بنام آنکه جان را فکرت آموخت

وزارت علوم، تحقیقات و فناوری
مرکز تحصیلات تکمیلی در علوم پایه
گاوازنگ - زنجان

گزارش پروژه‌ی درسی

نظریه‌ی محاسبات پیشرفته‌ی یک

گردآوری: آرمان دیدنده

شماره دانشجویی: ۸۷۴۱۰۵

نام استاد: دکتر سعید صالحی پورمهر

فصل: پائیز ۱۳۸۷

دانشگاه تحصیلات تکمیلی علوم پایه

گاوازنگ، زنجان، ایران

مقدمه

یکی از مباحث عمده و قابل اهمیت در بررسی‌های شناختی مسائل و راه‌کارها، شناخت هزینه‌هاست. این امر چه در زمینه‌های اقتصادی و چه در پروژه‌های جامعه‌شناختی به پای ثابت بحث‌های پیش‌کار و حتی در حین کار مبدل گشته است و خود را به یک نیاز شناختی به بشر شناسانده است. در مورد مسائل علمی نیز این نیاز همواره حضور دارد و حتی گاهی به اهم اشکال خود را بروز می‌دهد. هزینه‌ها عموماً گره-ای ریشه‌ای با منابع دارند که جداسازی این‌دو را بسیار دشوار و شاید غیرممکن می‌سازد.

در مورد علوم مرتبط با ریاضیات و علوم پایه‌ای رایانه‌ای، منابع معمولاً پیش از طرح گسترده‌ی مسأله شناخته شده‌اند. در نتیجه نحوه‌ی اختصاص این منابع است که باعث بالا و پایین شدن هزینه‌ها و سودآوری –البته در تعریف مشخص خود در این علوم- می‌گردد.

از زمان ظهور رایانه‌ها تا به امروز، دو هزینه‌ی عمده و شناخته شده، چالشی شگرف را برای متخصصان علوم رایانه پدیدار آورده است. این‌دو هزینه، همانا «هزینه/پیچیدگی زمانی» و «هزینه/پیچیدگی مکانی» می‌باشند. هر دوی این هزینه‌ها که به نوعی سایر کلاس‌های هزینه را در بر می‌گیرند، در شاخه‌ای از علوم پایه‌ای به‌نام «تئوری اطلاعاتی الگوریتمیک» مورد بحث و بررسی دقیق قرار می‌گیرند و راه‌کارهایی برای شناخت و کاهش آن‌ها ارائه می‌گردد؛ البته در اکثر موارد به‌دلیل ماهیت این دو هزینه و مسائل مطرح در این زمینه‌ها، این‌دو رودرروی یک-دیگر قرار می‌گیرند و متخصصین را مجبور به انتخاب یک trade off مابین این‌دو می‌نمایند.

در این متن تلاش بر معرفی یک تعریف جدید در شاخه‌ی تئوری اطلاعاتی الگوریتمیک برای پیچیدگی مکانی خواهد بود.

تعاریف و اطلاعات

در این بخش به بررسی برخی تعاریف مهم در تئوری اطلاعاتی الگوریتمیک می‌پردازیم:

پیچیدگی مکانی یک برنامه: میزانی برای سنجش هزینه‌ی مکانی یک برنامه از زمان شروع به اجرا تا اتمام است که شامل میزان مکان لازم برای ورودی‌ها و محاسبات است. اتمام یک برنامه اصولاً زمان رسیدن به پاسخ در نظر گرفته می‌شود. البته در حالت کلی این امکان وجود دارد که یک برنامه halt نکند و حافظه‌ی بسیار بیشتری را هم مصرف نماید.

