دانشکده مهندسی کامپیوتر

جزوه درس

ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۲۲

جدول هش

محتبی نافذ - ۱۳۹۸/۷/۱۸

جزوه جلسه ۲۲ام مورخ ۱۳۹۸/۷/۱۸درس ساختمان‌های داده تهیه شده توسط محتبی نافذ. در جهت مستند کردن مطالب درس ساختمان های داده، بر آن شدیم که از دانشجویان جهت مکتوب کردن مطالب کمک بگیریم. هر دانشجو می‌تواند برای مکتوب کردن یک جلسه داوطلب شده و با توجه به کیفیت جزوه از لحاظ کامل بودن مطالب، کیفیت نوشتار و استفاده از اشکال و منابع کمک آموزشی، حداکثر یک نمره مثبت از بیست نمره دریافت کند. خواهش‌مند است نام و نام خانوادگی خود، عنوان درس، شماره و تاریخ جلسه در ابتدای این فایل را با دقت پر کنید.

۱.۲۲ مقدمه ای بر تابع هش

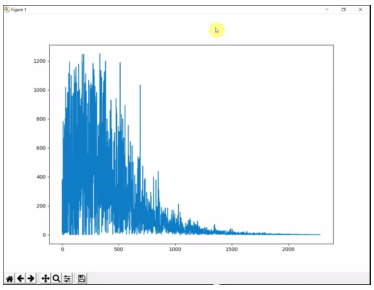
با توجه به مطالب جلسه گذشته این جلسه با ذکر چند نمونه تابع هش و مقایسه ی ان ها باهم مطالب را ادامه میدهیم
فرض کنید یک مجموعه از ۶۰۰ هزار رشته در اختیار داریم و میخواهیم یک تابع هش برای هش کردن این رشته ها پیاده سازی کنیم تا بتوانیم در یک جدول هش آن ها را دلخواه نگهداری کنیم
در اینجا چند تابع هش را بیان و نمودار مقدار هش (hash value)یکتا بر اساس تعداد collision هر hash value را رسم و توضیح میدهیم.

- یک ایده ی ساده جمع اعداد اسکی کاراکتر های موجود در رشته است

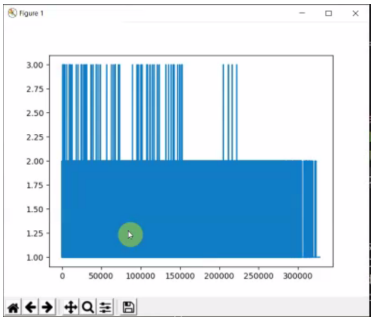
همانطور که می بینیم ۶۰۰ هزار رشته به حدود ۲۴۰۰ hash value یکتا متناظر شده که برای مثال در ۵۰۰ hash value حدود ۱۱۰۰ collisionرخ داده

- یک ایده ی دیگر هش کردن به صورت hash valueهای رندوم میباشد

همانطور که می بینیم این هش تعداد collisionبه شدت کم و پخش شده ای دارد که این عالی است اما یک ایراد بزرگ دارد این مزیت را از بین میبرد
زمانی که دو بار یک رشته را به آن میدهیم دو hash valueمتفاوت به ما میدهد پس در جدول هش ما هر مقداری را Insertکنیم نمیتوانیم دوباره ان را بازیابی کنیم



شکل ۱.۲۲: نمودار هش جمع کد اسکی



شکل ۲.۲۲: نمودار هش کاملا رندوم وار

آیا میتوانید یک تابع هش خوب برای رشته مثال بزنید؟

حال فرض کنید تعداد زیادی شماره تلفن داده شده و میخواهیم یک تابع هش برای متناظر نمودن ان ها به خانه های جدول هش پیاده سازی کنید.

- میتوان سه رقم اول هر شماره را به عنوان مقدار هش در نظر گرفت.

