داده‌گونه‌ی انتزاعی «پیدا–ادغام»

داده‌گونه‌های انتزاعی مفید و جالب برای ذخیره‌ی تعدادی مجموعه‌ی مجزا از عناصر که بر روی آن اعمال زیر انجام می‌شود:

- ایجاد مجموعه‌ها
- «پیداکردن یک عنصر» و
- «ادغام دو مجموعه»

نام‌های دیگر: Union-Find، Find-Merge یا Disjoint-Find-Merge

تعریف دقیق‌تر

فرض:

$U = \{۱, ۲, \ldots, n\}$ عناصر موجود

$A_۱$، $A_۲$ تا A_k $(k \leq n)$ مجموعه‌های مجزا $A_i \subseteq U$

که $\cup_{i=۱}^{k} A_i = U$.

اعمال

- ایجاد مجموعه $Create(i)$:
  $A_i = \{i\}$ را ایجاد می‌کند.
- پیدا کردن $Find(i)$:
  شماره‌ی j مجموعه‌ای که $i \in U$ عضو آن است $(i \in A_j)$ را محاسبه می‌کند.
- ادغام $Merge(i, j)$:
  $A_i \cup A_j$ را جایگزین A_i یا A_j می‌کند. (یکی از مجموعه‌های کم می‌شود)

کاربرد

- رده‌های هم‌ارزی در رابطه‌های هم‌ارزی
- اجزای هم‌بند در یک گراف
- اجزای قویاً هم‌بند
- الگوریتم کروسکال برای پیدا کردن درخت فراگیر کمینه

روش کلی برای به‌دست آوردن رده‌های هم‌ارزی

R یک رابطه‌ی هم‌ارزی بر روی عناصر $U = \{۱, ۲, \ldots, n\}$

۱) به‌ازای هر عنصر $i \in U$ مجموعه‌ی $A_i = \{i\}$ را ایجاد کن.

۲) برای هر عضو $(b, c) \in R$

۲-۱ b را پیداکن، فرض کن $b \in A_i$ ($i \leftarrow Find(b)$)

۲-۲ c را پیداکن، فرض کن $c \in A_j$ ($j \leftarrow Find(c)$)

۲-۳ اگر $i \neq j$، A_i و A_j را در هم ادغام کن ($Merge(i, j)$)

۳) هر مجموعه‌ی باقی‌مانده در انتها یک رده‌ی هم‌ارزی است.

تحلیل الگوریتم کلی

n بار عمل ایجاد مجموعه،

$۲|R|$ بار عمل پیداکردن و

حداکثر $n - ۱$ بار عمل ادغام

پیاده‌سازی‌ها مختلف

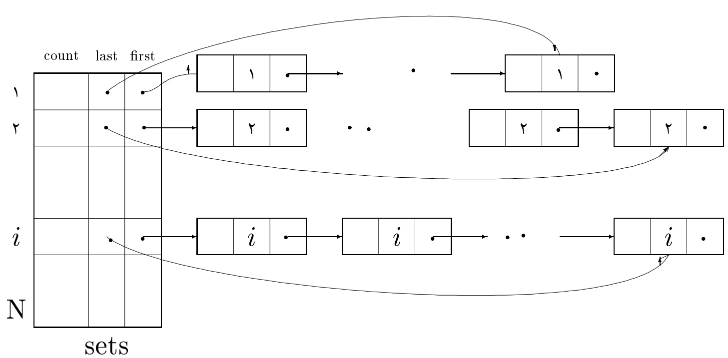
۱) ساده: هر یک از اعمال را در $O(n)$

۲) براساس لیست پیوندی: پیداکردن در $O(۱)$ و n عمل ادغام در مجموع در $O(n \lg n)$. (یعنی هزینه‌ی «سرشکن‌شده» (amortized) هر ادغام $O(\lg n)$ است.)

۳) مبتنی بر درخت: عمل ادغام $O(۱)$ و هر عمل پیدا کردن حداکثر در $O(\lg n)$.

