تحلیل متوسط الگوریتم Boyer-Moore

سید ابراهیم هاشمیان *محمد قدسی*

دانشکدهٔ مهندسی کامپیوتر
دانشگاه صنعتی شریف

چکیده

الگوریتم Boyer-Moore یکی از بهترین روش‌ها حل مسالهٔ تطابق رشته‌ها (String Matching) است و این امر به‌کمک نتایج تجربی زیاد نشان داده شده است. در این مقاله با استفاده از امید ریاضی، هزینهٔ متوسط اجرای این الگوریتم را محاسبه می‌کنیم و با این‌که الگوریتم در بدترین حالت مانند ساده‌ترین الگوریتم تطابق رشته‌ها رفتار می‌کند، نشان می‌دهیم که هزینهٔ متوسط الگوریتم تقریباً برابر است با هزینهٔ الگوریتم ساده در بهترین حالت، که این مؤید نتایج تجربی است.

کلیدواژه‌ها: تطبیق رشته‌ها، الگوریتم Boyer-Moore، تحلیل الگوریتم.

۱ مقدمه

جست‌وجوی یک کلمه در داخل یک متن، که تحت عنوان مسألهٔ تطابق رشته‌ها شناخته شده است یکی از زمینه‌های مهم در الگوریتم‌هاست که به دلیل کاربردهای گستردهٔ آن در موتورهای جست‌وجو و یاجست‌وجوی الگوهای تصویری در یک تصویر مرجع، مورد توجه قرار گرفته است. الگوریتم‌های متعددی برای این مسأله معرفی شده است و زمان متوسط برخی از آنها توسط روش‌های ریاضی محاسبه شده است تا معیاری جهت مقایسهٔ آن‌ها باشد. یکی از معروف‌ترین و بهترین الگوریتم‌های معرفی شده در این زمینه الگوریتم Boyer Moore است [۱] که با تجربه ثابت شده است که جزء سریع‌ترین روش‌های حل این مسأله است.

در این مقاله به کمک روشی جدید، هزینهٔ متوسط اجرای این الگوریتم محاسبه می‌شود و در نهایت نشان می‌دهیم که هزینهٔ اجرای متوسط آن تقریباً برابر است با هزینهٔ بهترین حالت اجرای الگوریتم ساده و این پاسخی است به این پرسش که چرااین الگوریتم همواره بسیار عالی عمل کرده است.

در بخش ۲ مقاله الگوریتم Boyer-Moore شرح داده می‌شود و در بخش ۳ تحلیل الگوریتم بیان خواهد شد و سرانجام در بخش ۴ خلاصه و نتیجه گیری می آید.

۲ الگوریتم Boyer-Moore

الگوریتم Boyer-Moore برای جست‌وجوی رشتهٔ P (با طول m) در متن T (با طول n)، رشتهٔ P را در ابتدای متن T قرار می‌دهد و مقایسهٔ رشتهٔ P با زیررشتهٔ متناظرش از متن T، را (به‌جای این‌که مثل الگوریتم‌های دیگر از اولین نویسه آغاز نماید) از آخرین نویسه شروع می‌کند و در صورت یکسان بودن این دو نویسه، نوبت به مقایسهٔ دو نویسهٔ ماقبل آخر می‌رسد و ... تا هنگامی که یا رشتهٔ مورد نظر پیدا شود و یا این‌که به دو نویسهٔ غیر همسان برسیم [۲] [۳]. اگر رشتهٔ مورد نظر پیدا شد، جست‌وجو برای یافتن نمونه‌ای دیگر از رشتهٔ P در متن T ادامه می‌یابد، اما اگر پس از مشاهدهٔ k نویسهٔ یکسان، $k+۱$اُمین نویسه‌ها متفاوت باشند، لازم است که رشتهٔ P جلوتر برود و مقایسهٔ حروف در موقعیت جدید آغاز شود. برای محاسبهٔ میزان جلو رفتن رشتهٔ P، الگوریتم Boyer-Moore از دو روش مکاشفه‌ای استفاده می‌کند.

