خلاصه فصل دوم کتاب Boaz

لیست کلی از چیزهایی که توی این فصل بهشون پرداخته میشه:

- تفاوت‌های بین specification و implementation یا هم ارزی بین mathematical functions و algorithms/programs
- نمایش دادن یک شی به صورت رشته (صفر و یک)
- مثال‌هایی برای نمایش دادن اشیاء معروف مثل اعداد و بردار ها ، لیست و گراف
- نمایش prefix-free
- تئوری کانتور : اعداد حقیقی نمیتونن کاملاً توسط رشته‌های متناهی نمایش داده بشن

ساده‌ترین تعریفی که ما از محاسبات داریم: یک precess هست که ورودی‌ها رو به خروجی map میکنه. اینجا خیلی مهمه که ما تفاوت اینو بدونیم که اصلاً مشخصات این تسکی که میخوایم انجامش بدیم چیه (what / specification) و حالا چطوری میخوایم به این تسک دست پیدا کنیم (how / implementation)

در این بخش تمرکز ما روی چیستی تسک ها هست یا همون what / specificaton پارت که اینجا بهش میگیم محاسبات (computation) که بر اساس همون تعریف ساده و اولیمون نیاز داریم که تمام ورودی و خروجی های ممکن رو در نظر بگیریم و بتونیم روی همشون عملیات انجام بدیم اما مسأله اینجاست که این ورودی و خروجی ها همیشه اعداد نیستن و ما امروزه روی هرچیزی محاسبه انجام میدیم از عکس گرفته تا گراف و بردار و ... پس هدف فعلی ما میشه نمایش دادن هر چیزی به صورت یک رشته از صفر و یک.

بخش جالب ماجرا اینجاست که ما میخوایم یه طیف نامحدود از اشیائی که وجود دارنو توسط یک رشته از کاراکترهای محدود مثل ۰ و ۱ نمایش بدیم و بیایم در ادامه روی این رشته‌ها محاسباتی رو تعریف کنیم که به کارمون میان.

نکات اصلی این فصل اینان:

- **نمایش با صفر و یک**: ما می‌تونیم همه چیز رو با رشته‌هایی از صفر و یک (باینری) نمایش بدیم، مثل اعداد صحیح و گویا.
- **ترکیب نمایش‌ها**: می‌تونیم با ترکیب نمایش‌های ساده، چیزای پیچیده‌تری مثل ماتریس، تصویر و گراف رو نشون بدیم. کدگذاری بدون پیشوند یکی از راه‌های این کاره.
- **فرق تابع و برنامه**: یه وظیفه محاسباتی فقط نشون می‌ده چه خروجی به چه ورودی ربط داره، و روش‌های مختلفی برای محاسبه یه تابع وجود داره. پس، خیلی مهمه که فرق بین «چی هست» (تابع) و «چطوری انجام می‌شه» (برنامه) رو بدونیم.
- **محدودیت‌های نمایش باینری**: با اینکه رشته‌های باینری بی‌نهایت هستن، نمی‌تونیم همه چیز، مثل اعداد حقیقی، رو دقیقاً باهاشون نمایش بدیم.

- **دو ایده اصلی**: ترکیب نمایش‌های ساده برای ساخت چیزای پیچیده و تمایز بین تابع و برنامه، دو تا ایده اصلی این فصله که توی بقیه کتاب هم بهشون برمی‌گردیم.

۲.۱ defining representations

هر بار که چیزی مثل عدد، تصویر، یا صدا رو روی کامپیوتر ذخیره می‌کنیم، در واقع یه نمایشی از اون رو ذخیره می‌کنیم، نه خود اون چیز رو. این موضوع فقط مخصوص کامپیوتر نیست؛ مثلاً وقتی یه متن می‌نویسیم یا یه نقاشی می‌کشیم، داریم ایده‌ها یا حس‌ها رو با نمادهایی نمایش می‌دیم. حتی مغزمون هم ورودی‌های حسی رو به‌صورت یه نمایش نگه می‌داره.