۴) مبتنی بر درخت با فشرده‌سازی مسیر (path compression): m تا از این اعمال در زمان $O(m\alpha(m, n))$ انجام شود. $\alpha(m, n)$ عکس تابع «اکِرمَن» (Ackermann) است که بسیار کند رشد می‌کند و برای مقادیر بسیار بزرگ n، $\alpha(n, n) \leq ۴$.

داده‌ساختار مبتنی بر لیست



```
Const N=?;
      NULL=0;
type  cursor=0..N;
      set_numbers:1..N;
      elements_numbers:1..N;
      set_record = record
         first, last: cursor;
         count: 1..N;
      end;
      element_record = record
         set_no: set_numbers;
         next: cursor
      end;
var   sets: array[set_numbers] of set_record;
      elements: array[element_numbers] of element_record;
```


NITIALIZE()

```
for i ← 1 to lenght[elements]
    do first[S[i]] ← i
       last[S[i]] ← i
       size[S[i]] ← 1
       setNo[elements[i]] ← i
       next[elements[i]] ← null
```


FIND(x)

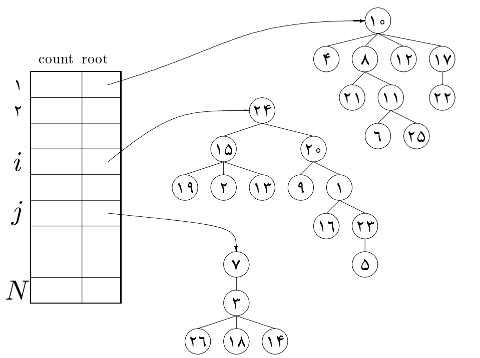
```
    ▷ finds the set number that x in an element of
return setNo[elements[x]]
```

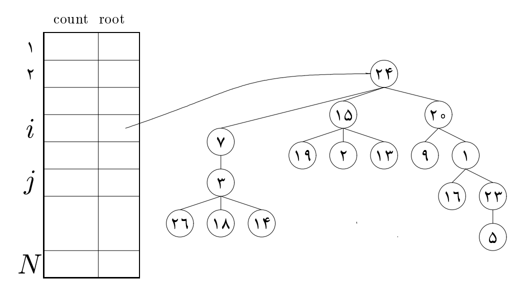

MERGE(i,j)

```
    ▷ S_x ← S_i ∪ S_j, x is either i or j
if size[S[i]] < size[S[j]]
  then r ← i
       k ← j
  else r ← j
       k ← i
p ← last[S[k]]
next[elements[p]] ← first[elements[r]]
size[S[k]] ← size[S[k]] + size[S[r]]
p ← next[elements[p]]
while p ≠ null
      do setNo[elements[p]] ← k
         p ← next[elements[p]]
```

لم ۱ هزینه‌ی n عمل «ادغام» در پیاده‌سازی فوق $O(n \lg n)$ است.

مبتنی بر درخت





ادغام مجموعه‌های i و j.

جزئیات پیاده‌سازی

```
onst N=?;
     NULL=0;
ype  cursor=0..N;
     set_numbers:1..N;
     elements_numbers:1..N;
     set_record = record
        root: element_numbers;
        height: integer;
     end;
     element_record = record
        set_no: set_numbers;
        father: cursor;
     end;
ar   sets:array[set_numners] of set_record;
     elements: array[element_numbers] of element_record;
```

INITIALIZE()

for $i \leftarrow 1$ **to** $length[elements]$
  do $height[S[i]] \leftarrow 0$
    $root[S[i]] \leftarrow i$
    $setNo[elements[i]] \leftarrow i$
    $parent[elements[i]] \leftarrow$ **null**

FIND(i)

  ▷ finds the set number that i in an element of
$p \leftarrow i$
while $parent[elements[p]] \neq$ **null**
    do $p \leftarrow parent[elements[p]]$
return $setNo[elements[p]]$

