

- 书面作业讲解
  - TC第7.1节练习2
  - TC第7.2节练习4
  - TC第7.3节练习2
  - TC第7.4节练习2
  - TC第7章问题4、5
  - TC第8.1节练习3、4
  - TC第8.2节练习4
  - TC第8.3节练习4
  - TC第8.4节练习2
  - TC第8章问题2
  - TC第9.1节练习1
  - TC第9.3节练习5、7

# TC第7.1节练习2

- Modify PARTITION so that  $\lfloor q=(p+r)/2 \rfloor$  when all elements in the array  $A[p..r]$  have the same value.
  - 方法1: if ( $A[j]==x$ ) count++;
  - 方法2: if ( $A[j]==x$ ) flag=true;
  - 方法3: if ( $A[p]==A[r]$ ) return  $\lfloor q=(p+r)/2 \rfloor$ ; 行不行?

## TC第7.3节练习2

- How many calls are made to RANDOM?
  - Worst case:  $\Theta(n)$
  - Best case:  $T(n)=T(\lfloor n/2 \rfloor)+T(\lceil n/2 \rceil)+1$ ,  $T(n)=\Theta(n)$

# TC第7.4节练习2

- 再次强调：要用数学归纳法严格证明，不能只用递归树来估计。这是态度问题！
- $T(n)=2T(n/2)+\Theta(n)$ ?
  - 教材P180,  $T(n)=\min(\dots)+\Theta(n)$

# TC第7章问题4

- (a) 如何严格证明？
  - 数学归纳法
  - loop invariant
- (b) Stack depth is  $\Theta(n)$ .
  - 单调增
  - 单调减行不行？
- (c) the worst-case stack depth is  $\Theta(\lg n)$ .
  - 找中位数作为pivot，行不行？
  - 在小半区间上递归，在大半区间上尾递归

# TC第8.1节练习4

- Hint: It is not rigorous to simply combine the lower bounds for the individual subsequences.
- $2^h \geq (k!)^{n/k}$

# TC第8.2节练习4

- return  $C[b]-C[a-1]$ , 有没有问题?
- if ( $a>0$ ) return  $C[b]-C[a-1]$
- else return  $C[b]$

# TC第8章问题2

- (b) Give an algorithm that satisfies criteria 1 and 3 above.
  - counting sort行不行?
  - bucket sort行不行?
  - 注意：输入是an array of data records，只能swap，不能简单置0/1
  - 类似quicksort的partition (pivot=0)
- (e) How to modify counting sort so that it sorts the records in place in  $O(n+k)$  time?
  - `CC = arraycopy(C);`
  - `for (j=A.length; j>=1; j--)`
  - `while (j != C[A[j]])`
  - `if (C[A[j]]<j<=CC[A[j]]) {`
  - `break;`
  - `} else {`
  - `swap (A[C[A[j]]], A[j]);`
  - `C[A[j]]--;`
  - `}`

	1	2	3	4	5	6	7	8
A	2	5	3	0	2	3	0	3
	0	1	2	3	4	5		
C	2	2	4	7	7	8		



# TC第9.3节练习7

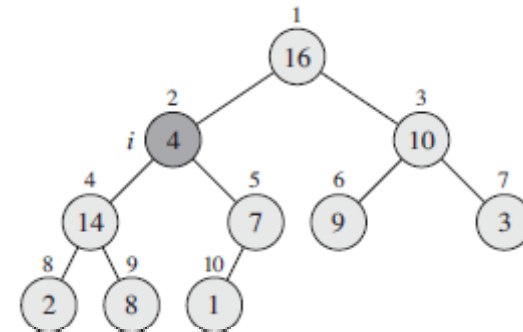
- $O(n)$ -time algorithm determines the  $k$  numbers in  $S$  that are closest to the median of  $S$ .
  1. 找中位数  $O(n)$
  2. 每个数减去中位数、取绝对值  $O(n)$
  3. 选 $k$ 次最小值  $O(kn)=O(n)$ ，行不行？
  4. 选第 $k$ 小的值  $O(n)$
  5. 选所有比它小的值  $O(n)$