اما در این صورت همه ی کسانی که در یک منطقه زندگی میکنند ده یک خانه متناظر میشوند

- یک ایده ی دیگر هش کردن به صورت hash valueهای رندوم میباشد

همانطور که اشاره کردیم در این صورت مقادیر قابل باز یابی نیستند.

آیا میتوانید یک تابع هش خوب برای اعداد مثال بزنید؟

۲.۲۲ ویژگی های تابع هش خوب

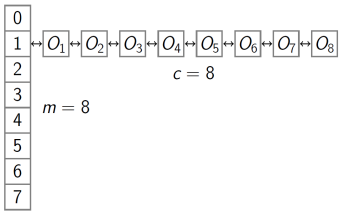
- یک مقدار را همیشه به یک hash valueمتناظر کند

- مقدار هش سریع محاسبه شود
- در تعداد مقادیر زیاد مقادیر را به صورت توزیع شده ای به هش های گوناگون متناظر کند
- collision آن کم باشد

نکته : باید توجه داشت که هر تابع هشی با توجه به یک مجوعه دیتا ممکن است خوب باشد یا بد اما ما باید طوری تابع هش پیاده سازی کنیم که عموما خوب باشد

۳.۲۲ denied of service attack (DOS)

DOS یک نوع حمله سایبری برای از کار انداختن یا کند کردن پاسخگویی یک سرور است به طوری که کاربران به صفحه ی وب یا اینترنت یا ایمیل و ... دسترسی نداشته باشنند.
اساس کار این حمله ایجاد collision زیاد است.
هر تابع هشی را میتوان ورودی هایی را داد که در نهایت همه ی ان ها به یک مقدار هش شوند شاید این کار سخت باشد اما امکان پذیر است



شکل ۳.۲۲: نوعcollisionبرای dos attack

در پارس کردن فایل ها در سرور ها بسیار از فایل jsonزاستفاده میشود این فایل نوعی جدول هش است حمله کننده اگر بتواند تعداد زیادی collisionپیدا کند میتواند در کار سرور اخلال ایجاد نماید سرور با جستجو کردن در این LinkListدچار مشکل خوهد شد. شرط این کار این است که حمله کننده تعداد زیاد رشته از قبل به سرور بدهد و خروجی هش را ببیند و بتواند تعدادی رشته ی collisionدار پیدا نماید در سال ۲۰۱۱ شرکت مایکروسافت برای حل این مشکل امنیتی از UseRandomizedStringHashAlgorithmبرای غیر قابل پیش بینی کردن collisionها در پیاده سازی جدول هش استفاده کرد. که در ادامه توضیح خواهیم داد.

۴.۲۲ Universal Family

ایده : یک مجموعه از توابع هش داریم و هر دفعه به طور رندوم یکی از این توابع را انتخاب میکنیم
فرض کنید یک مجموعه ی Uداریم که شامل ۰،۱،۲،....،m-۱ است
حال hرا یک تابع هش در نظر بگیرید که هر ورودی را به عددی عضو Uمتناظر میکند.
حال فرض کنید Hیک مجموعه از این توابع هش h است یعنی:

```
{ { 1-m, ..., 2, 1, 0 } <- U : h } = H
```

H, Universal family است اگر برای برای دو ورودی مختلف احتمال collision کم تر از 1/m است

```
P[h(x)=h(y)]<=1/m
```

x , y عضو U و x != y

Load Factor ۵.۲۲

به نسبت تعداد مقادیر ذخیره شده در جدول هش به سایز کل جدول هش load factor می گویند
e = n / m
نکته : اگر جدول هش یک universal familyبود میانگین زمان دسترسی به طولانی ترین زنجیره (chain)از order O(1 + e) میباشد
کاربرد دیگر load factorایرای مدیریت حافظه است
به این صورت که سعی میشود این نسبت بین ۵.۰ تا ۹.۰ باشد اگر بیشتر شد باید برای جلوگیری از collisionسایز جدول هش را افزایش داد

[1] که شبه کد Rehash کردن در زیر امده است. ؟؟

```
Data: T
Result: new table
Rehash( T ):
loadFactor = T.numberOfKeys/T.size;
if not at end of this document then
    Create new Tnew of size, Tsize * 2;
    Choose hnew with cardinality of Tnew size;
    while T not empty do
        Insert object in Tnew using new hnew;
    end
end
T = Tnew, h = hnew;
```

Algorithm 1: How to Rehash hash table?

