计算机问题求解 — 论题3-3

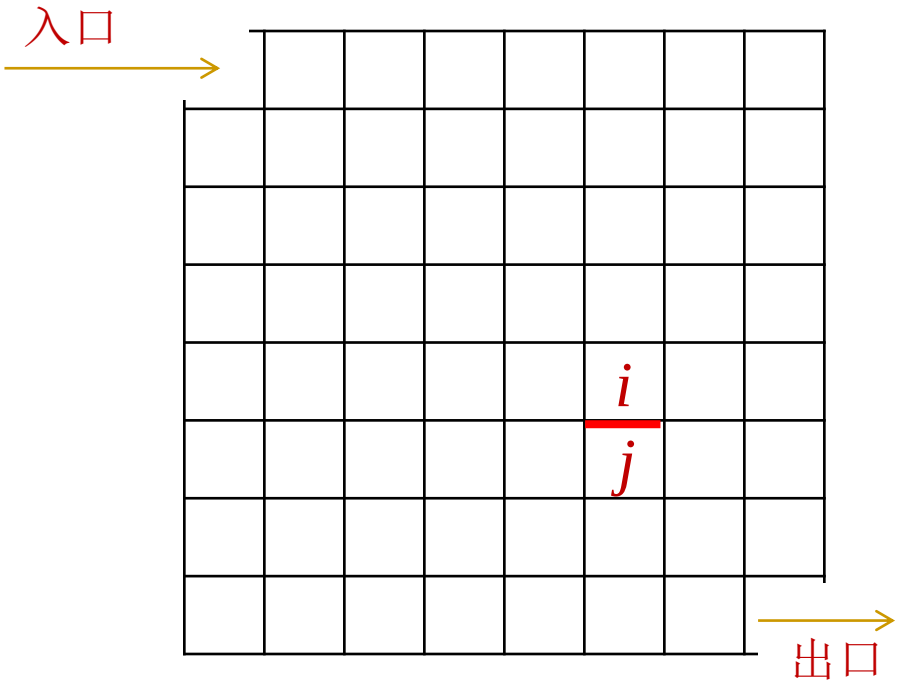
-用于动态等价关系的数据结构

2016年09月14日

问题1:

什么是等价关系?它有什么应用意义?

创建“迷宫”



问题2：

你能否基于动态等价关系的概念来解释这个生成过程？

问题3:

我们用什么数据结构实现动态等价关系？这个数据结构的基本操作是什么？

MAKE-SET(x) creates a new set whose only member (and thus representative) is x. Since the sets are disjoint, we require that x not already be in some other set.

UNION(x, y) unites the dynamic sets that contain x and y, say S_x and S_y, into a new set that is the union of these two sets. We assume that the two sets are disjoint prior to the operation. The representative of the resulting set is any member of $S_x \cup S_y$, although many implementations of UNION specifically choose the representative of either S_x or S_y as the new representative. Since we require the sets in the collection to be disjoint, conceptually we destroy sets S_x and S_y, removing them from the collection $\mathcal{S}$. In practice, we often absorb the elements of one of the sets into the other set.

FIND-SET(x) returns a pointer to the representative of the (unique) set containing x.

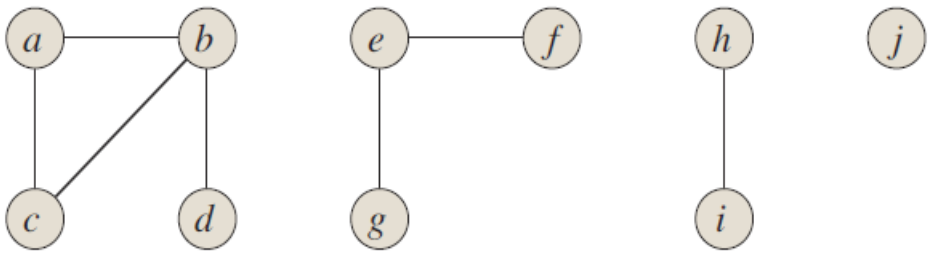
问题4:

什么是representative?

讨论数学与讨论数据结构时它有什么差别?