MERGE(i,j)

  ▷ $S_x \leftarrow S_i \cup S_j$
  ▷ x is either i or j
if $height[S[i]] < height[S[j]]$
  then $r \leftarrow i$
    $k \leftarrow j$
  else $r \leftarrow j$
    $k \leftarrow i$
$parent[elements[root[S[r]]]] \leftarrow root[S[k]]$
if $height[S[k]] = height[S[r]]$
  then $height[S[k]] \leftarrow height[S[k]] + 1$
$root[S[r]] \leftarrow$ **null**

لم ۲ هزینه‌ی هر عمل «پیداکردن» در پیاده‌سازی فوق $O(\lg n)$ است.

اثبات: با استقرا اثبات می‌کنیم که در یک درخت به ارتفاع h که با این الگوریتم ساخت می‌شود حداقل 2^h عنصر وجود دارد.

پایه‌ی استقرا: بدیهی.

فرض: دو درخت به‌ارتفاع‌های $h_1 \le h_2$ با هم ادغام می‌شوند.

- $h_1 < h_2$:

ارتفاع درخت حاصل برابر h_2.
با فرض استقراء، تعداد عناصر درخت حاصل از $2^{h_2} + 2^{h_1}$ و در نتیجه از 2^{h_2} کم‌تر نیست.

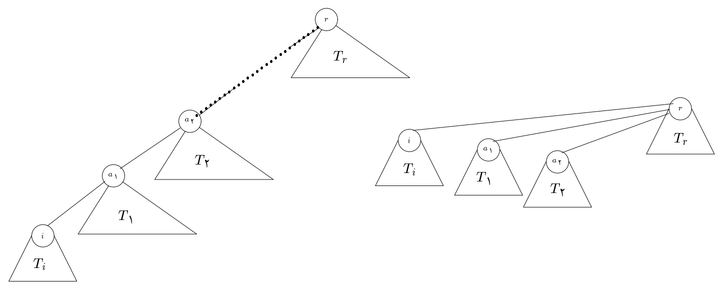
- $h_1 = h_2$:

ارتفاع درخت حاصل برابر $h_2 + 1$.
تعداد عناصر درخت حاصل از $2^{h_1} + 2^{h_2} = 2^{h_1+1}$ کم‌تر نیست.

$\Longleftarrow$ حداکثر ارتفاع یک درخت با m عنصر حداکثر $\lfloor \lg m \rfloor$ است.

$\Longleftarrow$ اثبات لم.

پیاده‌سازی با «فشرده‌سازی مسیر»



پس از $Find(i)$ درخت کوتاه می‌شود.

تابع اکرمن و عکس آن

برای فهم به‌تر تابع اکرمن (Ackermann's function) و عکس آن نماد زیر را برای تکرار توان ۲ تعریف می‌کنیم.

$$g(i) \equiv \left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\}i$$

این تابع به‌صورت بازگشتی زیر نیز تعریف می‌شود:

$$g(i) = \begin{cases} 2^1 & \text{if } i = 0, \\ 2^2 & \text{if } i = 1, \\ 2^{g(i-1)} & \text{if} i > 1. \end{cases}$$

مثلاً

$$g(4) = \left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\}4 = 2^{2^{2^{2}}} = 2^{65536}.$$

هم‌چنین برای تعریف $\lg^*$ داریم:

$$\lg^{(i)} n = \begin{cases} n & \text{if } i = 0, \\ \lg(\lg^{(i-1)} n) & \text{if } i > 0 \text{ and } \lg^{(i-1)} n) > 0, \\ \text{undefined} & \text{if } i > 0 \text{ and } \lg^{(i-1)} n) \le 0 \text{ or } \lg^{(i-1)} n) \text{ is undefined} \end{cases}$$