آنتروپی: که با نمایان H(x) نمایش داده می‌شود، میزانی برای سنجش عدم قطعیت x است در حالی‌که: $x \in X$

$$H(X) \equiv E[I(X)] \equiv -\sum_{x\in X} p(x) lg\,(p(x))$$

که در آن I(X) برای یک پیام خاص، سهم آنتروپی‌اش از اطلاعات است:

$$I(X) = -lg\big(P(X)\big) \Rightarrow H(X) \le lg\left(E\left[\frac{1}{P(X)}\right]\right) = lg\,(n)$$

آنتروپی در حالتی‌که بیشینه است که تمام پیام‌ها دارای احتمال برابر باشند که در این حالت:

$$H(X) \equiv lg\,(|X|)$$

تابع آنتروپی دوتایی: به شکل زیر تعریف می‌شود:

$$H(p) = -plg(p) - (1-p)\,lg(1-p)$$

تابع آنتروپی توأم: به شکل زیر تعریف می‌شود:

$$H(X,Y) = -\sum p(x,y) lg\,(p(x,y))$$

تابع آنتروپی شرطی: به شکل زیر تعریف می‌شود:

$$H(X/Y) = H(X,Y) - H(Y)$$

اطلاعات متقابل: به شکل زیر تعریف می‌شود:

$$I(X:Y) = H(X) + H(Y) - H(X,Y)$$

نابرابری کرافت: شرطی برای وجود یک کد قابل بازیابی یکتا به‌ازای یک مجموعه از طولِ کدهای مشخص که برای حالت دوتایی به شکل زیر بیان می‌شود:

$$\sum_{l\in Leaves} 2^{-depth(l)} \le 1$$

ثابت چایتین: که در آن |P| طول جریان بیتی است:

$$\Omega = \sum_{p\in P} 2^{-|P|} \le 1$$

آنتروپی شانون: به طور هم‌ارز، میزانی برای میانگین محتوی اطلاعاتی از دست رفته در حالی است که میزان متغییر تصادفی را ندانیم. این میزان حدی را برای بهترین تراکم بدون کسری در یک ایجاد ارتباط ارایه می‌نماید.

معرفی موضوع، هدف کار و تعاریف جدید

در این مطلب به صورت کلی، تعریف جدیدی از پیچیدگی مکانی ارائه شده و مباحث مرتبط با این تعریف جدید مورد بحث قرار می‌گیرد. H(A,B/C,D) را سایز کوتاه‌ترین برنامه‌ی محاسبه‌کننده‌ی A و B در بیت در نظر می‌گیریم، به‌شرطی‌که کوتاه‌ترین برنامه برای محاسبه‌ی C و D در دست باشد. البته دو شرط برای این تعریف هست که آن‌را از تعاریف قبلی جدا می‌سازد:

۱. خودمعین بودن: هیچ برنامه‌ای پیش‌وندی از برنامه‌ی دیگر نخواهد بود.
۲. به جای دانستن خود C و D به‌عنوان ورودی، برنامه‌های محاسبه‌کننده‌ی این‌دو در اختیار ما هستند.

این دو شرط، روال مقایسه را به روال آنتروپی مشابهت می‌دهد. دلیل این علاقه‌مندی، مشابهت متقاعدکننده‌ی بین پیچیدگی مکانی و مفهوم آنتروپی در نظریه‌ی اطلاعات است.

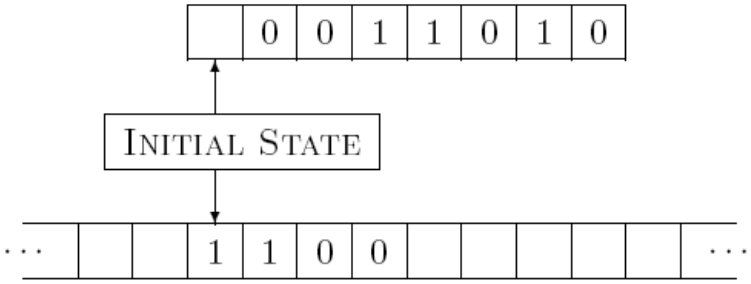
برای فرمول‌بندی این تعریف جدید، به رایانه به دید یک ابزار رمزگشایی در انتهای یک مجرای ارتباطی دوتایی می‌نگریم؛ برنامه‌ها را به مثابه کلمات رمز و نتیجه‌ی محاسبات را به کلمه‌ی رمزگشایی شده. در نتیجه برنامه‌ها باید یک مجموعه‌ی پیش‌وند-آزاد –یا یک رمز آنی- را تشکیل دهند تا بتوان پیام‌های ارسالی را جدا نمود. این مجموعه/رمزها تحت نابرابری کرافت مدیریت می‌شوند.