问题5:

我们讨论的不是一个算法，而是一个数据结构，那所谓“时间复杂性分析”究竟是什么意思呢？



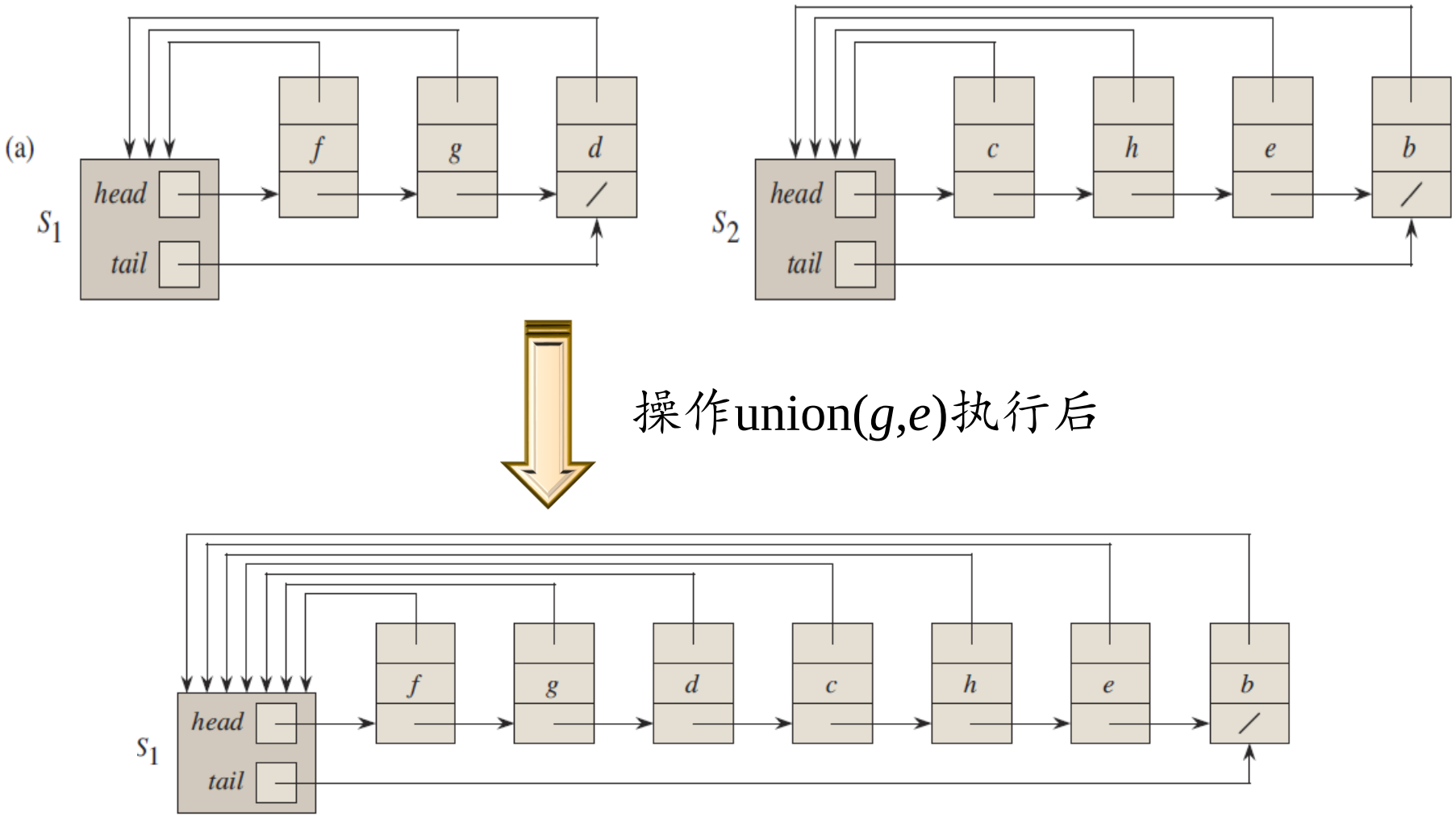

```
CONNECTED-COMPONENTS(G)
1  for each vertex v ∈ G.V
2      MAKE-SET(v)
3  for each edge (u, v) ∈ G.E
4      if FIND-SET(u) ≠ FIND-SET(v)
5          UNION(u, v)
```

```
SAME-COMPONENT(u, v)
1  if FIND-SET(u) == FIND-SET(v)
2      return TRUE
3  else return FALSE
```