مرتبه ی زمانی :O(n)
مرتبه زمانی تحلیل زمانی : O(1)

یک تابع هش universal family برای اعداد ۶.۲۲

```
{ m mod ( p mod (ax+b) ) = hp(x) } = Hp
```

p یک عدد رندوم اسست.
a, b هر عددی بین صفر تا p-1 هستند
مثال : a = 34, b= 2, p=10 000 019 فرض کنید عدد مورد نظر x= 1 482 567باشد
(۲ + ۱۴۸۲۵۶۷ * ۳۴) mod ۱۰۰۰۰۰۱۹ = ۴۰۷۱۸۵
۴۰۷۱۸۵ mod ۱۰۰۰ = ۱۸۵

[1] pseudocode

۱۸۵ =h(x)

۷.۲۲ یک تابع هش universal family برای رشته ها

تیپ کلی این مجموعه از توابع هش :

```
{ p mod sum(S[i]pow(x,i)) = hp(s) } = Pp
```

i از صفر تا اندازه ی رشته منها ی یک می باشد. S نام رشته ورودی است
p یک عدد اول ثابت و xبین یک تا p-1 است.
این هش به Polynomial Hashing معروف است.

[۲] شبه کد PolyHash ؟؟

Data: S,p,x
Result: hash
hash = 0;
for *i from |S| -1 down to 0* **do**
  hash = (hash * x + S[i]) mod p;
end
return hash;

Algorithm 2: How to implement PolyHash hash function ?

مثال: ۳ = |S|
۰ = hash
S[۲] mod p = hash
S[۲]x + S[۱] mod p = hash
S[۲]pow(x،۲) + S[۱]x + S[۰] mod p = hash

۸.۲۲ یافتن یک زیر رشته در یک رشته

سوال : یک زیررشته `P (pattern)` و یک رشته ی `T (Text)` داده شده و باید تمام ایندکس هایی که زیررشته در رشته قرار دارد را خروجی دهید.

راه حل معمولی :
مرحله ی اول یک تابع برای تشخیص برابری دو رشته مینویسیم:

[۲] pseudocode

[۳] شبه کد AreEqual ؟؟

```
Data: S1,S2
Result: boolean(true or false)
if |S1| != |S2| then
    return False;
end
for i from 0 to |S1| -1 do
    if S1[i] != S2[i] then
        return False;
    end
end
return hash;
```

Algorithm 3: Are Equal tow string ?

زمان اجرای کد بالا O(|P|) است.
مرحله بعد پیاده سازی راه حل است :

[۴] شبه کد FindSubStringNative ؟؟

```
Data: T,P
Result: positions
positions = empty list;
for i from 0 to |T| - |P| do
    if AreEqual(T[i..i+|P|-1], P) then
        positions.Append(i);
    end
end
return positions;
```

Algorithm 4: positions of pattern in text

زمان اجرای کد بالا O(|T||P|) است.
اما راه حل بهینه تر استفاده از هشینگ است.