حالا برای اینکه بتونیم این چیزها رو به‌عنوان ورودی برای محاسبات استفاده کنیم، باید دقیق مشخص کنیم چطوری اون‌ها رو به رشته‌های صفر و یک تبدیل کنیم. یه روش نمایش اینه که یه شیء رو به یه رشته باینری مخصوصش وصل کنیم.اما یه حداقل شروط بدیهی رو باید رعایت کنیم مثلاً برای اعداد طبیعی، باید یه طوری نمایش بدیم که هر عدد، یه رشته متفاوت داشته باشه، یعنی دو عدد متفاوت نباید با یه رشته مشابه نمایش داده بشن. یا به عبارتی تابعی که مارو از اعداد طبیعی به نمایش رشته‌ای صفر و یکی میبره باید **یک به یک** باشه.

$$E_{(x)} \in \{0,1\}^*$$

$$E: \mathbb{N} \longrightarrow \{0,1\}^*$$

$$\text{if } x \neq x' \text{ then } E_{(x)} \neq E_{(x')}$$

۲.۱.۱ representing natural numbers

برای نمایش اعداد طبیعی به صورت رشته‌ای از صفر و یک ها ما از تعریف زیر استفاده میکنیم:

$$NtS(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ NtS(\lfloor n/2 \rfloor)parity(n) & n > 1 \end{cases}$$

که در این تعریف تابع NtS به عنوان ورودی یک عدد طبیعی میگیره و در خروجی با یک رشته صفر و یک نمایشش میده. این تابع یک به یک هست. parity(n) هم یک تابعِ که زوج و فرد بودن رو چک میکنه در صورت زوج بودن صفر و در صورت فرد بودن یک برمیگردونه. منطق پشت تابع NtS همون تقسیم نرده بونی هست که هربار باقی مونده تقسیم عدد بر دو رو مینویسیم و خارج قسمت رو باز بر دو تقسیم میکنیم و میریم جلو و چون این تابع یک به یک هست پس ما میتونیم StN(s) رو هم تعریف کنیم که با گرفتن یک رشته و ضرب کردن هر مقدار در توان ۲ مقدار عدد طبیعی یک رشته رو بهمون برگردونه.

$$\text{if } s = 101110\ldots$$

$$\sum_{i=0}^{length(s)} s_i \times 2^{|s|-i-1}$$

۲.۱.۲ meaning of representations (discussion)

ما معمولاً ۲۳۶ رو "عدد واقعی" می‌دونیم و ۱۱۱۰۱۱۰۰ رو یه نمایش از اون. اما مثلاً برای اروپایی‌های قدیم، نمایش اصلی CCXXXVI بوده. فلسفه‌های مختلفی هست که می‌گن عدد واقعی چیه. افلاطون معتقد بود که اعداد تو یه دنیای ایده‌آل وجود دارن و نمایش‌هایی مثل ۲۳۶ فقط نماد اون اعداد ایده‌آل هستن. ولی ویتگنشتاین می‌گفت اعداد فقط علامت‌هایی روی کاغذن و هیچ وجود واقعی ندارن. (این بخش کلای چیز مهمی نیست و برای بحث کردنه میشه ازش گذشت)

۲.۲ representations beyond natural numbers

۲.۲.۱ representing potentially nagetive integers

در ادامه برای نمایش اعداد صحیح از یه تابع دیگه استفاده میکنیم که همون نمایش بیت علامتی هست که همه باهاش آشنایید:

$$ZtS(m) = \begin{cases} 0\ NtS(m) & m \geq 0 \\ 1\ NtS(-m) & m < 0 \end{cases}$$

این تابع هم یک به یک هست. دقت کنید که لازم نیست توابع ما پوشا باشن (یعنی لازم نیست ما کل $\{0,1\}^*$ رو پوشش بدیم) فقط کافیه کل ورودی رو پوشش بدن و یک به یک باشن.