Edge processed	Collection of disjoint sets									
initial sets	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$	$\{g\}$	$\{h\}$	$\{i\}$	$\{j\}$
(b,d)	$\{a\}$	$\{b,d\}$	$\{c\}$		$\{e\}$	$\{f\}$	$\{g\}$	$\{h\}$	$\{i\}$	$\{j\}$
(e,g)	$\{a\}$	$\{b,d\}$	$\{c\}$		$\{e,g\}$	$\{f\}$		$\{h\}$	$\{i\}$	$\{j\}$
(a,c)	$\{a,c\}$	$\{b,d\}$			$\{e,g\}$	$\{f\}$		$\{h\}$	$\{i\}$	$\{j\}$
(h,i)	$\{a,c\}$	$\{b,d\}$			$\{e,g\}$	$\{f\}$		$\{h,i\}$		$\{j\}$
(a,b)	$\{a,b,c,d\}$				$\{e,g\}$	$\{f\}$		$\{h,i\}$		$\{j\}$
(e,f)	$\{a,b,c,d\}$				$\{e,f,g\}$			$\{h,i\}$		$\{j\}$
(b,c)	$\{a,b,c,d\}$				$\{e,f,g\}$			$\{h,i\}$		$\{j\}$

问题6：

假如我们想知道原来未直接相连的两个顶点一旦连起来就会形成回路，应该如何解决？


(a)
S_1
head
tail
f
g
d
S_2
head
tail
c
h
e
b
操作union(g,e)执行后
S_1
head
tail
f
g
d
c
h
e
b