روش مکاشفه‌ای اوّل که روش مکاشفه‌ای «نویسهٔ بد» نام دارد با توجه به $k+۱$اُمین نویسهٔ متفاوت از متن T، و با توجه به مکان رخداد هر یک از حروف الفبا در رشتهٔ P (که در هنگام دریافت رشتهٔ P توسط تابع `Compute_Last_Occurance` محاسبه شده است) رشتهٔ P را تا مکان مشاهدهٔ $k+۱$اُمین نویسهٔ متفاوت جلو می‌برد.

روش مکاشفه‌ای دوّم هم که «پسوند خوب» نام دارد با توجه به k نویسه یکسان مشاهده شده، و با دانستن مکان بعديِ تکرار این زیر رشتهٔ k نویسه در رشتهٔ P، (که در هنگام دریافت رشتهٔ P توسط تابع `Compute_Good_Suffix` محاسبه شده است) رشتهٔ P را تا مکانی که این زیر رشتهٔ k نویسه، مجدداً مشاهده شود جلو می‌برد. می‌بینیم که هر یک از این دو مکاشفه برای جهش رشتهٔ P، مقداری را پیشنهاد می‌کنند. از بین این دو مقدار، عدد بزرگتر انتخاب می‌شود و پس از جهش رشتهٔ P به موقعیّت جدید، بار دیگر مقایسهٔ رشتهٔ P با زیررشتهٔ متناظرش از متن T، از آخرین نویسه آغاز می‌شود.

برنامهٔ کامپیوتری این الگوریتم را در شکل ۱ مشاهده می‌کنید. (متغیر s اندیس موقعیت ابتدای رشتهٔ P نسبت به ابتدای متن T است.)

هزینهٔ سطر ۳ به‌راحتی قابل مقایسه است و مقدار آن برابر با $O(|\Sigma|+m)$، هزینهٔ سطر ۴ هم برابر است با $O(m)$. در حقیقت هزینهٔ اصلی این الگوریتم هزینهٔ سطرهای ۱۳—٦ است که هدف از این مقاله محاسبهٔ آن است.

۳ تحلیل متوسط الگوریتم

هدف از این بخش، محاسبهٔ زمان متوسط اجرای این الگوریتم است یعنی اگر چندین دفعه این الگوریتم را با داده‌های تصادفی اجرا کنیم، متوسط زمان اجرای آن چه‌قدر خواهد بود؟ براساس یکی از قضایای بنیادین علم احتمالات، زمان متوسط فوق برابر است با امید ریاضی الگوریتم که بر اساس فرمول $E=\sum_{i=۱}^{k} P_iS_i$ محاسبه می‌شود، که P_i احتمال رخدادن وضعیت i و S_i هزینهٔ اجرای الگوریتم در صورت وقوع وضعیت i می‌باشد. پس اگر بتوانیم متوسط میزان جلو

```
Boyer_Moore(T,P,Σ)
1    n = Length(T)
2    m = Length(P)
3    λ = compute_last_occurance_function(P,m,Σ)
4    δ = compute_good_suffix_function(P,m)
5    s = 0
6    while s ≤ n − m
7         do j = m
8         while j > 0 and P[j]=T[s+j]
9              do j = j-1
10                if j = 0
11             then print "pattern occurs at shift" s
12                s = s+1
13             else s = s+max(δ[j],j-λ[T[s+j]])
```

شکل ۱: الگوریتم Boyer-Moore

رفتن رشتهٔ P به‌ازای هر مقایسه را به‌دست بیاوریم و عددی مثل E شود، پس کل متن T با $\frac{n}{E}$ بار مقایسه پیموده می‌شود و متوسط هزینهٔ سطرهای ۱۳—۶ الگوریتم برابر خواهد بود با$O(\frac{n}{E})$.