۹.۲۲ الگوریتم RabinKarp

ایده : به جای مقایسه خود زیر رشته های رشته اصلی با الگو میدانیم اگر مقدار هش ان ها برابر نبود پس قطعا باهم برابر نیستند.
ولی اگر برابر بودند چاره ای جز مقایسه کاراکتر به کاراکتر نیست.
الگوریتم اولیه :

---

[۳] pseudocode
[۴] pseudocode

[۵] شبه کد initial RabinKarp ؟؟

```
Data: T,P
Result: positions
p = big prime , x = random(1, p-1);
positions = empty list;
pHash = PolyHash(P, p, x);
for i from 0 to |T| - |P| do
    tHash = PolyHash( T[i..i+|P|-1], p, x);
    if pHash != tHash then
        continue;
    end
    if AreEqual(T[i..i+|P|-1],P ) then
        positions.Append(i);
    end
end
return positions;
```

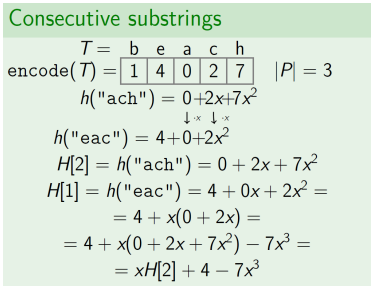
Algorithm 5: positions of pattern in text

هشدار : زمانی که است که دو زیر رشته باهم برابر باشند که در این صورت زمان اجرا بیشتر میشود ولی یادمان باشد که تعداد collision در یک هشuniversal family به شدت کم است

نکته : زمان اجرای الگوریتم بالا هم تقریبا همان $O(|T||P|)$ است. برای سریع تر شدن چه کنیم؟

ایده RabinKarpزیر رشته های متوالی یک رشته مقدار هش شان بسیار شبیه هم بوده و میتوان ارتباطی برای زود تر حساب شدن هش ها پیدا کرد.

توجه کنید که تابع هش مورد استفاده PolyHash می باشد
Pp = { hp(s) = sum(S[i]pow(x،i)) mod p }



شکل ۴.۲۲: ارتباط بین مقدار هش زیر رشته های متوالی

در نهایت رابطه ی بازگشتی زیر بین زیر رشته های متوالی برداشت میشود:

```
H[i] = x*H[i+1] + ( T[i] - (T[i+|P|] * pow(x, |P|)) mod p] )
```

که pow(x, |P|) را میتوان یکبار حساب و بعدا فقط استفاده نمود

[۵] pseudocode

برای شروع کد زنی باید قیل ااز حل تابعی پیاده سازی شود که با روش سریع و با استفاده از رابطه ی باگشتی کل هش زیر رشته ها را محاسبه کند.

[۶] شبه کد PreComputeHashes ؟؟

```
Data: T,|P|,p,x
Result: H
H = array of length |T| - |P| + 1 ;
S = T[|T| - |P| ... |T|-1];
H[|T| - |P|] = PolyHash(S, p, x);
y = 1;
for i from 1 to |P| do
    y = (y*x) mod p;
end
for i from |T|-|P|-1 down to 0 do
    H[i] = (x*H[i+1]+T[i]-y*T[i+|P|] mod p ;
end
return H;
```

Algorithm 6: hash of substring

حال ربین کارپ اصلی را بازنویسی میکنیم:

[۷] شبه کد main RabinKarp ؟؟

```
Data: T,P
Result: positions
p = big prime , x = random(1, p-1);
positions = empty list;
pHash = PolyHash(P, p, x);
H = PreComputeHashes(T, |P|,p,x);
for i from 0 to |T| - |P| do
    if pHash != H[i] then
        continue;
    end
    if AreEqual(T[i..i+|P|-1],P ) then
        positions.Append(i);
    end
end
return positions;
```

Algorithm 7: positions of pattern in text

زمان اجرای الگوریتم بالا از مرتبه ی زمانی $O(|T|+(q+1)*|P|)$ است که چون معمولا q کوچک است پس مرتبه ی آن خیلی کم تر از $O(|T|*|P|)$

همچنین سعی کنید [۱]. علاوه بر مراجع چنانچه ابزار یا وب‌سایت قابل توجهی موجود است خوب است به آن هم ارجاع دهید [۲].

---

[۶] pseudocode
[۷] pseudocode

Bibliography

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.

[2] D. Galles, "Data structure visualizations." `https://www.cs.usfca.edu/~galles/visualization/StackArray.html`. Accessed: 2019-12-10.