问题7：

为什么用链表实现，每个操作的平均代价可能会是线性的？

Union操作对象的次序不影响结果，
却影响效率，这对你有什么启发？

问题8：

为什么weighted-union能降低操作平均代价？

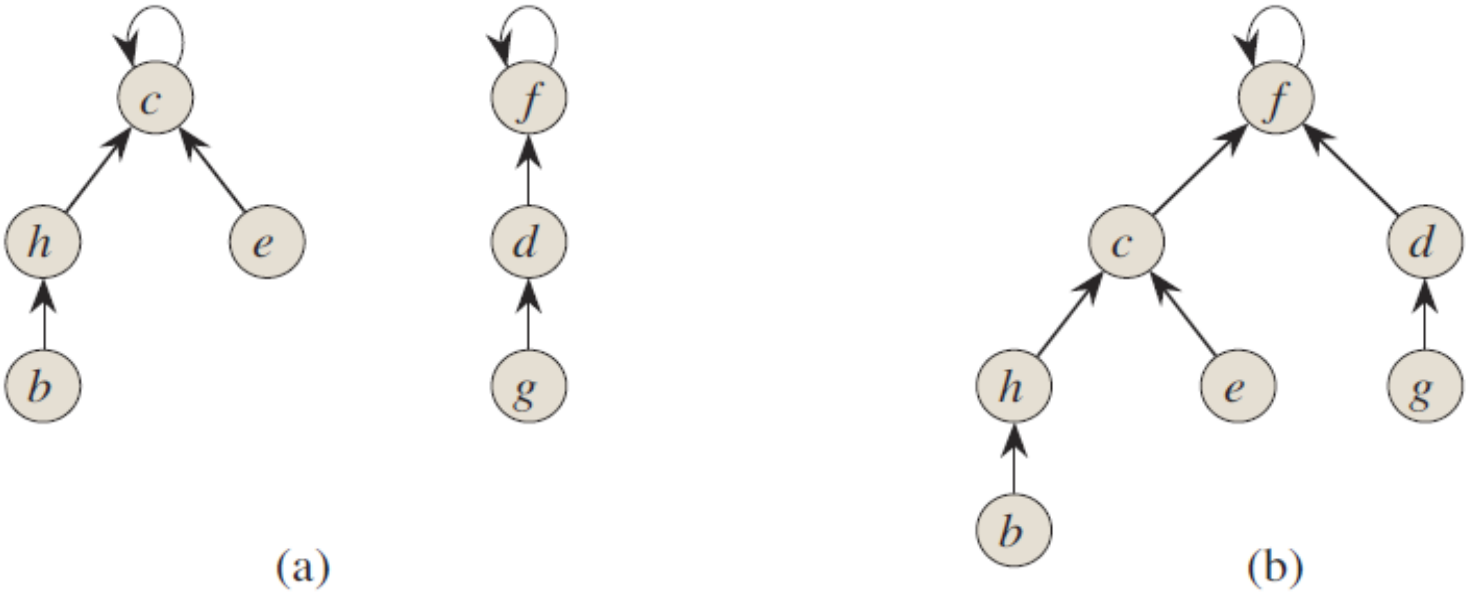
Theorem 21.1

Using the linked-list representation of disjoint sets and the weighted-union heuristic, a sequence of m MAKE-SET, UNION, and FIND-SET operations, n of which are MAKE-SET operations, takes $O(m + n \lg n)$ time.

问题9：

你能否说出此定理证明最核心的思想？

ADT inTree和Disjoint-set Forest



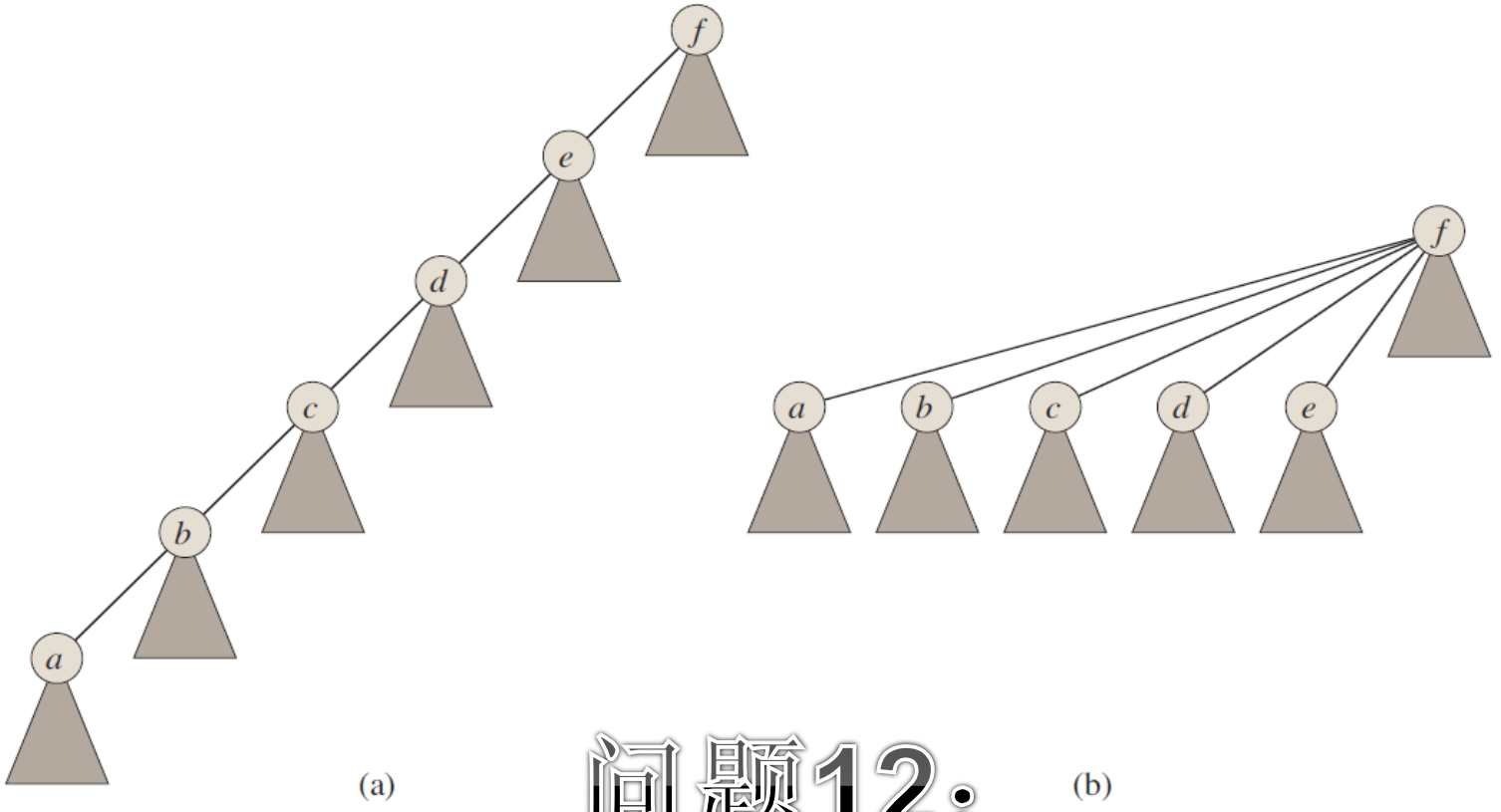
问题10:

这些树和前面介绍的搜索树有什么不同？你认为不同的意义在哪里？

问题11：

你觉得disjoint-set forest中的树结构性质中哪些只与操作代价有关，却与操作结果无关？这对你有什么启示？

顺手牵“羊”



问题12：

好处在哪里？

注意：
rank 如何定义与修改。

```
MAKE-SET(x)
1  x.p = x
2  x.rank = 0
```

问题13：

路径“压缩”在何处实现？

```
UNION(x, y)
1  LINK(FIND-SET(x), FIND-SET(y))
```

```
LINK(x, y)
1  if x.rank > y.rank
2       y.p = x
3  else x.p = y
4       if x.rank == y.rank
5            y.rank = y.rank + 1
```

```
FIND-SET(x)
1  if x ≠ x.p
2       x.p = FIND-SET(x.p)
3  return x.p
```

rank, which is an upper bound on the height of the node

效果显著

When we use both union by rank and path compression, the worst-case running time is $O(m\,\alpha(n))$, where $\alpha(n)$ is a *very* slowly growing function, which we define in Section 21.4. In any conceivable application of a disjoint-set data structure, $\alpha(n) \leq 4$; thus, we can view the running time as linear in m in all practical situations. Strictly speaking, however, it is superlinear.

问题14:

什么意思?

证明的基本思路

问题15：

为什么可以这样做？

- 用“MakeSet-Link-Findset序列”的分析代替对“MakeSet-Union-FindSet序列”的分析；
- 整个序列的操作代价是三种操作代价之和，所以可以分别分析其单个操作平均代价；
- 其中单个操作代价可能很大的是FindSet（如果操作对象到其所属的树根距离很大），但由于采用path compression，大代价FindSet出现的机会很有限，其条件是有多个Link操作导致的。因此，对于这个操作序列采用分摊分析法；
- 这里使用基于rank的the potential method。

问题16：

你能说说Rank最重要的性质吗？从何时开始它不会再变化了？

严格递增且有界。

一个增长极快的函数与其“逆”

For integers $k \geq 0$ and $j \geq 1$, we define the function $A_k(j)$ as

$$A_k(j) = \begin{cases} j+1 & \text{if } k = 0\,, \\ A_{k-1}^{(j+1)}(j) & \text{if } k \geq 1\,, \end{cases}$$

$$A_4(1) \gg 10^{80}$$

We define the inverse of the function $A_k(n)$, for integer $n \geq 0$, by

$$\alpha(n) = \min\{k : A_k(1) \geq n\}\ .$$

$$\alpha(n) = \begin{cases} 0 & \text{for } 0 \leq n \leq 2\,, \\ 1 & \text{for } n = 3\,, \\ 2 & \text{for } 4 \leq n \leq 7\,, \\ 3 & \text{for } 8 \leq n \leq 2047\,, \\ 4 & \text{for } 2048 \leq n \leq A_4(1) \end{cases}$$

