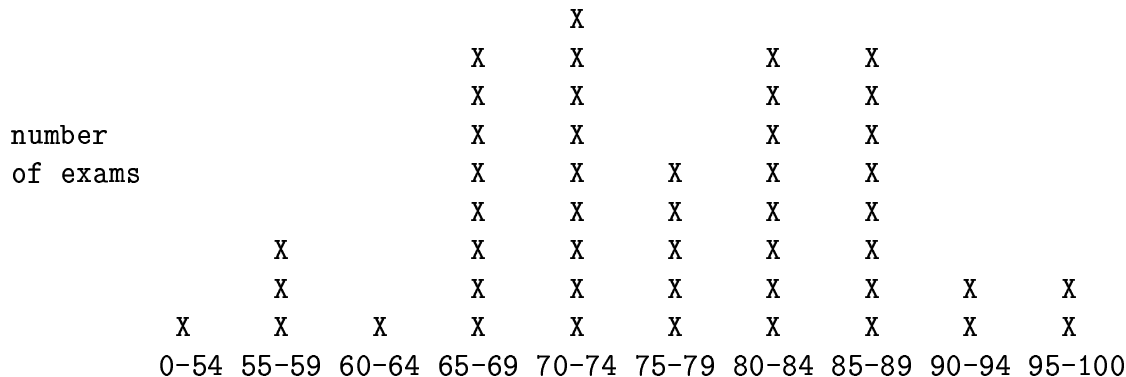


Analysis of Algorithms: Results of Exam 1



The histogram shows the distribution of grades, *not* including the bonus.

Problem 10

Suppose that you are using a programming language that allows only integer numbers and supports four operations on them: addition, subtraction, multiplication, and integer division; the running time of each operation is constant, that is, $\Theta(1)$. The result of the integer division of n over m is $\lfloor n/m \rfloor$; for example, $\lfloor 10/3 \rfloor = 3$. The language does *not* allow fractional numbers, and does *not* have operations for logarithms and exponentiation. Write an efficient algorithm $\text{POWER}(n, m)$ that computes n^m , where n and m are positive integers, and give the asymptotic time complexity (Θ -notation) of your algorithm.

Simple algorithm

The following brute-force computation takes $\Theta(m)$ time:

```

SIMPLE-POWER( $n, m$ )
 $k \leftarrow 1$ 
for  $i \leftarrow 1$  to  $m$ 
    do  $k \leftarrow k \cdot n$ 
return  $k$ 
    
```

Fast algorithm

We next note that, if m is even, then $n^m = (n \cdot n)^{\lfloor m/2 \rfloor}$; similarly, if m is odd, then $n^m = n \cdot (n \cdot n)^{\lfloor m/2 \rfloor}$. These observations lead to a faster algorithm, whose complexity is $\Theta(\lg m)$:

```

FAST-POWER( $n, m$ )
if  $m = 0$ 
    then return 1
if  $\lfloor m/2 \rfloor \cdot 2 = m$   $\triangleright$  Is  $m$  even?
    then return FAST-POWER( $n \cdot n, \lfloor m/2 \rfloor$ )
    else return  $n \cdot$ FAST-POWER( $n \cdot n, \lfloor m/2 \rfloor$ )
    
```