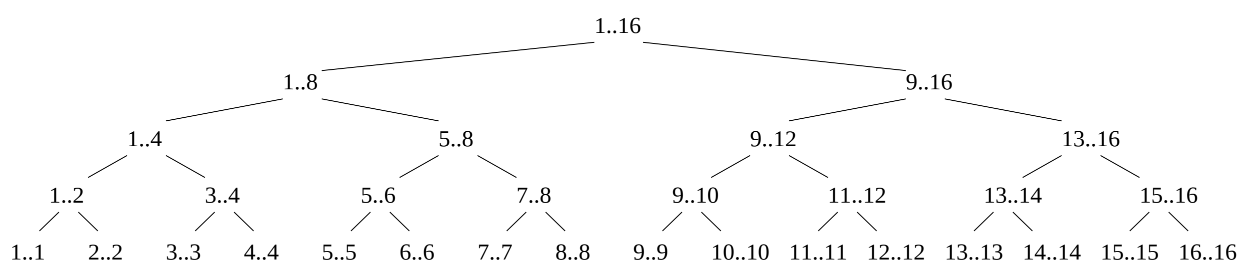
Analysis of Algorithms: Solutions 8

```
                                X
                                X
                                X
                                X
                                X
                                X
                          X     X
number of                 X     X
homeworks     X  X     X  X  X  X
              X  X     X  X  X  X
              X  X  X  X  X  X  X
           X  X  X  X  X  X  X  X  X
           -------------------------
           1  2  3  4  5  6  7  8  9
```

The histogram shows the distribution of grades for the homeworks submitted on time.

Problem 1
Using Figure 16.2 in the textbook as a model, draw the recursion tree for the MERGE-SORT procedure on a sixteen-element array. Explain why dynamic programming is ineffective for speeding up MERGE-SORT.



The MERGE-SORT procedure does *not* have overlapping subproblems, that is, all nodes of the recursion tree are distinct. We cannot re-use the results of recursive calls and, hence, dynamic programming does not improve the efficiency.

Problem 2
Determine a longest common subsequence of $\langle 1,0,0,1,0,1\rangle$ and $\langle 0,1,0,1,1,0,1,1\rangle$. Using Figure 16.3 in the book as a model, draw the table constructed by the LCS-LENGTH algorithm for these two sequences (you do *not* need to show arrows in your table).

The longest common subsequences are $\langle 0,0,1,0,1\rangle$, $\langle 1,0,1,0,1\rangle$, and $\langle 1,0,0,1,1\rangle$. The table computed by LCS-LENGTH is as follows:

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

Problem 3
Write an algorithm INCREASING-LENGTH(A,n) that determines the *length* of a longest increasing subsequence of an array $A[n]$.

Let $d[i]$ denote the length of a longest increasing subsequence whose last element is $A[i]$. For example, suppose that $A=\langle 1,2,1,2,3\rangle$ and $i=3$; then, the longest increasing subsequence that ends at $A[3]$ is $\langle 1,1\rangle$, which implies that $d[3]=2$.

Observe that this subsequence contains at least one element and, hence, $d[i]\geq 1$. Furthermore, if $j<i$ and $A[j]\leq A[i]$, then $d[i]\geq d[j]+1$, because we may construct a $(d[j]+1)$-element increasing subsequence that ends at $A[i]$, by adding $A[i]$ to the longest increasing subsequence that ends at $A[j]$. These observations lead to the following algorithm, whose running time is $\Theta(n^2)$:

```
INCREASING-LENGTH(A, n)
d-max ← 0
for i ← 1 to n
    do d[i] ← 1
        for j ← 1 to i − 1
            do if A[i] ≥ A[j] and d[i] < d[j] + 1
                    then d[i] ← d[j] + 1
    if d-max < d[i]
        then d-max ← d[i]
return d-max
```