دانشکده مهندسی کامپیوتر

جزوه درس

ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۲۰

جدول هش / hash table

نگار زین العابدین - ۱۳۹۸/۹/۴

جزوه جلسه ۲۰ام مورخ ۱۳۹۸/۹/۴درس ساختمان‌های داده تهیه شده توسط نگار زین العابدین. 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.

۱.۲۰ نکته هایی از مطالب قبل

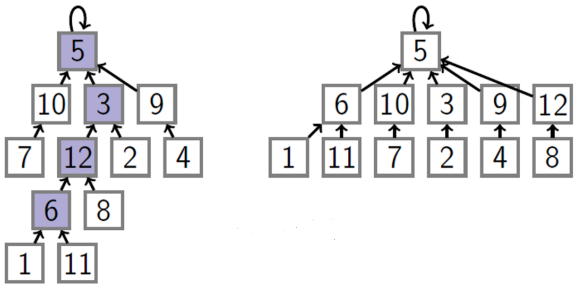
- در مبحث path copression ،وقتی از find استفاده می کنیم ، ساختار درخت ما عوض می شود.در حالی که تنها زمانی این اتفاق می افتاد ، (rank ما تغییر می کرد) موقع merge بود.چون وقتی به سمت ریشه حرکت می کنیم ، همه ی راس های سر راه را بچه ریشه قرار می دهیم.در نتیجه ارتفاع کم تر می شود.(موقع find دست به rank نمی زنیم.)
- وقتی که n تا node داشته باشیم ، اون node هایی که rank شون k باشه ، حداقل ۲ به توان k تا node داریم.

۲.۲۰ function hash

- هش فانکشن به چه دردی می خورد؟

۱ Block Chain

۲ در زبان های برنامه نویسی ، برای hash function ، data structure هایی پیاده سازی شده است.برای مثال در زبان سی شارپ، dictionary و یا در جاوا ، hash map می باشد.



۳ serach کردن پوشه:به این صورت که وقتی نام پوشه را وارد می کنیم ، نام تبدیل به هش شده و سپس مکان پوشه جست و جو می شود.

۴ برای امضا رقمی کردن

۵ یکی دیگر از کاربرد های hash function ، این است که وقتی تلفن زنگ می خورد ، متوجه این بشویم که شماره متعلق به کیست. اگر دفترچه تلفن گوشی یک hash table داشته باشد ، با استفاده از هش شماره متوجه می شویم که شماره متعلق به چه کسی است و هم چنین بالعکس.(اسم را داریم و شماره اسم را می خواهیم.)

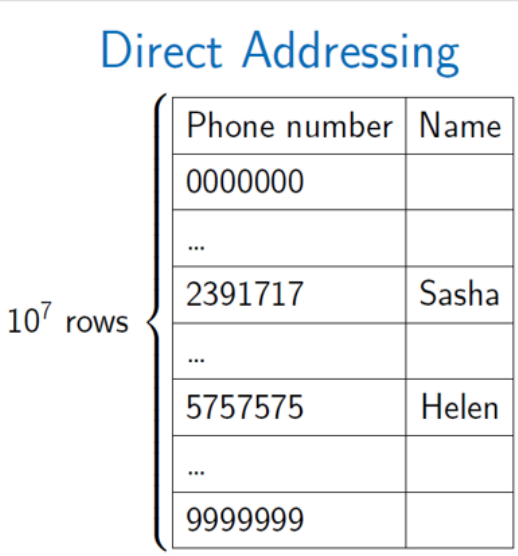
۳.۲۰ Addressing Direct

- همان طور که در بالا توضیح داده شد ، در ذخیره سازی شماره تلفن ها و پیدا کردن نام شماره مورد نظر ، می توانیم از روش Direct Addressing استفاده کنیم . در این روش برای این که به صورت بهینه کار ذخیره سازی و جست و جو را انجام دهیم ، می توانیم شماره را تبدیل کنیم به یک عدد مثلا ۷ رقمی و سپس یه آرایه به اندازه ۱۰ به توان ۷ درست کنیم و بعد به مکان آن عدد ۷ رقمی در آرایه برویم که آن جا نام شماره ذخیره شده است.

