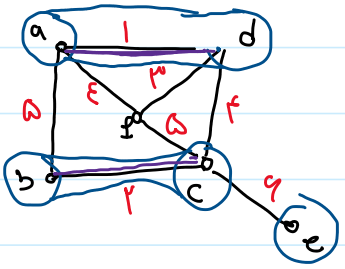
و جلسه بیست و پنجم

Tuesday, December 22, 2020 13:23

موضوع : مجموعه مجزا (disjoint_set).



ADT } union_find : داده ساختاری که ۳ عمل زیر را پشتیبانی کند :

۱. Make set(x) : یک مجموعه شامل عنصر x ایجاد می کند.

۲. find(x) : مجموعه ای که شامل عنصر x است را برمی گرداند

۳. union(x و y) : مجموعه شامل x و y را با هم ادغام می کند.

union(x,y) → X = find(x), Y = find(y), union(X,Y)

union(X و Y) : مجموعه X و Y را با هم ادغام می کند.*

۱- پیاده سازی با استفاده از آرایه : Quick find عناصر ۱ ... n

set (برچسب)

set[i] : مجموعه مربوط به عنصر i

در ابتدا، به ازای هر i ، set[i] = i .

```
int find(i){
  return(set[i])
}
```

```
union(int i, int j){
  x = set[i]
  y = set[j]
  for(l : 1 → n)
    if set[l] = y
      set[l] = x
}
```

زمان : find = O(1)

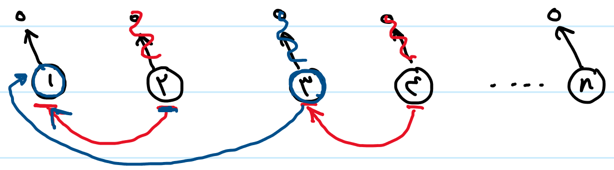
زمان : $find = O(1)$

$union = O(n)$

* ادغام وزن دار روش ۱ : $Find\ O(1)$

$union = O(n)$

۲ - پیاده سازی درختی: Quick union



اشاره‌گر Parent.

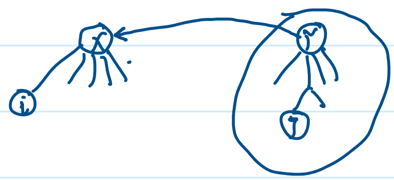
union(۱,۲)

union (۳,۴)

union(۳,۴) →

```
int Find(i){
    if (Parent[i]==0)
        return(i)
    return Find(Parent[i])
}
```

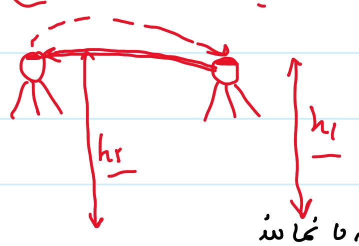
```
Union(i,j){
    X = find(i)
    Y = find(j)
    Parent[Y]=X   → if
}
```



زمان اجرا : $Find : O(n)$

$union : O(1)$ به شرط ادغام دو نماینده

* بهبود: ادغام وزن دار ! هر بار ریشه درخت با ارتفاع کمتر را فرزند ریشه درخت با ارتفاع بیشتر قرار بدید



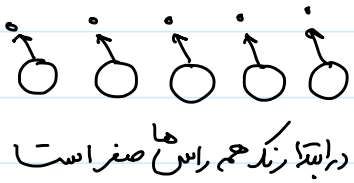
$h_1 \overset{?}{=} h_2$ (<, >)

تا ریشه یا نماینده

به هر راسی یک مقدار رتبه نسبت می‌دهیم

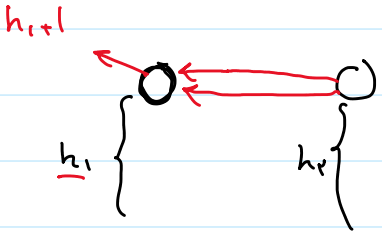
```
union (X, Y){
    if rank[X] > rank[Y]
```