اما مشکلی که به دوتا نمایش NtS , ZtS وارده این هست که ما دیگه نمیدونیم یه رشته رو چطوری به عدد باید تبدیلش کرد. درواقع اگه رشته ای داشته باشیم که توسط این دوتا تابع تولید شده باشه نمیتونیم متوجه بشیم اون عدد چی بوده تا وقتی بهمون نگن توسط کدوم تابع تبدیل شده به رشته.

۲.۲.۲ two’s complement representation (optional)

اگر ما عدد طبیعی یا صحیح k رو داشته باشیم به طوری که

$$k \text{ in the set } \{-2^n, -2^n+1, \ldots, 2^n-1\}$$

اون موقع تابع زیر بیانگر نمایش مکمل ۲ هست:

$$ZtS_n(k) = \begin{cases} NtS_{n+1}(k) & 0 \leq k \leq 2^n - 1 \\ NtS_{n+1}(2^{n+1}+k) & -2^n \leq k \leq -1 \end{cases}$$

که در این نمایش هم $NtS_l(m)$ به معنای نمایش NtS با استفاده از l بیت هست.

۲.۲.۳ rational numbers and representing pairs of strings

برای نمایش اعداد گویا که به صورت کسری نوشته میشن اگه ما کاغذ و قلم داشتیم خیلی راحت یه خط برای فاصله گذاشتن استفاده میکردیم و الفبامون به جای بیت‌های صفر و یک میشد صفر و یک و |. اما چون همچین کاری نمیتونیم انجام بدیم و در صورتی هم که پشت سر هم بنویسیمشون مشکلای زیادی پیش میاد مثلاً جفت عدد ۴ و ۴۳ که به صورت ۱۰۰ و ۱۰۱۰۱۱ نمایش داده میشن در کنار همدیگه ۱۰۰۱۰۱۰۱۱ رو تشکیل میدن

که ما میتونیم جفت عدد رو ۱۱ و ۱۸ هم در نظر بگیریم. برای اینکه همچین مشکلاتی پیش نیان میایم از یه تریک کوچیک استفاده میکنیم و بیت‌ها و علامت فاصلمونو به مجموعه‌ی {۰۰, ۰۱ , ۱۱ } مپ میکنیم که بتونیم وقتی دوتا رشته کنار هم میان از هم تشخیصشون بدیم.

برای مثال:

$$r = -\frac{5}{8}$$

$$-5 = 1101 \ , \ 8 = 0100$$

$$1101 \mid 01000$$

$$0 \to 00 \ , \ 1 \to 11 \ , \ \mid \to 01$$

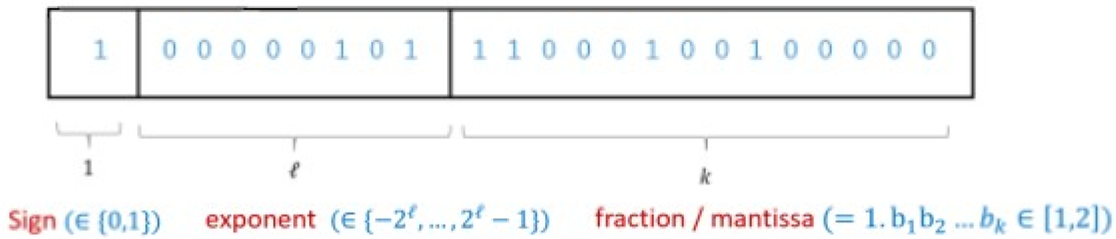
$$\Rightarrow -\frac{5}{8} = 1111001101001100\,00$$

✔ **big idea** : اگه ما بتونیم یک شی از نوع T رو به صورت رشته نمایش بدیم اون موقع میتونیم یک لیست از این اشیاء رو درکنار همدیگه نگهداری کنیم.

این ایده در ادامه به ما کمک خواهد کرد که به نمایش prefix-free برسیم.

