمانی چندجمله‌ای برحسب n حل کرد، POL را هم می‌توان در همین زمان حل کرد (و P=NP).

اثبات. فرض می‌کنیم S یک دستگاه معادلات چندجمله‌ای با اندازهٔ n باشد. تک‌جمله‌ای $m = x_1 \odot x_2 \odot \ldots \odot x_p$ را که در این دستگاه هست درنظر می‌گیریم، و به‌آن $p-1$ مجهول جدید، یعنی $a_2, \ldots, a_p$

نسبت می‌دهیم. دستگاه $a_1 \otimes x_1 = u_1$، $a_2 \otimes x_2 = u_2$، $a_3 \otimes x_3 = u_3$، ... $a_p \otimes x_p = v$ معادل است با شرط $m = v$. سپس یکی از معادلات S به‌شکل $P = m_1 \oplus \ldots \oplus m_q = 0$ را درنظر می‌گیریم که به تک‌جمله‌ایهای آن متغیرهای $w_1, \ldots, w_q$ نسبت داده شده‌اند. اکنون $q-2$ مجهول جدید، یعنی $w_3, \ldots, w_q$ را معرفی می‌کنیم و مشاهده می‌کنیم که دستگاه $w_3 = v_1 \oplus v_2$، $w_4 = w_3 \oplus v_3$، ... $w_q = v_q \oplus v_{q-1} = 0$ هم ارز است با معادلهٔ $P = 0$.

بنابراین می‌بینیم که دستگاه T متشکل از تمام این معالات جدید دارای جواب است اگر و تنها اگر دستگاه S اولیه دارای جواب باشد. تمام این معادلات به‌صورت $x = 0$، $x = 1$ (حالت تک جمله‌ای ثابت!) هستند، و در نتیجه این دستگاه واقعاً داده‌ای است برای مسألهٔ SAT۳. طول آن چقدر است؟ حداکثر n^2 تک‌جمله‌ای وجود دارد که هر یک بیش از n مجهول جدید و n معادله وارد کار نمی‌کنند. بنابراین اندازهٔ T از مرتبهٔ n^3 است. بنابر فرض، وجود یک جواب برای T را می‌توان در یک زمان چندجمله‌ای برحسب n^3 تعیین کرد، و این یک چندجمله‌ای در n نیز هست. **فهوالمطلوب.**

بنابراین POL۳ ساده‌تر از POL نیست؛ و این امر چندان هم اسرارآمیز نیست؛ همین تبدیل را از LIN به حالت خاص آن LIN۳ داریم، که در آن هر معادله‌ای بیش از سه متغیر ندارد (قرار می‌دهیم $a_1 = a_1 \otimes x_1$، $a_2 = a_1 \oplus a_2 \otimes x_2$، و غیره)، اما این کار به ما در حل LIN کمک زیادی نمی‌کند! برعکس، مسألهٔ POL۲، که در آن هر معادلهٔ چندجمله‌ای فقط شامل یک یا دو متغیر می‌شود، به‌سادگی قابل حل است.

قضیه. مسألهٔ POL۲ـ یعنی این مسأله که آیا دستگاهی مرکب از n معادلهٔ چندجمله‌ای با n مجهول که در آن هر معادله فقط شامل یک یا دو متغیر می‌شود دارای جواب است یا نه ـ از نظر الگوریتمی در یک زمان چندجمله‌ای قابل حل است.

اثبات. مشاهده می‌کنیم که اگر الگوریتم حذف را در مورد معادلات دو متغیره به‌کار ببریم، تنها معادلاتی با (حداکثر!) دو متغیر تولید می‌شوند. در این‌صورت ضرب چندجمله‌ایها سریعاً انجام می‌شود، زیرا یک چندجمله‌ای با دو متغیر x و y متشکل از بیش از چهار تک‌جمله‌ی $(1, y, x, x \otimes y)$ نیست؛ به‌علاوه، تعداد معادلات نمی‌تواند فوق‌العاده زیاد شود، زیرا بیش از $8n \times (n-1)$ چندجمله‌ای شامل دو متغیر که از میان $x_1 \ldots x_n$ انتخاب شده‌اند وجود ندارد. در هر مرحله معادلات را غربال می‌کنیم و تکراریها را حذف می‌کنیم، و در یک زمان چندجمله‌ای به نتیجه می‌رسیم. **فهوالمطلوب.**