با معرفی مجموعه‌ی X به‌شکل زیر تعاریف لازم را استخراج نموده و مسیر حرکت برای شناخت پیچیدگی مکانی را طی خواهیم کرد:

$$X = \{\Lambda, 0, 1, 00, 01, 10, 11, 000, \dots\}$$

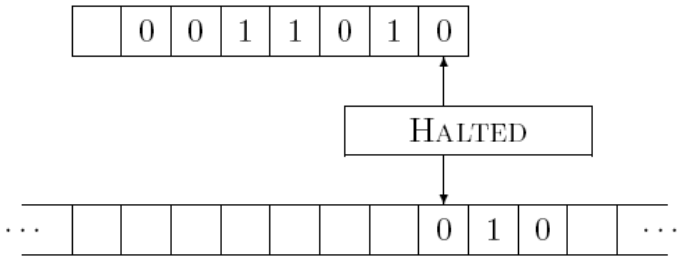
که در آن X مجموعه‌ی رشته‌های متناهی دوتایی است و X^{∞} نشان‌دهنده‌ی مجموعه‌ی رشته‌های نامتناهی دوتایی. شایان ذکر است که ترتیب لغت‌نامه‌ای برای مجموعه‌ی فوق‌الذکر در ادامه‌ی کار درنظر گرفته می‌شود. حال تعاریف لازم برای کار را ارایه داده و در موردشان بحث می‌نماییم:

تعریف محسوس/ذاتی رایانه: رایانه‌ی C یک ماشین تورینگ دو نواره است که یکی از نوارهایش را برای ذخیره‌ی برنامه در نظر می‌گیرد و آن دیگری را به‌عنوان نوار کار استفاده می‌کند. نوار برنامه تنها-خواندنی و دارای طول متناهی است و سرک خواندن بر روی آن تنها به سمت راست حرکت می‌کند. خانه‌ی سمت راست نوار برنامه حاوی حرف پوچ است و سایر خانه‌های از حروف 0 و 1 پر شده‌اند. نوار کار خواندنی-نوشتنی و دارای طول نامتناهی است و سرک خواندن و نوشتن آن به هر دو سمت حرکت می‌کند. خانه‌های این نوار شامل مقادیر $\mathscr{b}$ و 0 و 1 است. در شکل زیر حالت شروع محاسبه برای این رایانه نشان داده شده است. دقت شود که نوار کار می‌تواند در ابتدا خالی بوده و یا شامل مقدار q باشد.



شکل یک: آغاز محاسبات با p برابر 0011010 و q برابر 1100

در شکل زیر حالت ماشین را هنگامی‌که توقف می‌نماید مشاهده می‌نماییم :



شکل دو: انتهای یک محاسبه‌ی موفق که در آن C(p, q) = 010

در این حالت از توقف داریم:

$$C(p,q) = 010$$

دقت شود که توقف باید در حالت خواندن سمت راست ترین خانه‌ی نوار برنامه اتفاق بیافتد.

تعریف رمز آنی/مجموعه‌ی پیش‌وند آزاد: یک مجموعه نظیر S که در آن داریم:

$$S = \{s_i\}_{i\in\mathbb{N}} \nexists\, i,j \ni (s_i = Prefix(s_j))$$

تعریف انتزاعی رایانه: C یک تابع بازگشتی جزیی به شکل زیر است:

$$C : X \times X \to X$$

که در آن:

$$\forall\, q: Domain[C(.,q)] := a\ prefix - free\ set$$

دقت شود که اگر C(p, q) تعریف شده باشد و p یک پیش‌وند از p' باشد، آن‌گاه C(p', q) تعریف‌پذیر نخواهد بود.