- 教材答疑和讨论
  - TC第6章
  - SB第2章

# 问题1: heap和heapsort

MAX-HEAPIFY( $A, i$ )

```
1  $l = \text{LEFT}(i)$ 
2  $r = \text{RIGHT}(i)$ 
3 if  $l \leq A.\text{heap-size}$  and  $A[l] > A[i]$ 
4    $\text{largest} = l$ 
5 else  $\text{largest} = i$ 
6 if  $r \leq A.\text{heap-size}$  and  $A[r] > A[\text{largest}]$ 
7    $\text{largest} = r$ 
8 if  $\text{largest} \neq i$ 
9   exchange  $A[i]$  with  $A[\text{largest}]$ 
10  MAX-HEAPIFY( $A, \text{largest}$ )
```



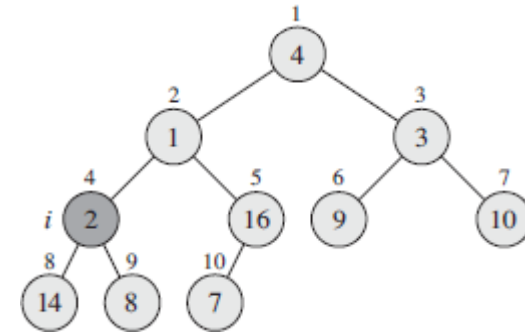
- 这个算法的作用是什么？
- 你能简要概括它的基本原理吗？
- 如何证明它的正确性？
- 它的运行时间是多少？

# 问题1: heap和heapsort (续)

BUILD-MAX-HEAP( $A$ )

```
1  $A.heap-size = A.length$   
2 for  $i = \lfloor A.length/2 \rfloor$  downto 1  
3   MAX-HEAPIFY( $A, i$ )
```

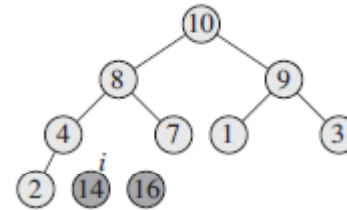
- 这个算法的作用是什么?
- 你能简要概括它的基本原理吗?
- 如何证明它的正确性?
- 它的运行时间是多少?



# 问题1: heap和heapsort (续)

HEAPSORT( $A$ )

```
1 BUILD-MAX-HEAP( $A$ )
2 for  $i = A.length$  downto 2
3   exchange  $A[1]$  with  $A[i]$ 
4    $A.heap-size = A.heap-size - 1$ 
5   MAX-HEAPIFY( $A, 1$ )
```



- 这个算法的作用是什么?
- 你能简要概括它的基本原理吗?
- 如何证明它的正确性?
- 它的运行时间是多少?

## 问题2: priority queue

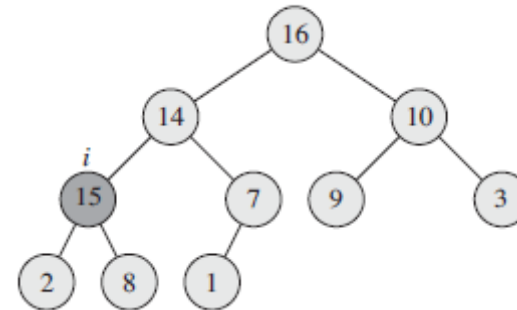
```
HEAP-EXTRACT-MAX(A)
1  if A.heap-size < 1
2      error "heap underflow"
3  max = A[1]
4  A[1] = A[A.heap-size]
5  A.heap-size = A.heap-size - 1
6  MAX-HEAPIFY(A, 1)
7  return max
```

- 这个算法的作用是什么?
- 你能简要概括它的基本原理吗?
- 如何证明它的正确性?
- 它的运行时间是多少?

## 问题2: priority queue (续)

HEAP-INCREASE-KEY( $A, i, key$ )

```
1  if  $key < A[i]$ 
2      error "new key is smaller than current key"
3   $A[i] = key$ 
4  while  $i > 1$  and  $A[\text{PARENT}(i)] < A[i]$ 
5      exchange  $A[i]$  with  $A[\text{PARENT}(i)]$ 
6       $i = \text{PARENT}(i)$ 
```



- 这个算法的作用是什么?
- 你能简要概括它的基本原理吗?
- 如何证明它的正确性?
- 它的运行时间是多少?