势函数

$$\phi_q(x) = \begin{cases} \alpha(n) \cdot x.rank & \text{if } x \text{ is a root or } x.rank = 0, \\ (\alpha(n) - \underline{\text{level}(x)}) \cdot x.rank - \underline{\text{iter}(x)} & \text{if } x \text{ is not a root and } x.rank \geq 1 \end{cases}$$

For every node x, and for all operation counts q, we have

$$0 \leq \phi_q(x) \leq \alpha(n) \cdot x.rank .$$

$$\text{level}(x) = \max\{k : x.p.rank \geq A_k(x.rank)\}$$

$$0 \leq \text{level}(x) < \alpha(n)$$

$$\text{iter}(x) = \max\left\{i : x.p.rank \geq A^{(i)}_{\text{level}(x)}(x.rank)\right\}$$

$$1 \leq \text{iter}(x) \leq x.rank$$

非根结点的势不会增加

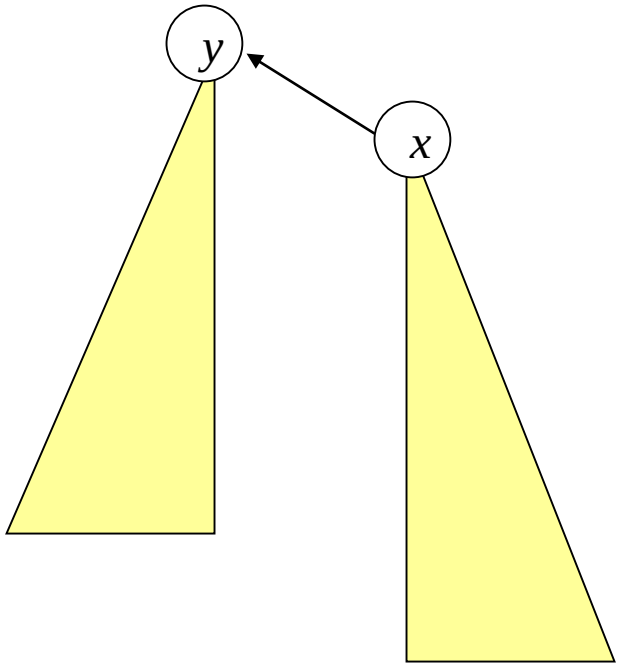
Let x be a node that is not a root, and suppose that the qth operation is either a LINK or FIND-SET. Then after the qth operation, $\phi_q(x) \le \phi_{q-1}(x)$. Moreover, if $x.rank \ge 1$ and either level(x) or iter(x) changes due to the qth operation, then $\phi_q(x) \le \phi_{q-1}(x) - 1$. That is, x's potential cannot increase, and if it has positive rank and either level(x) or iter(x) changes, then x's potential drops by at least 1.

在势函数定义式的第二个式子中，$x.rank$和$\alpha(n)$均不会变，所以势的变化取决于level(x)和iter(x)的变化。

问题17:

一次Link操作后，势最多增加$\alpha(n)$，为什么？

只有新的根势会增加



- From the definition of $\phi_q(x)$, we see that, since x was a root just before the qth operation, $\phi_{q-1}(x) = \alpha(n) \cdot x.rank$. If $x.rank = 0$, then $\phi_q(x) = \phi_{q-1}(x) = 0$. Otherwise,