قضیه‌ی ۱: تعاریف ارایه شده در بالا از رایانه با یک‌دیگر معادلند.

تعریف رایانه‌ی جهانی بهینه: U را یک رایانه‌ی جهانی بهینه نامیم اگر و فقط اگر:

برای هر رایانه‌ی C، ثابت sim(C) موجود باشد که:

$$if\ C(p,q)\ is\ defined, then: \{\exists\, p': U(p',q) = C(p,q)\}\ and\ \{|p'| \le |p| + sim(C)\}$$

قضیه‌ی ۲: یک رایانه‌ی جهانی بهینه U وجود دارد.

! در ادامه از این رایانه‌ی جهانی بهینه برای سنجش پیچیدگی مکانی (اندازه‌ی برنامه) استفاده می‌شود.

تعریف برنامه‌های متعارف:

$$s^* = Min\ (p)\{U\ (p,\Lambda) = s\}$$

که در آن s* اولین عنصر در مجموعه‌ی مرتب X است که یک برنامه برای محاسبه‌ی s می‌باشد. به‌عبارت دیگر s^* کوتاه‌ترین رشته‌ایست که برنامه‌ای در U برای محاسبه‌ی s است و در صورتی‌که تعدد در برنامه‌های هم‌اندازه پیش آید، ترتیب لغت‌نامه‌ای مورد بحث در بالا برای انتخاب رعایت می‌گردد.

تعاریف پیچیدگی:

$$H_C(s) = Min(|p|)\{C\ (p,\Lambda) = s\}$$

$$H(s) = H_U(s)$$

$$H_C(s/t) = Min(|p|)\{C\ (p,t^*) = s\}$$

$$H(s/t) = H_U(s/t)$$

$$H_C(s{:}t) = H_C(t) - H_C(t/s)$$

$$H(s{:}t) = H_U(s{:}t)$$

تعاریف احتمالات:

$$P_C(s) = \sum 2^{-|p|}\ \{C\ (p,\Lambda) = s\}$$

$$P(s) = P_U(s)$$

$$P_C(s/t) = \sum 2^{-|p|}\ \{C\ (p,t^*) = s\}$$

$$P(s/t) = P_U(s/t)$$

$$\Omega = \sum 2^{-|p|} \{U(p, \Lambda) \text{ is defined}\}$$

! ***Ω به ثابت چایتین معروف است و براساس نابرابری کرافت (که شرایط برای وجود یک رمز قابل بازگشایی یکتا برای یک مجموعه طول کلمات رمز ارایه می‌دهد) که طول کلمات رمز را در یک رمز پیش‌وند-آزاد محدود می‌سازد، برای درختان دوتایی رابطه‌ای را ارایه می‌کند که در این‌جا به‌شکل زیر در خواهد آمد:***

$$\Omega = \sum 2^{-|p|} \{p \in P\} \leq 1$$

! ***یک برنامه در زبان لیسپ وجود دارد که Ω را در شکل حدی محاسبه می‌کند. زمان اجرای این برنامه به‌صورت تقریبی ۱۲۷.۵۸۵۳۹۹ ثانیه است.***

! ***نمایان H(s:t) معرف اطلاعات الگوریتمی متقابل برای s و t است که میزان مناسبی برای سنجش درجه‌ی استقلال این‌دو از یک‌دیگر است. به بیان دیگر، این میزان بیان‌گر این است که دانستن s تا چه حد به محاسبه‌ی t کمک می‌کند.***

! ***Ω احتمال توقف رایانه‌ی U با داده‌ی آزاد Λ بر روی تمامی مقادیر ممکن p است. مشخص است که این مقدار باید کوچک‌تر از یک باشد.***