۶. توضیح دربارهٔ مراجع مقاله

مراجع مربوط به سه نتیجه‌ای که در این مقاله ذکر کردیم عبارت‌اند از [۷] [۱۰] و [۳] (که در آن وجود خود مسائل NPـتمام نشان داده شده است). اگر فکر می‌کنید نگارندهٔ این سطور در مورد موضوع الگوریتم خالص و ناب قصور ورزیده به منابع کلاسیک در این مورد رجوع کنید: [۱] و [۸]. اثر مرجعی که باید برای گم نشدن در جنگل مسائل NPـبدان مراجعه کرد، مرجع [۵] است. در مورد مدارهای بولی دو مرجع اصلی وجود دارد: [۴] و [۱۱]. قبل از خواندن این آثار مهم، خوانندهٔ‌ای که از اسلوب این مقاله خوشش آمده می‌تواند یک کتاب راهنمای جدید [۹] را که در آن مطالب اصلی در مورد کاربرد مدارها در الگوریتم یادآوری شده مطالعه کند. این اثر در چارچوبی قرار دارد که در [۲] معرفی شده و در [۶] بسط و گسترش یافته و در آن محاسبات تنها به کمک دو نماد ۰ و ۱ انجام نشده، بلکه به‌وسیلهٔ عناصر یک ساختار دلخواه انجام شده است. این روش ممکن است کمتر از روش محاسبهٔ دوتایی واقعگرایانه باشد (که این‌هم جای بحث دارد)، اما امتیاز آن این است که مسائلی را مطرح می‌کند که از لحاظ ریاضی جالب توجه هستند، و شاید هم ـ البته مسلم نیست ـ آسانتر از دو مسألهٔ «P = NP?» و «P = NC?» باشند.

مراجع

1. A.V. Aho, J. E. Hopcroft & J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley (1968).
2. Lenore Blum, Mike Shub & Steve Smale, "On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions, and universal machines", *Bulletin of the American Mathematical Society*, **21** (1989) 1-46.
3. S.A. Cook, "The complexity of theorem proving procedures", *Proc. Third Annual ACM Symposium on the Theory of Computing* (1971) 151-158.
4. Paul E. Dunne, *The Complexity of Boolean Networks*, Academic Press (1988).
5. M.R. Garey & D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, H. Freeman, San Francisco (1979).
6. John B. Goode, "Accessible telephone directories", *The Journal of Symbolic Logic*, **59** (1994) 92-105.
7. H.J. Hoover, M.M. Klawe & N.J. Pippenger, " Bouding fan-out in logical networks", *Journal of the Association for Computing Machinery*, **31** (1984) 13-18.
8. Donald E. Knuth, *The Art of Computer Programming*, vol. 3, Addison-Wesley (1973).
9. Bruno Poizat, *Les Petits Cailloux*, Nur al-Mantiq wal Ma'rifah n° 3, Editions Aléas, Lyon, France (1995).
10. Spira, "On the time necessary to compute switching functions", *IEEE Transactions on Comp.*, **20** (1971) 104-105.
11. Ingo Wegener, *The Complexity of Boolean Functions*, Wiley-Teubner (1988).

******

• این مقاله را نویسنده برای چاپ در نشر ریاضی نوشته است. متن فرنسهٔ مقاله، پس از انتشار ترجمهٔ فارسی آن در نشر ریاضی، در مجلهٔ فرنسوی *Gazette des Mathématiciens* چاپ خواهد شد.

* برونو پوازا، مؤسسه زیرار دزارگ، دانشگاه لیون فرنسه، دانشگاه قزاقستان در قراگاند
poizat@jonas.univ-lyon1.fr