## 问题2: priority queue (续)

MAX-HEAP-INSERT( $A, key$ )

1  $A.heap-size = A.heap-size + 1$

2  $A[A.heap-size] = -\infty$

3 HEAP-INCREASE-KEY( $A, A.heap-size, key$ )

- 这个算法的作用是什么?
- 你能简要概括它的基本原理吗?
- 如何证明它的正确性?
- 它的运行时间是多少?

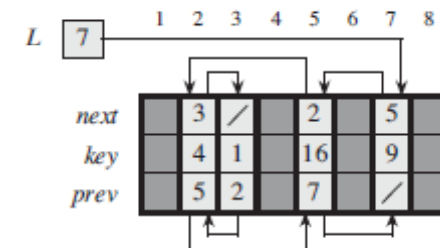
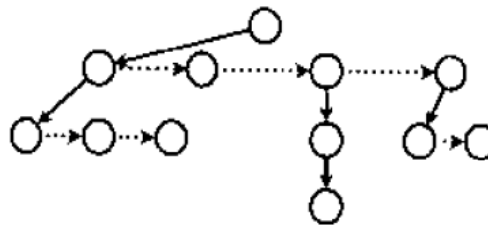
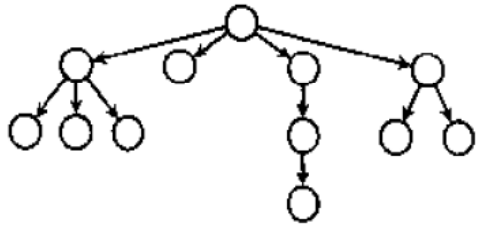


## 问题2: priority queue (续)

- priority queue  $\leftarrow$  heap  $\leftarrow$  array
- 结合这个例子, 谈谈你对ADT的理解
- 这样做有什么优缺点?
  - 正确性分析
  - 性能分析

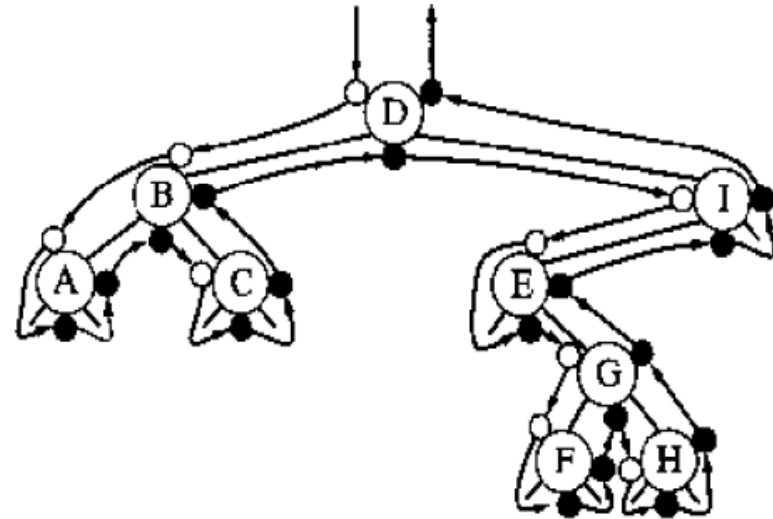
# 问题3: tree

- 你能谈谈rooted tree是如何分层抽象的吗?



# 问题3: tree (续)

```
void traverse(BinTree T)
  if (T is not empty)
    Preorder-process root(T);
    traverse(leftSubtree(T));
    Inorder-process root(T);
    traverse(rightSubtree(T));
    Postorder-process root(T);
  return;
```



- 你是怎么理解preorder/inorder/postorder的?
- 它们遍历的顺序分别是什么?

Preorder (white dots): D B A C I E G F H  
Inorder (gray dots): A B C D E F G H I  
Postorder (black dots): A C B F H G E I D

# 问题4： 其它ADT

- (linked) list
- (binary) tree
- stack
- queue
- heap
- priority queue
- union-find
- dictionary
- ...