برای محاسبهٔ E می‌دانیم که متوسط میزان جلو رفتن رشتهٔ P در صورت مقایسهٔ بین T[s+i] و P[i]، خود یک امید ریاضی است که به مقدار i بستگی دارد پس متوسط میزان پرش P را می‌توان به‌صورت مجموع احتمال وقوع رخداد مقایسه بین P[i] و T[s+i]، ضربدر میزان پرش ناشی از این مقایسه در نظر گرفت. شکل ریاضی جملهٔ فوق به این صورت است: $E = \sum_{i=0}^{m-1} A(i)B(i)$ که در آن $A(i)$ احتمال وقوع مقایسهٔ عنصر P[i] با T[s+i] و $B(i)$ متوسط میزان پرش ناشی از این مقایسه می‌باشد.

۱.۳ محاسبهٔ $A(i)$

$A(i)$ احتمال وقوع مقایسه بین عنصر P[i] و T[s+i] است. مقایسهٔ P[i] و T[s+i] هنگامی ضرورت می‌یابد که کلیهٔ نویسه‌های قبل از i در رشتهٔ P با نویسه‌های متناظرشان در متن T، یکسان بوده باشند. (چون مقایسه، از انتهای رشتهٔ P آغاز می‌شود پس منظور از $i = 0$، در حقیقت مقایسهٔ آخرین نویسهٔ P است.) از آن‌جا که می‌دانیم احتمال یکسان بودن نویسه که از مجموعهٔ Σ انتخاب شده و در T[s+i] قرار گرفته با نویسه متناظرش در P[i] برابراست با $\frac{1}{|\Sigma|}$ (به‌ازای هر j بین ۰ تا $i-1$) پس احتمال یکسان بودن کلیهٔ موقعیت‌های j به‌طور هم‌زمان، حداکثر برابر است با: $A(i) = \frac{1}{|\Sigma|^i}$. پس با احتمال $A(i)$ مقایسهٔ بین دو نویسهٔ P[i] و T[s+i] لازم می‌شود.

۲.۳ محاسبهٔ $B(i)$

$B(i)$ متوسط میزان پرش ناشی از مقایسهٔ بین عنصر P[i] و T[s+i] است. بدیهی است که میزان پرش به نتیجهٔ این

مقایسه بستگی دارد. اگر دو عنصر یکسان باشند، میزان پرش صفر است و نوبت به مقایسهٔ بعدی می‌رسد پس به‌احتمال $\frac{۱}{|\Sigma|}$ میزان پرش برابر صفر است ولی به‌احتمال $۱ - \frac{۱}{|\Sigma|}$ دو عنصر با هم متفاوت هستند و پرشی صورت می‌گیرد. اگر متوسط میزان پرش ناشی از یکسان نبودن دو عنصر را $C(i)$ بنامیم، مقدار $B(i)$ به صورت زیر محاسبه می‌شود:

$$B(i) = \frac{۱}{|\Sigma|} * (۰) + (۱ - \frac{۱}{|\Sigma|}) * C(i) = (۱ - \frac{۱}{|\Sigma|}) * C(i)$$

۳.۳ محاسبهٔ $C(i)$

$C(i)$ متوسط میزان پرش ناشی از یکسان نبودن عنصر P[i] و T[s+i] است. طبق الگوریتم Boyer-Moore، $C(i)$ برابر ماگزیمم مقدار پرش حاصل از دو روش مکاشفه‌ای «نویسهٔ بد» و «پسوند خوب» است. اگر متوسط پرش ناشی از «نویسهٔ بد» را $D(i)$ و متوسط پرش ناشی از «پسوند خوب» را $E(i)$ بنامیم، $C(i)$ به شکل $C(i) = \max(D(i), E(i))$ محاسبه می‌شود.

