دانشکده مهندسی کامپیوتر

جزوه درس

ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۱۹

ادامه ی Hash + Disjoint sets Table

آرمین غلام پور - ۱۳۹۸/۹/۲

۱.۱۹ Disjoint sets

Disjoint sets : مجموعه هایی که اشتراکی با هم ندارند.
عملیات های موجود برای این ساختار داده :

- makeset : برای ساختن یک مجموعه ی جدید با یک عضو و با یک id مخصوص به همان مجموعه

```
MakeSet(x)
  smallest[i] <-- i
```

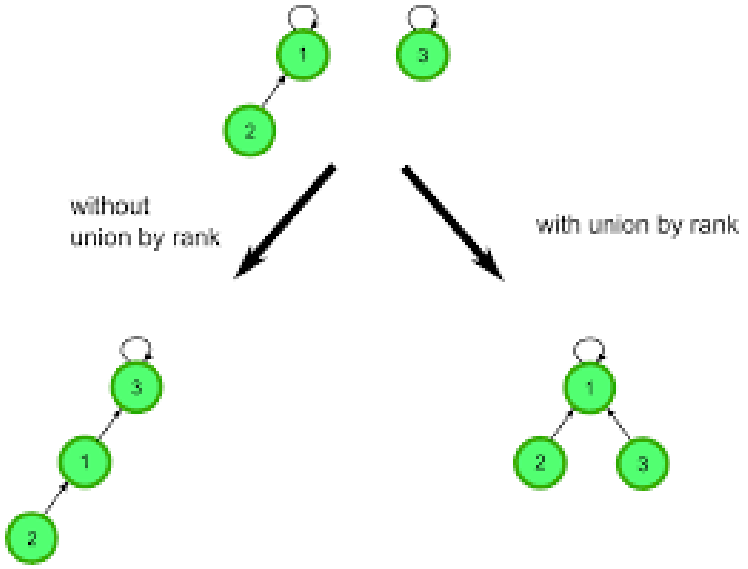
- find : پیدا کردن id یک عضو (پیدا کردن روت مجموعه ای که این عضو در آن قرار دارد)

```
Find(x)
  return smallest[i]
```

- union(i,j) : مجموعه های شامل دو عضو i و j را پیدا میکند. سپس id تمامی عضو های یک مجموعه را به id مجموعه دوم تغییر میدهد.

```
Union(i, j)
i.id <-- Find(i)
j.id <-- Find(j)
if i.id = j.id:
  return
m <-- min(i.id, j.id)
for k from 1 to n:
  if smallest[k] in <i.id, j.id>:
    smallest[k] <-- m
```

union by rank : به روشی از union کردن گفته میشود که اردر آن از حالت عادی بهتر است و مقدار آن O(logn) میشود. در این روش root مجموعه ی کوچکتر را بچه ی root مجموعه ی بزرگتر میکنیم.
در این الگوریتم اثبات میشود که ارتفاع درخت حداکثر logn میشود.(اثبات با استقرا و استفاده از اینکه درختی که ارتفاع k دارد حداکثر ۲ به توان k نود دارد)

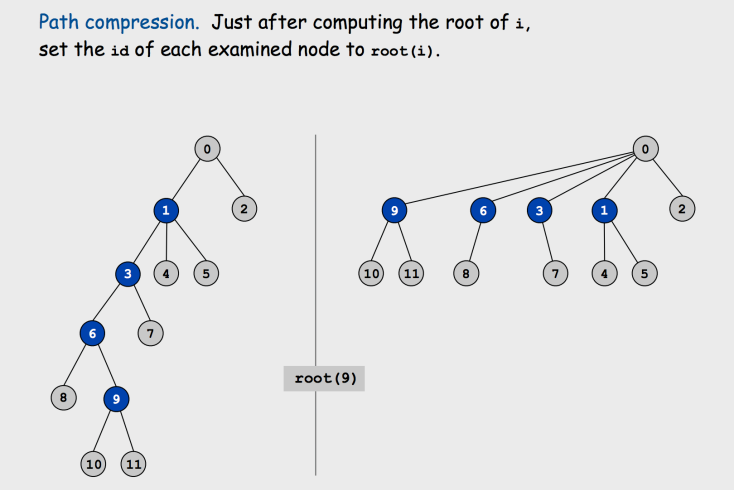


شکل ۱.۱۹: Union By Rank
[۱]