نسبت‌های درهم



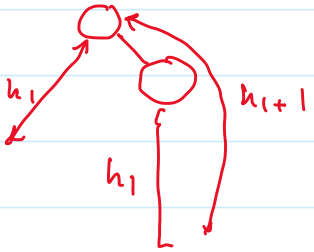
در ابتدا رنک همه راس‌ها صفر است

```
if rank[x] > rank[Y]
    Parent[Y] = x
else if rank[Y] > rank[x]
    Parent[x] = Y
else {
    Parent[Y] = x
    rank[x]++
}
}
```



$h_1 > h_2$

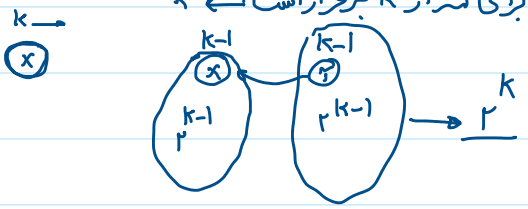
$h_1 = h_2$



** مشاهده ۵ : اگر تعداد رنک یک راس k باشد، زیردرخت شامل آن راس دارای حداقل 2^k راس است.

استقرا: $k=0$ ← ۱ راس

فرض برای کمتر از k برقرار است ← k



نتیجه: تعداد راس‌های با رنک k حداکثر $\frac{n}{2^k}$ است.

نتیجه ۲: ارتفاع درخت‌ها حداکثر $\log n$ است.

← زمان عمل Find(x)، $O(\log n)$ می‌شود...

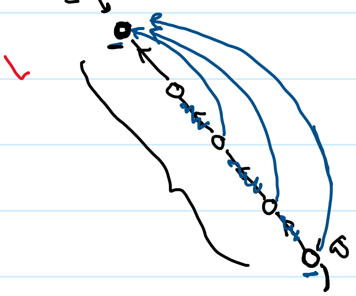
بهبود ۲:

```
find(x) {
    if parent[x] = 0
```

```
if parent[x] = 0
    return x
else {
    * Parent[x] = find(Parent[x])   **
    return Parent[x]
}
```

→ Parent[x] = Par...

ریشه



نتیجه بهبود: union : O(1) (دو نماینده)   find : $O(\alpha(n))$

$: O(\log^* n)$ ←

$x = 2^{2^{2^{\cdot^{\cdot}}}}$ (k) $\log^* x = k+1$

$\log^* 4 = 2$

$\log^* 16 = 3$

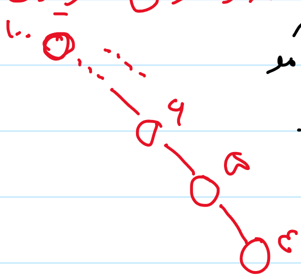
$\log^* 65536 = 4$

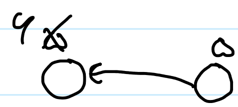
⋮

* هزینه سرشکن هر عمل find برابر با $O(\log^* n)$ است.

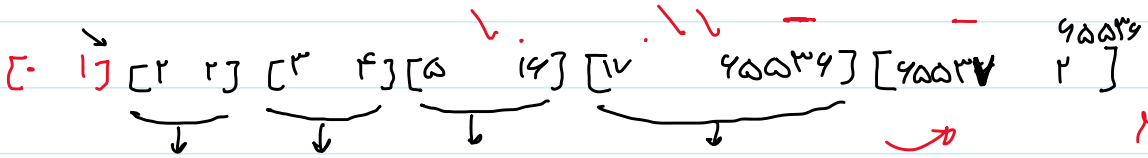
مشاهده ۳) با حرکت از هر راسی به سمت ریشه، مقدار رنک راس ها زیاد می شود.

مشاهده ۴) اگر زمانی راسی از ریشه بودن درآمد مقدار رنک آن دیگر عوض نمی شود.