! ***با تغییری جزیی در تعریف رایانه، اگر نوار برنامه نامتناهی بوده و محاسبات موفق نیازی به ایستادن سرک بر روی سمت راست‌ترین خانه‌ی حاوی p نداشته باشد، و این‌که تمام خانه‌های نوار به‌جز خانه‌ی زاید اول شامل صفر و یک‌ها به این گونه باشند که انتخاب براساس پرتاب یک سکه‌ی متوازن باشد؛ داریم:***

$P_C(s)$ برابر احتمال به دست آوردن s در محاسبات است به‌شرطی‌که نوار کار در ابتدا خالی باشد.

$P_C(s/t)$ برابر احتمال به دست آوردن s در محاسبات است به‌شرطی‌که نوار کار در ابتدا حاوی t^* باشد.

قضیه‌ی ۳: این قضیه شامل گزاره‌های زیر است:

$a)\ H(s) \leq H_C(s) + sim(C)$

$b)\ H(s/t) \leq H_C(s/t) + sim(C)$

$c)\ s^* \neq \Lambda$

$d)\ s = U(s^*, \Lambda)$

$e)\ H(s) = |s^*|$

$f)\ H(s) \neq \infty$

$g)\ H(s/t) \neq \infty$

$h)\ 0 \leq P_C(s) \leq 1$

$i)\ 0 \leq P_C(s/t) \leq 1$

$j)\ \sum_s P_C(s) \leq 1$

$k)\ \sum_s P_C(s/t) \leq 1$

$l)\ P_C(s) \ \geq\ 2^{-H_C(s)}$

$m)\ P_C(s/t) \ \geq\ 2^{-H_C(s/t)}$

$n)\ 0\ < P(s) < 1$

$o)\ 0\ < P(s/t) < 1$

$p)\ \#\{s: H_C(s)\} \ <\ 2^n$

$q)\ \ \#\{s: H_C(s/t)\} <\ 2^n$

$r)\ \#\left\{s: P_C(s)\ > \frac{n}{m}\right\} < \frac{m}{n}$

$s)\ \#\left\{s: P_C(s/t)\ > \frac{n}{m}\right\} < \frac{m}{n}$

تعریف چندتایی‌های رشته‌ای: با فرض یک تابع یک‌به‌یک و پوشا نظیر $b: X \times X\ \rightarrow X$ داریم:

$< s_i > :=\ s_i$
and
$for\ n \geq 2:\quad < s_1, s_2, \dots, s_n > := b(< s_1, s_2, \dots, s_{n-1} >, s_n)$

گسترش مفاهیم قبلی در مورد چندتایی‌های رشته‌ای: از آنجایی‌که می‌توان هر s_k را یک رشته‌ی متشکل از صفر و یک‌ها در زبان لیسپ در نظر گرفت و لیست مرتب $<s_1, \dots, s_n>$ در این زبان قابل تعریف است، برای مثال داریم:

$H(s_1, s_2, s_3/s_4, s_5, s_6) \equiv H(< s_1, s_2, s_3 >/< s_4, s_5, s_6 >)$
$H(s, t) \equiv H(< s, t >)$

گسترش مفاهیم قبلی در مورد اعداد طبیعی(اعداد صحیح نامنفی): در چندتایی‌ها، می‌توان به جای رشته‌ها در عناصر چندتایی‌ها، از اعداد طبیعی استفاده نمود. برای این‌کار می‌توانیم n را به صورت‌های زیر کد نماییم:

- ✓ n := n^{th} string in X = {Λ, 0, 1, 00, 01, 10, 11, 000, ...}
- ✓ n:= the list consisting n 1's, then X = { Λ, 1, 11, 111, 1111, …}

⇒
$H(n) \equiv H(the\ n^{th}\ element\ of\ X)$
$\{U(p, \Lambda) =\ n\} \quad \equiv\ \ \{U(p, \Lambda) =\ the\ n^{th}\ element\ of\ X\}$

بررسی اصلی

دو هدف اصلی ما در این بخش بررسی موارد زیر است:

۱. این‌که تابع مورد بحث پیچیدگی اندازه‌ی برنامه از نابرابری‌های اساسی و شناسه‌های تئوری اطلاعات تبعیت می‌نماید؛ مثلاً اطلاعات درون s در مورد t متقارن است.

