Cuckoo Hashing

پویا میرزایی زاده
امید سلطانی نژاد

دیدکلی:

Cuckoo Hashingطرحی برای حل برخورد کردن (زمانی که دوقسمت از داده در Hash table از یک کلید استفاده کنند.)مقادیر توابع هش در Hash table ،با زمان جستوجوی ثابت در بدترین حالت است. نام این روش برگرفته از گونه ای ازپرندگان است.که در آن جوجه ها هنگام بیرون آمدن از لانه تخم ها یا دیگر جوجه ها را ها می دهند.دقیقا مانند قرار دادن یک کلید جدید در جدول کوکو هشینگ است. که ممکن است کلید های قبلی ها را در جدول جا به جا کند.

تاریخچه

اولین بار توسط Rasmus Pagh, Femming FricheRodler در سال 2001در مقاله کنفرانسی ارائه شد.
این مقاله در سال 2020 جایزه ی تست زمان را در European Symposium on Algorithmsدریافت کرد.

عملیات ها:

Cuckoo hashingنوعی ازopen addressingاست که در آن خانه های جدول هش با یک کید یا یک کید و مقدار پر شده است.
تابع هش(Hash function) برای تعیین کردن مکان هر یک از کلید در جدول استفاده می شود،و مقدار مربوط به کلید یا حضورش در جدول با برسی آن خانه می توان پیدا کرد.
ایده Cuckoo Hashing برای حل مشکل برخورد ها در open addressing(زمانی رخ می دهد که به یک خانه دو کلید داده می شود.) با استفاده از دو تابع هش به جای یک تابع است. این راه دو مکان مناسب در جدول هش برای هر کلید تولید می کند.در الگوریتم رایج ، Hash Tableبه دو جدول هم اندازه تقسیم می شود و هر تابع یک index در یکی از این دو جدول می سازد. ولی این امکان وجود دارد که هر دو تابع در یک جدول indexبگذارند.

عملیات ها:

- جستوجو

درCuckoo hashingبا دو جدول هشT1, T2با اندازه ی r توابع به صورت زیر تعریف می شوند.

$$h_1,\ h_2\ :\ \cup \rightarrow \{0,\ldots,r-1\} \text{ and } \forall x \in S$$

به طوری که xکلید و S مجموعه ایست که کلید ها در آن به طوری که $h_1(x)$ of T_1 or $h_2(x)$ of T_2باشد ذخیره می شوند.

تابع جستوجو:

```
function lookup(x) is
  return T1[h1(x)] = x ∨ T2[h2(x)] = x
end function
```

به دلیل وجود "یا" بدترین حالت O(1)خواهد بود.

عملیات ها:

- حذف:

حذف با O(1) انجام می شود. اگر که اندازه جول کوچک باشد هزینه ی تقسیم کردن جدول محاسبه نمی شود.

- الحاق:

اولین قدم در اضافه کردن یک ایتم جدید چک کردن این است که آیا $h_1(x)$ در جدول T_1 اشغال شده است یا خیر؟

اگر که اشغال نشده باشد که آیتم جدید را همان جا اضافه می کنیم اگر که اشغال شده باشد آیتم جدید جایگزین آیتم قبلی می شود و آیتم قبل با همین روش در جدول T_2 قرار می گیرد. این روند تا جایی انجام می شود تا جای خالی پیدا شود.

برای جلوگیری از تکرار نامحدود MAX_LOOP به صورت زیر تعریف می شود که اگر تکرار از یک حد مشخص بالا تر رفت هر دو جدول T_1 و T_2 دوباره دوباره با دو تابع جدید هش می شوند و پروسه ی الحاق دوباره تکرار می شود.

```
1    function insert(x) is
2      if lookup(x) then
3        return
4      end if
5      loop Max-Loop times
6        if T_1[h_1(x)] = ⊥ then
7          T_1[h_1(x)] := x
8          return
9        end if
10       x ↔ T_1[h_1(x)]
11       if T_2[h_2(x)] = ⊥ then
12         T_2[h_2(x)] := x
13         return
14       end if
15       x ↔ T_2[h_2(x)]
16     end loop
17     rehash()
18     insert(x)
19   end function
```

در خط 10 و 15 می توانیم نمایی از Cuckoo Hashing را ببینیم که کلید ها جای هم دیگر را گرفته تا جایی که همه کلید ها جای مشخصی داشته باشند.

علامت $\leftrightarrow$ نشان دهنده ی swapاست.

تئوری

عملیات الحاق در زمان مورد نظر انجام می شود حتی اگر باز سازس جدول را در نظر بگیریم تا وقتی که تعداد کلید ها کمتر از نیمی از ظرفیت جدول باشد ظریب بارگذاری کمتر از 50% است.

یکی از روش های اثبات استفاده از تئوری گراف های تصادفی است

تئوری

از آنجایی که یک تابع هش تصادفی به فضای زیادی نیاز دارد یک سوال که پیش می اید این است که کدام تابع هش برایCuckoo hashing کافی است؟

یک راه استفاده از K-independent hashingاست $O(\log n)$کافی است وحداقل 6_independence hashingمورد نیاز است.

و یک روش دیگر در سال 2014 نشان داد که اگر با روش Stash کمی جدول را تغییر دهیم چیزی بیشتر از 2_independent hash نیاز نخواهیم داشت.