# 问题4：其它ADT (续)

- union-find

`UnionFind create(int n)`

*Precondition:* none.

*Postconditions:* If `sets = create(n)`, then `sets` refers to a newly created object; `find(sets, e) = e` for  $1 \leq e \leq n$ , and is undefined for other values of  $e$ .

`int find(UnionFind sets, e)`

*Precondition:* Set  $\{e\}$  has been created in the past, either by `makeSet(sets, e)` or `create`.

`void makeSet(UnionFind sets, int e)`

*Precondition:* `find(sets, e)` is undefined.

*Postconditions:* `find(sets, e) = e`; that is,  $e$  is the set id of a singleton set containing  $e$ .

`void union(UnionFind sets, int s, int t)`

*Preconditions:* `find(sets, s) = s` and `find(sets, t) = t`, that is, both  $s$  and  $t$  are set ids, or "leaders." Also,  $s \neq t$ .

*Postconditions:* Let `/sets/` refer to the state of `sets` before the operation. Then for all  $x$  such that `find(/sets/, x) = s`, or `find(/sets/, x) = t`, we now have `find(sets, x) = u`. The value of  $u$  will be either  $s$  or  $t$ . All other `find` calls return the same value as before the union operation.

# 问题4: 其它ADT (续)

- dictionary

**Dict create()**

*Precondition:* none.

*Postconditions:* If  $d = \text{create}()$ , then:

1.  $d$  refers to a newly created object;
2.  $\text{member}(d, \text{id}) = \text{false}$  for all  $\text{id}$ .

**boolean member(Dict  $d$ , DictId  $\text{id}$ )**

*Precondition:* none.

**Object retrieve(Dict  $d$ , DictId  $\text{id}$ )**

*Precondition:*  $\text{member}(d) = \text{true}$ .

**void store(Dict  $d$ , DictId  $\text{id}$ , Object  $\text{info}$ )**

*Precondition:* none.

*Postconditions:*

1.  $\text{retrieve}(d, \text{id}) = \text{info}$ ;
2.  $\text{member}(d, \text{id}) = \text{true}$ ;

- (linked) list
- (binary) tree
- stack
- queue
- heap
- priority queue
- union-find
- dictionary
- ...

你准备好迎接一次综合挑战了吗？！

# 问题5: single-linkage agglomerative clustering

The screenshot shows the Clusty search interface. At the top, there is a navigation bar with links for 'web', 'news', 'images', 'maps', 'blogs', 'wikipedia', 'jobs', and 'more'. The search bar contains the text 'sun'. Below the search bar, there are tabs for 'clouds', 'sources', 'sites', and 'time'. The 'clouds' tab is selected, showing a list of categories and their counts. The 'Java Software' category is highlighted in red and contains 2 documents. The main content area displays the results for the 'Java Software' cluster, including a link to a news article about Sun Microsystems and Microsoft, and a link to the Java website.

Clusty

web news images maps blogs wikipedia jobs more »

sun Search advanced preferences

Cluster Microsystems » Java Software contains 2 documents.

clouds sources sites time remix

All Results (222)

- Microsystems (61)
  - Oracle (11)
  - Open Source, Community (5)
  - Downloads, Technology (5)
  - Linux (4)
  - Storage, Emcs (4)
  - Design (4)
- Java Software (2)**
  - Simpler, Nasdaq (3)
  - Legal, Federal (2)
  - Sun Microsystems Unveils (2)
- Images (23)
- Solar System (12)
  - Largest object (3)
  - Planets, Solar System Exploration (3)
  - Astronomy (2)
  - Dictionary (2)
  - Observer (2)
  - Other Topics (1)

[Sun, Microsoft snipe over Java Bid for world standard reignites war of words](#)

sun Microsystems Inc. and Microsoft Corp. yesterday traded insults over sun 's campaign to have its Java computing system certified as a worldwide standard for **computer programming**. Java is a system that enables programmers to write software that will run on any type of machine. The same Java [www.highbeam.com/...7E10106A1603071C4A275E3F3C44390F7678700E720E0A60651A617F133D](#) - [cache] - Highbeam