۲. این‌که تابع P یک میزان احتمالاتی شرطی به صورت تقریبی است؛ مثلاً $P(t/s)$ و $P(t,s)/P(s)$ حداکثر یک ضریب ثابت از یک‌دیگرند.

قضیه‌ی ۴: این قضیه شامل گزاره‌های زیر است:

$a)\ H(s,t) =\ H(t,s) +\ O(1)$

$b)\ H(s,s) = O(1)$

$c)\ H(H(s)/s)\ = O(1)$

$d)\ H(s) \leq H(s,t) +\ O(1)$

$e)\ H(s/t)\ \leq H(s) +\ O(1)$

$f)\ H(s,t)\ \leq H(s) +\ H(t/s) +\ O(1)$

$g)\ H(s,t)\ \leq H(s) +\ H(t) +\ O(1)$

$h)\ H(s:t) \geq O(1)$

$i)\ H(s:t) \leq H(s) + H(t) - H(s,t) + O(1)$

$j)\ H(s:s) = H(s) + O(1)$

$k)\ H(\Lambda:s) = O(1)$

$l)\ H(s:\Lambda) = O(1)$

قضیه‌ی ۵: گسترش شرط نابرابری کرافت برای وجود یک رمز آنی/مجموعه‌ی پیش‌وند-آزاد:

فرضیه: $\{<s_k, n_k> : k=0, 1, \ldots\}$ لیستی متناهی یا نامتناهی از تقاضاها برای ساخت یک رایانه است. تقاضاها را استوار می‌نامیم اگر :

$$\sum_k 2^{-n_k} \leq 1$$

با این فرض که این تقاضاها استوار هستند به کار ادامه می‌دهیم. زوج $<s_k, n_k>$ بیان‌گر این امر است که برنامه‌ای از طول n_k دارای نتیجه‌ی s_k باشد.

رایانه‌ی C این تقاضاها را پاسخ‌گو خواهد بود، اگر دقیقاً به تعداد برنامه‌های p از طول n که شرط $C(p, \Lambda) = s$ را ارضا می‌کنند، زوج‌های $<s, n>$ داشته باشیم. این رایانه‌ی C دارای دو خاصیت زیر است:

$$P_C(s) = \sum_{s_k=s} 2^{-n_k}$$

$$H_c(s) = Min(n_k)\ \{s_k = s\}$$

نتیجه: چنین رایانه‌هایی وجود دارند که این تقاضاها را پاسخ می‌گویند. علاوه بر این، اگر تقاضاها را یک به یک به ما بدهند، این امکان هست که رایانه‌ای را برای‌شان شبیه‌سازی نماییم. از این‌جا به بعد به این رایانه، رایانه‌ی مشخص شده با تقاضاها نام می‌دهیم.

قضیه‌ی ۶: محاسبه‌ی H_C و P_C در حالت حدی/تخمین‌های بازگشتی برای این دو مقدار:

با فرض وجود رایانه‌ی C داریم:

۱. مجموعه‌ی تمام گزاره‌های درست به شکل زیر به طور بازگشتی محاسبه‌پذیر است:

$$H_c(s) \leq n$$

با در دست داشتن t^*، می‌توان به صورت بازگشتی، مجموعه‌ی تمام گزاره‌های درست به شکل زیر را محاسبه نمود:

$$H_c(s/t) \leq n$$

۲. مجموعه‌ی تمام گزاره‌های درست به شکل زیر به طور بازگشتی محاسبه‌پذیر است:

$$P_C(s) > \frac{n}{m}$$

با در دست داشتن t^*، می‌توان به صورت بازگشتی، مجموعه‌ی تمام گزاره‌های درست به شکل زیر را محاسبه نمود:

$$P_C(s/t) > \frac{n}{m}$$

! *توجه شود که مجموعه‌ی تمام گزاره‌های درست به شکل گفته شده لزوماً به طور بازگشتی محاسبه‌پذیر نخواهد بود.*

قضیه‌ی ۷: برای هر رایانه‌ی C ثابت c وجود دارد به‌طوری‌که:

$a)\ H(s) \leq -lg(P_C(s)) + c$

$b)\ H(s/t) \leq -lg(P_C(s/t)) + c$

قضیه‌ی ۸: برای هر رایانه‌ی C ثابت c وجود دارد به‌طوری‌که:

$a)\ P(s) \geq 2^{-c} P_C(s)$

$b)\ P(s/t) \geq 2^{-c} P_C(s/t)$

$c)\ H(s) = -lg(P(s)) + O(1)$

$d)\ H(s/t) = -lg(P(s/t)) + O(1)$

قضیه‌ی ۹: تعداد برنامه‌های مینیمال محدود است:

$a)\ \#\{p: U(p,\Lambda) = s\ \&\ |p| \leq H(s) + n\} \leq 2^{n+O(1)}$

$b)\ \#\{p: U(p,t^*) = s\ \&\ |p| \leq H(s/t) + n\} \leq 2^{n+O(1)}$

قضیه‌ی ۱۰: بیان می‌کند که:

$$P(s) \cong \sum_t P(s,t)$$

قضیه‌ی ۱۱: رایانه‌ی C و ثابت c وجود دارد، به‌طوری‌که:

$$H_C(t/s) = H(s,t) - H(s) + c$$

قضیه‌ی ۱۲: بیان می‌کند که:

$$a)\ H(s,t) = H(s) + H(t/s) + O(1)$$
$$b)\ H(s{:}t) = H(s) + H(t) - H(s,t) + O(1)$$
$$c)\ H(s{:}t) = H(t{:}s) + O(1)$$
$$d)\ P(t/s) \cong P(s,t)/P(s)$$
$$e)\ H(t/s) = lg(P(s)/P(s,t)) + O(1)$$
$$f)\ H(s{:}t) = lg\ (P(s,t)/P(s) * P(t)) + O(1)$$

! بنابراین، نتایج زیر به‌صورت کاملاً واضح امکان استخراج از قضایای بالا را دارند:

$$H(s_1) \le H(s_1/s_2) + H(s_2/s_3) + H(s_3/s_4) + H(s_4) + O(1)$$
$$H(s_1,s_2,s_3,s_4) = H(s_1/s_2,s_3,s_4) + H(s_2/s_3,s_4) + H(s_3/s_4) + H(s_4) + O(1)$$

و علاوه‌بر این نتایج، مشخصه‌های دیگری نیز وجود دارند که با تابع H جدید مطابقت دارند، ولی در نمونه‌های مشابه تابعی در نظریه‌ی اطلاعاتی الگوریتمیک آن‌ها را بیرون نمی‌دهند. از آن جمله:

$$H(H(s)/s) = O(1)$$
$$H(s,H(s)) = H(s) + O(1)$$

! با در نظر گرفتن قسمت اول از قضیه‌ی ۱۲ و تغییر رایانه‌ی C به ترتیبی که علاوه بر موارد قبلی، H(s) و H(t/s) را نیز با زیرروال‌هایش به‌کمک s^* و t محاسبه کند؛حالت توقف برای این رایانه‌ی جدید به صورت <s, t, H(s), H(t/s)> خواهد بود و نتیجه خواهد داد که:

$$H(s,t) = H(s,t,H(s),H(t/s)) + O(1)$$
$$H(H(s),H(t/s)/s,t) = O(1)$$
$$H(H(s),H(t),H(t/s),H(s/t),H(s,t)/s,t) = O(1)$$
$$H(H(s{:}t)/s,t) = O(1)$$