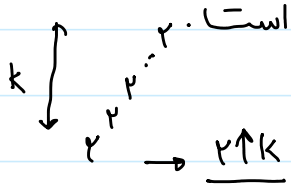




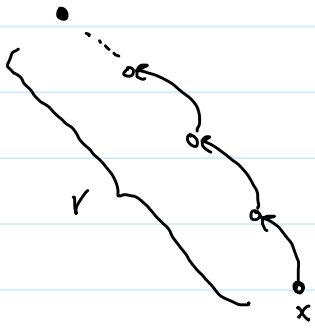
حال، راس ها را براساس مقدار رنکشان، در $\log^*$ گروه تخلف قرار می دهیم



بازه ها صورت $[2^k \; 2^{k+1}]$ است.



$2 \uparrow k$



find(x)

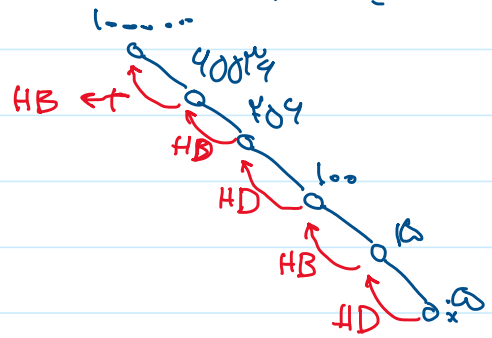
هزینه find : ۲

اگر ۲ بار بالا برویم، هزینه find برابر با ۲ است. این هزینه را ۲ قسمت می‌کنیم

HB ۱ : اگر راس v راس ریشه است، یا $Parent[v]$ در بازه دیگری نیست v است، یک واحد در HB می ریزیم.

هزینه بین گره‌ها!

HB ۲ : اگر راس v و پدرش در یک بازه بودند، ۱ واحد در HB می ریزیم.



بعد از n عملیات داخل HD و HB چقدر پول است؟

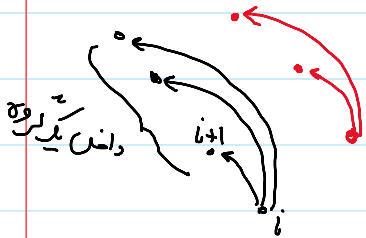
union و find

اول HB ؟ به ازای هر find ، حداکثر $\log^* n$ شارژی شود

بعد از n عملیات : $O(n \log^* n)$

دوم HD : حداکثر چند راس با رنک i داریم : $* \frac{n}{2^i}$

دوم ... حداکثر چند راس با رنک i داریم: $\frac{n}{2^i}$ *



$[k+1 \quad 2^k] \longrightarrow$ (عنصر با رنک i، حداکثر 2^k به HD اضافه می‌کند.

چند راس با رنک i = ($\frac{n}{2^i}$

$2\uparrow k \qquad 2^2$

$$\sum_{k=0}^{\log^*} \sum_{i=2\uparrow k+1}^{2\uparrow(k+1)} \frac{n}{2^i} \times 2\uparrow(k+1)$$

← کل جمع کمتر از n شد

$[(2\uparrow k)+1 \qquad 2\uparrow(k+1)]$

جمله اول $\frac{n \times 2\uparrow(k+1)}{2 \times 2^{2\uparrow k}} = \frac{n \times 2\uparrow(k+1)}{2 \times 2\uparrow(k+1)} = \frac{n}{2}$

$+$

جمله دوم $= \frac{n}{4}$

$+$

$\frac{n}{8}$

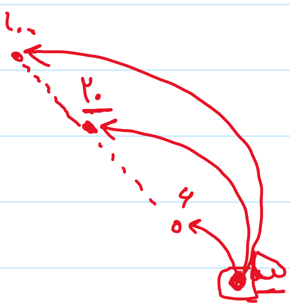
$\vdots$

$\wedge$

n

$$\sum_{k=0}^{\log^* n} n = n\log^* n$$

هزینه کل: $O(n\log^* n)$ ←



$[k+1 \qquad 2^k]$

$i \longrightarrow 2^k$. HD

□