برای اینکه اردر عملیات ها کمتر شود از روش compression path استفاده میکنیم.
path compression : در این روش هر بار که متد find را صدا میزنیم، نود مورد نظر و تمامی نود هایی که در مسیر پیدا کردن root آن مجموعه هستند را، بچه ی root میکنیم. سر آخر آرایه ی parents را update میکنیم.
در این روش ارتفاع درخت ها کم و اردر عملیات نیز کمتر میشود. با این کار اردر عملیات ثابت (log*n)

میشود. (log*n برابر است با تعداد دفعاتی که باید از n ، log بگیریم تا به ۱ برسیم. این عدد تا ۲ به توان ۶۵۰۰۰ حداکثر برابر ۵ خواهد شد)

```
Find(i)
if i != parent[i]:
  parent[i] <-- Find(parent[i])
return parent[i]
```



شکل ۲.۱۹: Path Compression
[۲]

موارد استفاده disjoint sets [۳] :

- برای دسته بندی مولفه های همبندی یک گراف بدون جهت
- برای تشخیص دادن دور در گراف
- برای محاسبه ی minimun spanning tree در الگوریتم kruskal
- در تولید و یا حل مساله های شامل maze
- پیدا کردن دوست های مشترک در روابط اجتماعی

[۴] *visualizations for disjoint set + union by rank + path compression*

Hash Table ۲.۱۹

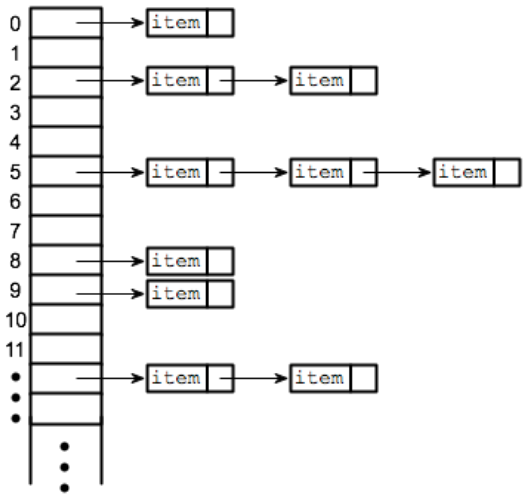
هر المان در یک hash table دارای دو ویژگی است :

- کلید
- مقدار

hash table یک ساختار داده ای است که داده ها را در غالب آرایه ای که خانه هایش بصورت لینک لیست هستند نگه داری میکند.

به اینصورت که برای هر داده به وسیله ی یک hash function یک عدد درست میکند که داده را هر چه که باشد (string, int, char, ...) به یک int یا long متناظر میکند.این عدد در واقع اندیس خانه ای از آرایه است که داده مورد نظر در لینک لیست موجود در آن خانه وجود دارد.

یک hash function خوب آن است که تعداد conflict ها در آن داده ها مینیمم باشد.
conflict : یعنی دو داده ی متفاوت دارای مقدار hash code یکسان باشند.این باعث میشود که در آن خانه از آرایه که اندیسش برابر عدد بدست آمده است، دو داده (یا بیشتر) ذخیره شود.
اگر به conflict بخوریم باید در لینک لیست مورد نظر داده های جدید را به طوری ذخیره کنیم که هر خانه از لینک لیست هم دارای مقدار و هم کلید داده های کانفلیکت خورده باشد تا بتوانیم بعدا تفاوت داده های آن لینک لیست را متوجه شویم.



شکل ۳.۱۹: Hash Table
[۵]

موارد استفاده ی hash table :

- blockchain
- file system
- digital signature
- phone book

[۶] *visualization for Hash Table*

Bibliography

[1] "union by rank visualisation." https://camo.githubusercontent.com/6e1fe8bd8ec9bec09c3f1c5fe91b9fef0dae7f72/68747470733a2f2f656e637279707465642d74626e302e67737461746963732e636f6d2f696d616765733f713d7

[2] "compression visualisation." https://bryanelidimas.files.wordpress.com/2016/02/compression.png.

[3] "disjoint sets use cases." https://www.oodlestechnologies.com/blogs/Understanding-Disjoint-Set-And-Their-Use-Cases-in-Computer-Science/.

[4] D. Galles, "Data structure visualizations." https://www.cs.usfca.edu/~galles/JavascriptVisual/DisjointSets.html.

[5] "hash table visualisation." https://he-s3.s3.amazonaws.com/media/uploads/0e2c706.png.

[6] D. Galles, "Hash table visualizations." https://www.cs.usfca.edu/~galles/visualization/OpenHash.html.