۴.۳ محاسبهٔ $D(i)$

$D(i)$ را برابر متوسط پرش ناشی از «نویسهٔ بد» در صورت یکسان نبودن عنصر P[i] و T[s+i] تعریف کردیم. بدیهی است که اگر رشتهٔ P به‌اندازهٔ کافی بزرگ باشد، و توزیع نویسه‌ها نیز به‌صورت یکنواخت باشد، دورهٔ تناوب هر نویسه $|\Sigma|$ است و با شروع جست‌وجو از هر موقعیت P، به‌احتمال $\frac{۱}{|\Sigma|}$ در اولین مقایسه، نویسه مورد نظرمان را پیدا می‌کنیم که در این‌صورت میزان پرش برابر ۱ است، به‌احتمال $\frac{۱}{|\Sigma|}$ در دومین مقایسه، نویسهٔ مورد نظرمان را پیدا می‌کنیم که در این‌صورت میزان پرش ۲ است و ... و به‌احتمال $\frac{۱}{|\Sigma|}$ در $|\Sigma|$اُمین مقایسه، نویسهٔ مورد نظرمان را پیدا می‌کنیم که در این‌صورت میزان پرش $|\Sigma|$ است. پس متوسط میزان پرش برابر است با:

$$\frac{۱}{|\Sigma|} * ۱ + \frac{۱}{|\Sigma|} * ۲ + \cdots + \frac{۱}{|\Sigma|} * |\Sigma| = \frac{|\Sigma| + ۱}{۲}$$

یا همان $\frac{|\Sigma|}{۲}$. اما اگر طول P کم‌تر از $|\Sigma|$ باشد آن‌گاه میزان پرش از $m - i$ بیشتر نمی‌شود. بنابراین رابطهٔ فوق به‌صورت زیر در می آید:

$$\frac{۱}{|\Sigma|} * ۱ + \frac{۱}{|\Sigma|} * ۲ + \cdots + \frac{۱}{|\Sigma|} * (m-i) + \cdots + \frac{۱}{|\Sigma|} * (m-i)$$

$$= \sum_{i=۱}^{m-i} \frac{i}{|\Sigma|} + \frac{(|\Sigma| - m + i) * (m-i)}{|\Sigma|}$$

$$= \cdots = (m-i) * (۱ - \frac{m-i-۱}{۲ * |\Sigma|})$$

معادلات به‌دست آمده را می‌توان به‌شکل زیر خلاصه کرد:

$$D(i) = min(m-i, |\Sigma|) * (۱ - \frac{min(m-i, |\Sigma|)}{۲ * |\Sigma|})(i >= ۰)$$

۵.۳ محاسبهٔ $E(i)$

$E(i)$ متوسط پرش ناشی از «پسوند خوب» در صورت یکسان نبودن عنصر P[i] و T[s+i] است. در این‌جا هم اگر رشتهٔ P به‌اندازهٔ کافی بزرگ باشد، میزان جلو رفتن برای یافتن مجدد زیر رشتهٔ تطبیق یافته در رشتهٔ P (براساس استدلالی مشابه) برابر است با $\frac{|\Sigma|^i}{۲}$ و در صورت کم‌تر بودن طول رشتهٔ P از $|\Sigma|^i$ (هم‌چون محاسبهٔ بخش دوم $D(i)$) برابر خواهد شد با $(m-i) * (۱ - \frac{m-i-۱}{۲*|\Sigma|^i})$.

معادلات به‌دست آمده را می‌توان به شکل زیر خلاصه کرد:

$$E(i) = min(m-i, |\Sigma|^i) * (۱ - \frac{min(m-i, |\Sigma|^i)}{۲ * |\Sigma|^i})(i >= ۱)$$

۴ نتیجهٔ نهایی و ساده‌سازی

اکنون که محاسبات لازم انجام شده‌است می‌توان با ترکیب معادلات به‌دست آمده به فرمول نهایی رسید. فرمولی که در زیر مشاهده می‌شود هزینهٔ متوسط الگوریتم است:

$$\frac{n}{\sum_{i=۰}^{m-۱} (\frac{|\Sigma|-۱}{|\Sigma|^{i+۱}}) * max(D(i), E(i))}$$

که در آن،

$$D(i) = min(m-i, |\Sigma|) * (۱ - \frac{min(m-i, |\Sigma|)}{۲ * |\Sigma|})(i >= ۰)$$

$$E(i) = min(m-i, |\Sigma|^i) * (۱ - \frac{min(m-i, |\Sigma|^i)}{۲ * |\Sigma|^i})(i >= ۱)$$