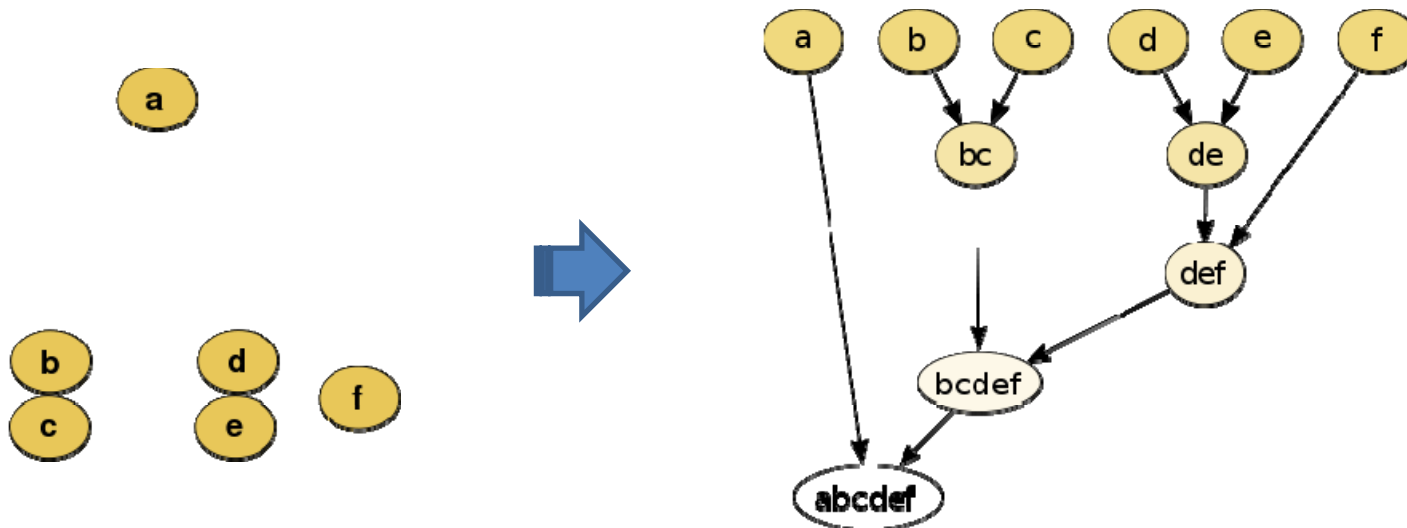
[java.com: Java + You](#)

Get the latest **Java Software** and explore how Java technology provides a better digital experience. [www.java.com](#) - [cache] - Yippy Sources III, Yippy Sources II



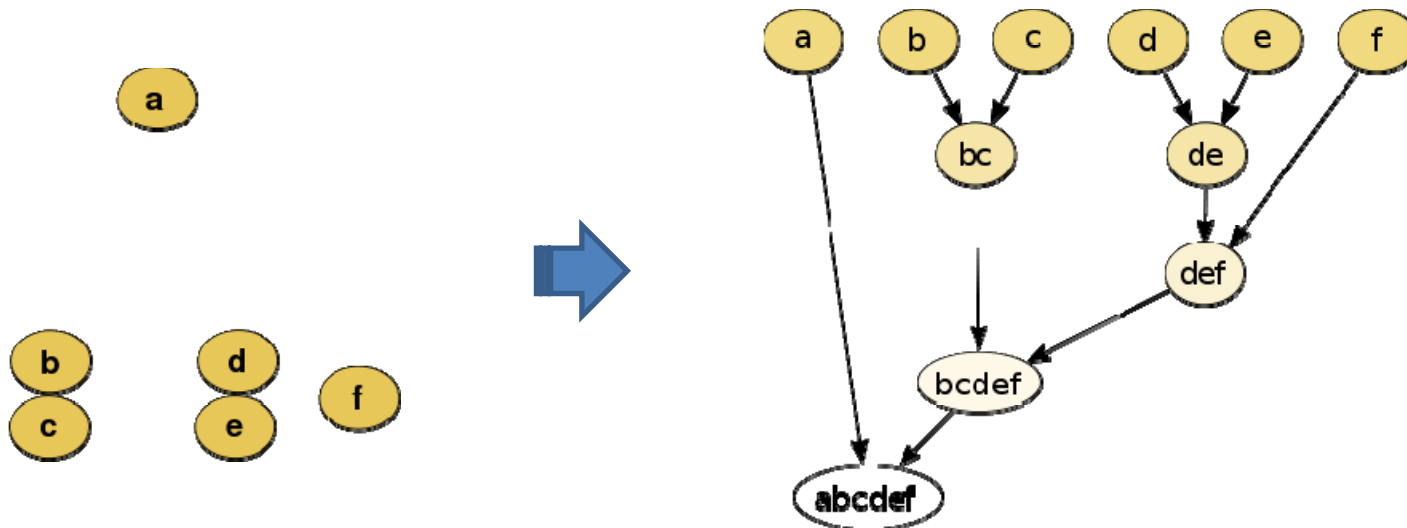
## 问题5: single-linkage agglomerative clustering (续)