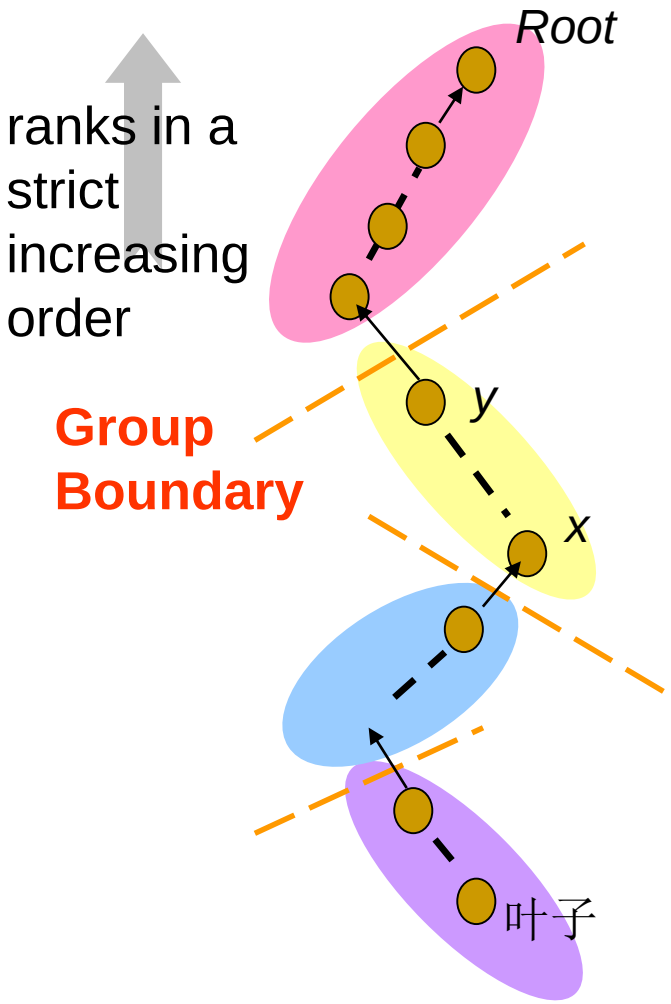
$$\begin{aligned} \phi_q(x) &< \alpha(n) \cdot x.rank \quad \text{(by Corollary 21.9)} \\ &= \phi_{q-1}(x) , \end{aligned}$$

and so x's potential decreases.

- Because y is a root prior to the LINK, $\phi_{q-1}(y) = \alpha(n) \cdot y.rank$. The LINK operation leaves y as a root, and it either leaves y's rank alone or it increases y's rank by 1. Therefore, either $\phi_q(y) = \phi_{q-1}(y)$ or $\phi_q(y) = \phi_{q-1}(y) + \alpha(n)$.

问题18:

一次Find-Set操作后，至少有$\max(0, s-(\alpha(n)+2))$个点势会下降，为什么？



$$
\begin{aligned}
x.p.rank &\geq A_k^{(\mathrm{iter}(x))}(x.rank) \\
y.p.rank &\geq A_k(y.rank) \\
y.rank &\geq x.p.rank
\end{aligned}
$$

$$
\begin{aligned}
y.p.rank &\geq A_k(y.rank) \\
&\geq A_k(x.p.rank) \\
&\geq A_k(A_k^{(\mathrm{iter}(x))}(x.rank)) \\
&= A_k^{(i+1)}(x.rank)\ .
\end{aligned}
$$

各操作的Amortized Cost

Lemma 21.11

The amortized cost of each MAKE-SET operation is $O(1)$

Lemma 21.12

The amortized cost of each LINK operation is $O(\alpha(n))$

增势，最多为1。

Lemma 21.13

The amortized cost of each FIND-SET operation is $O(\alpha(n))$.

减势。与通路长度相关。

Open topics

- A:
  - 随机生成n(n<=20,000)个（互不相同）11位电话号码以及这些电话之间的m(m<=20,000,000)次通话记录：格式如下：
    - 第1行，两个整数n，m；
    - 第2~n+1行，每行一个11位电话号码
    - 第n+2~n+m+1行，每行格式如下：
      - from to time
      - from与to表示两个不同的电话号码
      - time 为通话时长（单位：秒），10<=time<=1200
- B：利用并查集解决
  - 累计通话时间超过10分钟被认定为密友；
  - 密友的密友仍为密友；
  - 一个密友圈总累计通话时间最长的为“首脑”；
  - 按照电话号码的字典序，逐行输出每一个首脑及其在集团内累计通话时间，每行格式如下：
    - head total_time #member
    - 其中，head：首脑号码，total_time：圈内累计通话时间, #member:朋友圈成員數量

课外作业

- TC pp.564-: ex.21.1-2, 21.1-3
- TC pp.567-: ex.21.2-1, 21.2-3, 21.2-6
- TC pp.572-: ex.21.3-1, 21.3-2, 21.3-3
- TC pp.582-: prob.21-1