  *یکی از مشکلاتی که این روش دارد این است که فضای زیادی را در بر می گیرد.پس باید به دنبال روش دیگری بود. البته می توان از ساختمان داده هایی که تا الان خوانده ایم ، استفاده کنیم مانند Link List,Sorted Array,... . اما ممکن است که پیچیدگی محاسباتی بعضی از عملگر ها در پیاده سازی آن ها با استفاده از این ساختمان داده ها زیاد شود که ما این را نمی خواهیم.

- ایده ای در این باره وجود دارد ، این گونه است که برای اندازه آرایه خود یه مقدار ثابت در نظر بگیریم و سپس هر کدام از شماره ها را در خانه ای از آرایه خود قرار دهیم.در حقیقت وقتی hash function را روی داده خود صدا میزنیم ، یک عدد به دست می آوریم.حال داده خود را در خانه ای از آرایه قرار می دهیم که شماره اش با باقی مانده عدد به دست آمده بر اندازه آرایه (که ثابت در نظر گرفتیم) برابر باشد.

- در این ایده ، باید تلاش کرد که تا حد امکان تعداد collision ها کم تر باشد.



collision ۴.۲۰

- تعریف: اگر دو تا داده یا key متفاوت داشته باشیم که hash شون باهم برابر باشد ، در واقع به collision برخورده ایم.باید سعی کنیم که طوری hash function را پیاده سازی کنیم که تا حد امکان تعداد collision ها کم تر باشد.

Map ۵.۲۰

- در map تابع های زیر را داریم:
  - HasKey(object) :چک می کند که آیا این کلید را داریم یا نه.در حقیقت تبدیل به هش کرده و سپس می رود به خانه مورد نظر و می بیند که آیا وجود دارد یا خیر.
  - Get(object) : value مناسب با کلید را به ما می دهد.
  - Set(object,value) :معادل کلید را مقدار value قرار می دهد.
- implementation
  - در hash table ما آرایه ای از Link List ها داریم.
  - به پیاده سازی های زیر لطفا توجه کنید.
- پیچیدگی محاسباتی hash table ، اگر m اندازه آرایه و n مجموع تمام چیزهایی که در آرایه است باشد ، n+m می باشد.

```
GetName(phoneNumber)
index ← ConvertToInt(phoneNumber)
return phoneBookArray[index]
```

```
SetName(phoneNumber, name)
index ← ConvertToInt(phoneNumber)
phoneBookArray[index] ← name
```

شکل ۱.۲۰: تابع های مورد نیاز در روش
direct addressing

```
HasKey(object)
chain ← Chains[hash(object)]
for (key, value) in chain:
  if key == object:
    return true
return false
```

شکل ۲.۲۰: HasKey

```
Get(object)
chain ← Chains[hash(object)]
for (key, value) in chain:
  if key == object:
    return value
return N/A
```

شکل ۳.۲۰: Get

۶.۲۰ Has Key

- می توان به جای hash table ، hash set داشت.ولی در این حالت ما فقط می خواهیم ببینیم که آیا key داده شده را داریم یا نه.

```
Set(object, value)
chain ← Chains[hash(object)]
for pair in chain:
  if pair.key == object:
    pair.value ← value
    return
chain.Append((object, value))
```

شکل ۴.۲۰: Set

- به پیاده سازی های زیر لطفا توجه کنید.

```
Add(object)
chain ← Chains[hash(object)]
for key in chain:
  if key == object:
    return
chain.Append(object)
```

شکل ۵.۲۰: Add

```
Remove(object)
if not Find(object):
  return
chain ← Chains[hash(object)]
chain.Erase(object)
```

شکل ۶.۲۰: Remove

```
Find(object)
chain ← Chains[hash(object)]
for key in chain:
  if key == object:
    return true
return false
```

شکل ۷.۲۰: Find

۷.۲۰ منابع بیش تر

- هش فانکشن چیست و چگونه یک هش فانکشن خوب انتخاب کنیم؟

https://www.geeksforgeeks.org/what-are-hash-functions-and-how-to-choose-a-good-hash-functio

- source reference

https://referencesource.microsoft.com/#q=hash