- Agglomerative clustering
  - Each element starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Single linkage
  - The distance between two clusters is computed as the distance between the two closest elements in the two clusters.



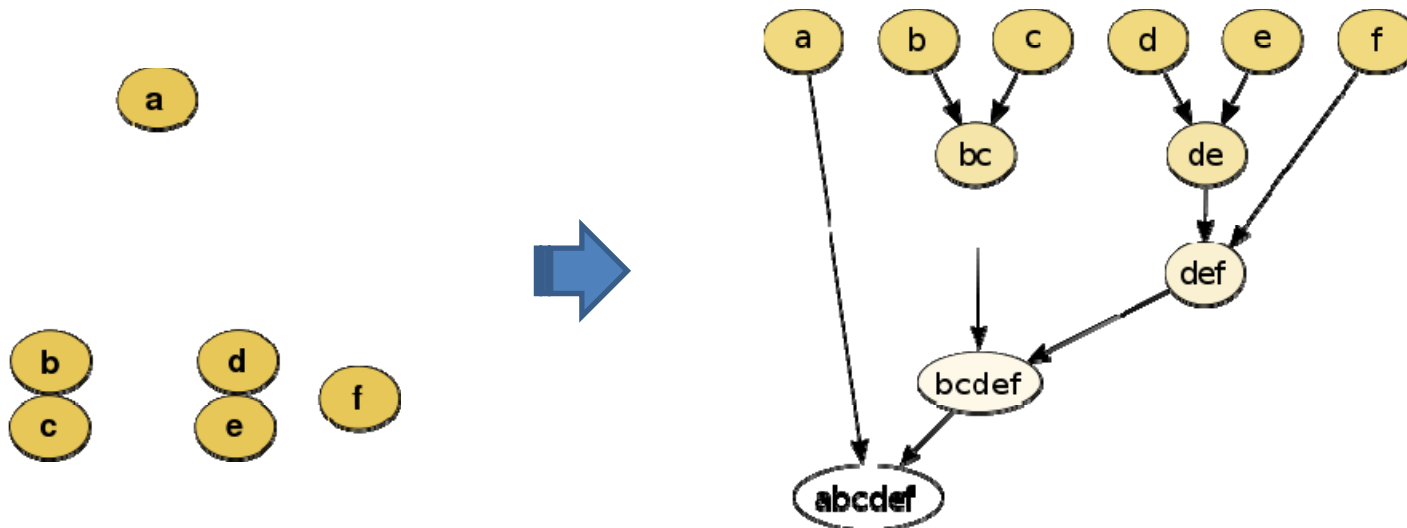
## 问题5: single-linkage agglomerative clustering (续)

- 最终的hierarchy应该用怎样的ADT来保存呢?
  - binary out-tree



## 问题5: single-linkage agglomerative clustering (续)

- 怎样计算出这个hierarchy呢, 需要用到哪些ADT?
  - priority queue (+ dictionary)



## 问题5: single-linkage agglomerative clustering (续)

- 如果element非常多, 计算需要很久, 如何在计算过程中监控任意一个element当前所属的cluster?
  - union-find

