2. Write an algorithm that finds the m smallest numbers in a list of n numbers.

6. Write an algorithm that finds both the smallest and largest numbers in a list of n numbers. Try to find a method that does at most $1.5n$ comparisons of array items.

10. Define basic operations for your algorithms in Exercises 1–7, and study the performance of these algorithms. If a given algorithm has an every-case time complexity, determine it. Otherwise, determine the worst-case time complexity.

12. Write a $\Theta(n)$ algorithm that sorts n distinct integers, ranging in size between 1 and kn inclusive, where k is a constant positive integer. (*Hint:* Use a kn-element array.)

25. Presently we can solve problem instances of size 30 in 1 minute using algorithm A, which is a $\Theta(2^n)$ algorithm. On the other hand, we will soon have to solve problem instances twice this large in 1 minute. Do you think it would help to buy a faster (and more expensive) computer?

26. Consider the following algorithm:

```
for (i = 1; i <= 1.5n; i++)
      cout << i;
for (i = n; i >= 1; i--)
      cout << i;
```

(a) What is the output when $n = 2$, $n = 4$, and $n = 6$?

(b) What is the time complexity $T(n)$? You may assume that the input n is divisible by 2.

27. Consider the following algorithm:

```
j = 1;
while (j <= n/2) {
    i = 1;
    while (i <= j) {

        cout << j << i;
        i++;
    }
    j++;
}
```

(a) What is the output when $n = 6$, $n = 8$, and $n = 10$?

(b) What is the time complexity $T(n)$? You may assume that the input n is divisible by 2.

30. What is the time complexity $T(n)$ of the nested loops below? For simplicity, you may assume that n is a power of 2. That is, $n = 2^k$ for some positive integer k.

```
    ⋮
i = n;
while (i >= 1){
    j = i;
    while (j <= n){
        < body of the while loop>    //Needs Θ(1).
        j = 2 * j;
    }
        i = ⌊i/2⌋;
}
    ⋮
```