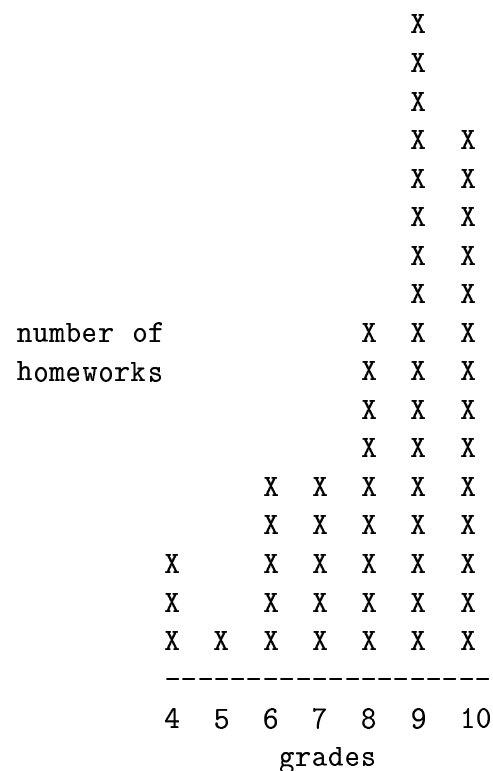


Algorithms: Solutions 4



The histogram shows the distribution of grades.

Problem 1

Let $A[1..n]$ be a sorted array of n distinct integer numbers. Write an efficient algorithm $\text{INDEX-SEARCH}(A, n)$ that finds an index i such that $A[i] = i$. If the array does not have such an element, the algorithm should return 0.

The algorithm is almost identical to BINARY-SEARCH , and its time complexity is $O(\lg n)$. It works only for integer arrays, since it is based on the assumption that, for every two indices p and r (where $p \leq r$), we have $A[r] - A[p] \geq r - p$.

```

INDEX-SEARCH( $A, n$ )
 $p \leftarrow 1$ 
 $r \leftarrow n$ 
while  $p < r$ 
    do  $q = \lfloor (p + r) / 2 \rfloor$ 
        if  $q \leq A[q]$ 
            then  $r \leftarrow q$ 
        else  $p \leftarrow q + 1$ 
if  $p = A[p]$ 
    then return  $p$ 
    else return 0
    
```

Problem 2

Suppose that $A[1..n]$ is a heap and you need to change its i th element. Give an algorithm $\text{CHANGE-ELEMENT}(A, i, n, k)$ that sets $A[i] \leftarrow k$ and updates the heap appropriately.

```

CHANGE-ELEMENT( $A, i, n, k$ )
if ( $A[i] > k$ )
    then  $A[i] \leftarrow k$ 
        HEAPIFY( $A, i, n$ )
    else while  $i > 1$  and  $A[\text{PARENT}(i)] < k$ 
        do  $A[i] \leftarrow A[\text{PARENT}(i)]$ 
             $i \leftarrow \text{PARENT}(i)$ 
     $A[i] \leftarrow k$ 

```

The running time of this algorithm is $O(\lg n)$.

Problem 3

A d -ary heap is like a binary heap, but instead of 2 children, nodes have d children.

(a) How would you represent a d -ary heap with n elements in an array? What are the expressions for determining the parent of a given element, $\text{PARENT}(i)$, and a j -th child of a given element, $\text{CHILD}(i, j)$, where $1 \leq j \leq d$?

The following expressions determine the parent and j -th child of element i (where $1 \leq j \leq d$):

$$\begin{aligned}\text{PARENT}(i) &= \left\lfloor \frac{i + d - 2}{d} \right\rfloor, \\ \text{CHILD}(i, j) &= (i - 1)d + j + 1.\end{aligned}$$

(b) Write an efficient implementation of HEAPIFY and HEAP-INSERT for a d -ary heap.

The HEAPIFY algorithm is somewhat different from the binary-heap version, whereas HEAP-INSERT is identical to the corresponding algorithm for binary heaps. The running time of HEAPIFY is $O(d \cdot \log_d n)$, and the running time of HEAP-INSERT is $O(\log_d n)$.

```

HEAPIFY( $A, i, n, d$ )
     $largest \leftarrow i$ 
    for  $l \leftarrow \text{CHILD}(i, 1)$  to  $\text{CHILD}(i, d)$     ▷ loop through all children of  $i$ 
        do if  $l \leq n$  and  $A[l] > A[largest]$ 
            then  $largest \leftarrow l$ 
    if  $largest \neq i$ 
        then exchange  $A[i] \leftrightarrow A[largest]$ 
            HEAPIFY( $A, largest$ )

```

```

HEAP-INSERT( $A, key$ )
     $heap\text{-}size[A] \leftarrow heap\text{-}size[A] + 1$ 
     $i \leftarrow heap\text{-}size[A]$ 
    while  $i > 1$  and  $A[\text{PARENT}(i)] < key$ 
        do  $A[i] \leftarrow A[\text{PARENT}(i)]$ 
             $i \leftarrow \text{PARENT}(i)$ 
     $A[i] \leftarrow key$ 

```

Problem 4

What is the height of a d -ary heap of n elements in terms of n and d ?

The height h of a heap is *approximately* equal to $\log_d n$. The exact height is

$$h = \lceil \log_d(nd - n + 1) - 1 \rceil.$$