و

$$\lg^* n = \min\{i \ge 0 : \lg^{(i)} n \le 1\}$$

حال تابع اکرمن را برای مقادیر $i,j \geq ۱$ به‌صورت زیر تعریف می‌کنیم:

$$\begin{aligned} A(۱,j) &= ۲^j && \text{for } j \geq ۱, \\ A(i,۱) &= A(i-۱,۲) && \text{for } i \geq ۲, \\ A(i,j) &= A(i-۱, A(i,j-۱)) && \text{for } i,j \geq ۲. \end{aligned}$$

مقادیر تابع اکرمن $A(i,j)$ برای مقادی کوچک i و j

	$j=۱$	$j=۲$	$j=۳$	$j=۴$
$i=۱$	$۲^۱$	$۲^۲$	$۲^۳$	$۲^۴$
$i=۲$	$۲^۲$	$۲^{۲^۲}$	$۲^{۲^{۲^۲}}$	$۲^{۲^{۲^{۲^۲}}}$
$i=۳$	$۲^{۲^۲}$	$\left.۲^{۲^{\cdot^{\cdot^{\cdot^{۲}}}}}\right\}۱۶$	$\left.۲^{۲^{\cdot^{\cdot^{\cdot^{۲}}}}}\right\}\left.۲^{۲^{\cdot^{\cdot^{\cdot^{۲}}}}}\right\}۱۶$	$\left.۲^{۲^{\cdot^{\cdot^{\cdot^{۲}}}}}\right\}\left.۲^{۲^{\cdot^{\cdot^{\cdot^{۲}}}}}\right\}\left.۲^{۲^{\cdot^{\cdot^{\cdot^{۲}}}}}\right\}۱۶$

مقادیر تابع اکرمن $A(i,j)$ برای مقادی کوچک i و j.

تعریف عکس تابع اکرمن:

$$\alpha(m,n) = \min\{i \geq ۱ : A(i, \lfloor m/n \rfloor) > \lg n\}.$$

نشان می‌دهیم که برای ارقام واقعی $\alpha(m,n) \leq ۴$.

چون $m \geq n$ مقدار $\lfloor m/n \rfloor$ حداقل ۱ است.

چون تابع اکرمن اکیداً صعودی است، به‌ازای $\lfloor m/n \rfloor \geq ۱$ داریـ
$A(i, \lfloor m/n \rfloor) > A(i,۱)$.

بنابراین $A(۴, \lfloor m/n \rfloor) \geq A(۴,۱)$.

ولی داریم

$$A(۴,۱) = A(۳,۲) = \left.۲^{۲^{\cdot^{\cdot^{\cdot^{۲}}}}}\right\}۱۶$$

که بسیار بیش‌تر از تخمین تمامی اتم‌ها در جهان (حدود $۱۰^{۸۰}$) است. فقط برای مقادی
غیر عملی بزرگ n ممکن است $A(۴,۱) \leq \lg n$.

$\Longleftarrow$ **برای همه‌ی مقادیر عملی** $\alpha(m,n) \leq ۴$.

۱) S مجموعه‌ای از اعداد حقیقی بین صفر و n است: $S = \{x | 0 \leq x \leq n\}$
هم‌چنین I_j برای $j = 0 \cdots n-1$ زیرمجموعه هائی از S هستند به‌طوری‌که
$I_j = \{x | j \leq x < j+1\}$ و $S = \cap_{j=0}^{n-1} I_j$. یک داده‌ساختار برای ذخیره اعداد مجموعه
S (و در نتیجه I_j ها) پیشنهاد کنید به‌طوری‌که بتوان اعمال زیر را با مرتبه های خواسته
شده انجام داد.

۱) $\text{Insert}(x)$: در $O(log(|S|)$ x را به S و نیز به I_j مربوطه اضافه کند (در صورتی که
قبلا وجود نداشته باشد .)

۲) $\text{Delete}\ (x)$: در $O(log(|S|)$ x را از S و نیز از I_j حذف کند.

۳) $\text{List}\ (j)$: عناصر موجود در I_j را در $O(|I_j|)$ بنویسید.