رشته‌های تصادفی

همان‌طور که می‌دانیم، تصمیم‌ناپذیری مسأله‌ی توقف از مسائل مهم از نظریه‌ی بازگشتی است. بیان این مسأله در نظریه‌ی اطلاعاتی الگوریتمیک به شکل زیر است:

نمایان دوتایی احتمال توقف U، یک رشته‌ی نامتناهی پیچیده‌ی بیشینه است.

در این بخش تلاش بر قاعده‌مند نمودن دقیق این بیان و اثبات آن داریم.

قضیه‌ی ۱۳: این قضیه محدودیت‌هایی بر پیچیدگی اعداد طبیعی قرار می‌دهد:

$$a)\ \sum_n 2^{-H(n)} \le 1$$

تابع زیر را در نظر بگیرید:

$$b)\ if\ \sum_n 2^{-f(n)}\ diverges\ (= \infty), then{:}\ H(n) > f(n)\ infinitely\ often$$

$$c)\ if\ \sum_n 2^{-f(n)}\ converges\ (< \infty), then{:}\ H(n) \ge f(n) + O(1)$$

! با این تفاصیل می‌توان تابع پیچیدگی اندازه‌ی برنامه را نمونه‌ای مناسب از یک تابع مینیمال محاسبه‌پذیر در شکل حدی دانست که در مرز بین همگرایی و واگرایی رابطه‌ی زیر دانست:

$$\sum 2^{-H(n)}$$

قضیه‌ی ۱۴: این قضیه روابط زیر را در مورد رشته‌های بیتی با حداکثرپیچیدگی بیان می‌دارد:

$$a)\ Max\ H(s)\{|s| = n\} = n + H(n) + O(1)$$
$$b)\ \#\{s{:}\ |s| = n\ \&\ H(s) \le n + H(n) - k\} \le 2^{n-k+O(1)}$$

c) با در نظر گرفتن این‌که رشته‌ی نامتناهی α از پرتاب یک سکه‌ی متوازن برای هر بیت‌اش ساخته می‌شود، داریم:

برای تمام مقادیر n به‌جز تعداد متناهی از آن‌ها: $H(\alpha_n) > n$

نتیجه‌گیری

براساس تمام موارد بیان شده در این نگاشته، می‌توان پیچیدگی مکانی مناسبی برای برنامه‌های خود در هر زبان برنامه‌نویسی تعریف نماییم. این تعریف علاوه بر دربر داشتن تمام خصوصیات تعاریف قبلی در زیر شاخه‌ی نظریه‌ی اطلاعاتی الگوریتمیک، کاربردهای دیگری نیز دارد و به بسیاری از فعالیت‌های دیگر ما سهولت می‌بخشد. به‌عنوان مثال، در زیر تعریفی برای میزان تصادفی بودن یک رشته با استفاده از این تابع ارایه می‌کنیم.

تصادفی‌بودن:

برای پاسخ دادن به سؤال در مورد میزان تصادفی بودن یک رشته نظیر S می‌توان به این سؤال بازگشت که: «مقدار تابع H(S) چه میزان به حداکثر مقدار بیشینه‌اش در اندازه‌ی خود نزدیک است؟»

برای رشته‌های متناهی، رشته بیشینه‌ی تصادفی است، اگرر:

$$H(s) \cong |s| + H(|s|)$$

برای رشته‌های نامتناهی، رشته بیشینه‌ی تصادفی است، اگرر:

$$\exists\, c\ \forall n: H(\alpha_n) > n - c$$

یک تقریب بسیار کارآمد برای انتخاب بین تصادفی بودن و عدم تصادفی بودن هنگامی است که:

$$H(s) \cong |s|$$