۲.۳ representing real numbers

یک راه برای نمایش تقریبی اعداد حقیقی استفاده از ممیز شناور هست



$$-1 \times 2^5 \times \left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{64} + \frac{1}{512}\right) = -56.5625$$

اما مشکل این مدل نمایش تقریب اون هست. یعنی ما اومدیم و عدد رو با یه مقدار خطا تقریب زدیم و نوشتیم.

برای مثال عدد پی با تقریباً $۱۰^{-۳}$ برابر با ۳۱۱/۹۹ یا همون ۳.۱۴ هست و اگه بخوایم دقت رو ببریم بالا تر و تقریب بهتری داشته باشیم باید با ۲۲/۷ نمایشش بدیم.

۲.۴ cantor’s theorem, countable sets, and string representation of real numbers

با توجه به دقیقق نبودن ممیز شناور سؤالی که پیش میاد این هست که ممکنه بتونیم اعداد حقیقی رو کاملاً دقیق نمایش بدیم؟

با توجه به قضیه های زیر جواب خیر هست.

قضیه ۲.۵. تئوری کانتور: هیچ تابع یک به یکی وجود ندارد که اعداد حقیقی را به استرینگ ها مپ کند. یعنی همچین تابعی نخواهیم داشت:

$$RtS : \mathbb{R} \to \{0,1\}^*$$

countable sets: وقتی میگوییم یک مجموعه شمارا است که یک تابع یک به یک از اعداد طبیعی به اون مجموعه داشته باشیم. همون طور که در قبل داشتیم و میدونیم میتونید یک تابع یک به یک از نمایش باینری به اعداد طبیعی داشته باشیم و چون یک تابع یک به یک وجود داره پس تابع وارونش پوشا خواهد بود و میتوان نتیجه گرفت که مجموعه‌ی $\{۰,۱\}^*$ شمارا است. پس میتونیم بگیم یک مجموعه مثل S شمارا هست اگر و تنها اگر یک تابع پوشا از مجموعه‌ی $\{۰,۱\}^*$ به S داشته باشیم.

قضیه ۲.۶. تئوری کانتور (بیان هم ارز): اعداد حقیقی ناشمارا هستند زیرا هیچ تابع پوشایی به شکل زیر نداریم:

$$NtR : \mathbb{N} \to \mathbb{R}$$

برای اثبات تئوری کانتور ما چند تا مرحله رو باید طی کنیم اما قبل از اونا باید یک مجموعه رو تعریف کنیم.

> **Definition 2.7** We denote by $\{0,1\}^\infty$ the set $\{f \mid f : \mathbb{N} \to \{0,1\}\}$.

درواقع یک تابع مثل f در مجموعه $\{۰,۱\}^\infty$ هست اگر و تنها اگر دامنش اعداد طبیعی باشن و بردش (هم دامنش) مجموعه‌ی $\{۰و۱\}$ باشن.

لم ۲.۸: هیچ تابع یک به یکی مثل FtS از $\{۰,۱\}^\infty$ به $\{۰,۱\}^*$ نداریم.

لم ۲.۹: یک تابع یک به یک مثل FtR از $\{۰,۱\}^\infty$ به اعداد حقیقی داریم.

لم‌های بالا با همدیگه قضیه ۲.۵ رو ثابت میکنن. برای اثبات این قضیه میتونید از تناقض استفاده کنیم. اگرفرض کنیم یک تابع یک به یک RtS وجود داره و چون ترکیب دوتا تابع یک به یک ، یک به یک هست پس تابع FtS به شکل RtS(FtR) وجود خواهد داشت که یک به یک خواهد بود که با لم ۲.۸ در تضاد هست.

ساده‌تر بگم اگه فرض کنیم یک تابع یک به یک از اعداد حقیقی به رشته‌های باینری وجود داره با استفاده از دوتا لم ۲.۸ و ۲.۹ به تناقض میریسم. پس کافیه این دوتا لم رو ثابت کنیم تا قضیمون ثابت بشه.