از آن‌جایی‌که رابطهٔ فوق، پیچیده‌تر از آنست که قابل مقایسه با سایر الگوریتم‌ها باشد، آن‌را ساده‌تر می‌کنیم. ابتدا با کمی توجه در می‌یابیم که به‌ازای $i = ۰$، $C(i) = D(i)$ و به‌ازای سایر مقادیر i، داریم $C(i) = E(i)$. (با در نظر گرفتن سه ناحیهٔ $|\Sigma| \leq (m-i)$، $|\Sigma| < (m-i) < |\Sigma|^i$ (چون i بزرگتر از صفر است) و $|\Sigma|^i \leq (m-i)$ و نوشتن مقدار $D(i)$ و $E(i)$ در هر ناحیه و مقایسهٔ آن‌ها صحت بیان فوق را می‌توان دریافت.)

علاوه بر این، با در نظر گرفتن $|\Sigma| - ۱$ به‌جای $|\Sigma|$، هزینهٔ متوسط الگوریتم به‌صورت زیر در می آید:

$$\frac{n}{min(m,|\Sigma|)*(۱-\frac{min(m,|\Sigma|)}{۲*|\Sigma|})+\sum_{i=۱}^{m-۱}(\frac{۱}{|\Sigma|^i})*min(m-i,|\Sigma|^i)*(۱-\frac{min(m-i,|\Sigma|^i)}{۲*|\Sigma|^i})}$$

از آن‌جا که جملهٔ دومِ مخرج، تقریباً یک تصاعد هندسی است (با قدر نسبت $\frac{۱}{|\Sigma|}$) پس حاصل سری آن، تقریباً برابر با اولین جملهٔ آن است، پس فرمول هزینه به‌صورت زیر در می آید:

$$\frac{n}{min(m,|\Sigma|)*(۱-\frac{min(m,|\Sigma|)}{۲*|\Sigma|})+\frac{min(m,|\Sigma|)}{|\Sigma|}*(۱-\frac{min(m,|\Sigma|)}{۲*|\Sigma|})}$$

از جملهٔ دومِ مخرج در برابر جملهٔ اولِ آن می‌توان چشم پوشی کرد و به فرمول نهایی زیر رسید:

$$\frac{n}{min(m,|\Sigma|)*(۱-\frac{min(m,|\Sigma|)}{۲*|\Sigma|})}$$

برای $m >= |\Sigma|$ داریم: $\frac{n}{|\Sigma|/۲}$ و برای $m < |\Sigma|$ داریم: $\frac{n}{m*(۱-\frac{m}{۲*|\Sigma|})}$

۵ نتیجه گیری

همان‌طور که مشاهده شد زمان اجرای متوسط الگوریتم Boyer-Moore تقریباً از مرتبهٔ $\frac{n}{min(m,|\Sigma|)*(۱-\frac{min(m,|\Sigma|)}{۲*|\Sigma|})}$ است. که این فرمول به‌ازای m های کوچک، تقریباً از مرتبهٔ $\frac{n}{m}$ است و با بزرگ شدن m، هزینهٔ الگوریتم افزایش می‌یابد تا سرانجام با رسیدن m به $|\Sigma|$ به $\frac{n}{|\Sigma|/۲}$ می‌رسد و در همان حد باقی خواهد ماند. از آن‌جایی‌که می‌دانیم زمان $\frac{n}{m}$، بهترین حالت (Best Case) برای حل این مسأله است، به‌راحتی می‌توان دریافت که چرا این الگوریتم در تجربه این‌قدر موفق عمل می‌کند.

مراجع

[1] R. S. Boyer, and J.S. Moore, "*A Fast String Matching Algorithm,*" CACM, 20(10):762-772, 1977.

[2] Udi Mabner, *Introduction to Algorithms: A Creative Approach*, Addison Wesley, 1989.

[3] T. Cormen, C. Leiserson, R. Riverst, *Introduction to Algorithms,* MIT Press, 1992.