

Lecture 26. Basics Counting Operations in Algorithms

Example 26.1. What is the total number of additions performed by the code?

```
for  $i \in \{1, 2, \dots, n\}$  do  
  print  $i + i + i$ 
```

Loop

Example 26.1. What is the total number of additions performed by the code?

```
for  $i \in \{1, 2, \dots, n\}$  do  
  print  $i + i + i$ 
```

Multiplication Principle for Algorithms. If **STATEMENT** requires m of a certain type of operation, then a **loop** that repeats **STATEMENT** n times requires mn operations.

Example 26.2. Let A be the set of 26 letters of alphabet. Interpret the following algorithm and count how many strings does it print?

```
for  $c \in A$  do
    print  $c$ 
for  $c \in A$  do
    ⌈ for  $d \in A$  do
        ⌋    print  $cd$ 
for  $c \in A$  do
    ⌈ for  $d \in A$  do
        ⌈ for  $e \in A$  do
            ⌋    ⌋    print  $cde$ 
```

Array

An **array** is a sequence of variables $x_1, x_2, x_3, \dots, x_n$; e.g.,

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
2.4	7.3	3.1	2.7	1.1	2.4	8.2	0.3	4.9	7.3

Notice that the order of the elements in an array matters.

Array

An **array** is a sequence of variables $x_1, x_2, x_3, \dots, x_n$; e.g.,

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
2.4	7.3	3.1	2.7	1.1	2.4	8.2	0.3	4.9	7.3

Notice that the order of the elements in an array matters.

Example 26.3. The following algorithm counts the number of duplicates in the array x_1, x_2, \dots, x_n , where $n > 1$. How many "=" comparisons does the algorithm perform? For example, when $x_1 = 20, x_2 = 50, x_3 = 50, x_4 = 18$. (Use the trace table of the algorithm.)

```
t ← 0
for i ∈ {1, 2, 3, ..., n - 1} do
  for j ∈ {i + 1, ..., n} do
    if  $x_i = x_j$  then t ← t + 1
```

Trace of the algorithm in Example 26.3

x_1	x_2	x_3	x_4	t	i	j	Comparison
20	50	50	18	0	1	2	$20 \stackrel{?}{=} 50$
20	50	50	18	0	1	3	$20 \stackrel{?}{=} 50$
20	50	50	18	0	1	4	$20 \stackrel{?}{=} 18$
20	50	50	18	0	2	3	$50 \stackrel{?}{=} 50$
20	50	50	18	1	2	4	$50 \stackrel{?}{=} 18$
20	50	50	18	1	3	4	$50 \stackrel{?}{=} 18$

The number of comparisons is evidently $3+2+1=6$ in this case. In general, this algorithm will make

$$S_n = (n-1) + (n-2) + \cdots + 2 + 1 =$$

Trace of the algorithm in Example 26.3

x_1	x_2	x_3	x_4	t	i	j	Comparison
20	50	50	18	0	1	2	$20 \stackrel{?}{=} 50$
20	50	50	18	0	1	3	$20 \stackrel{?}{=} 50$
20	50	50	18	0	1	4	$20 \stackrel{?}{=} 18$
20	50	50	18	0	2	3	$50 \stackrel{?}{=} 50$
20	50	50	18	1	2	4	$50 \stackrel{?}{=} 18$
20	50	50	18	1	3	4	$50 \stackrel{?}{=} 18$

The number of comparisons is evidently $3+2+1=6$ in this case. In general, this algorithm will make

$$S_n = (n-1) + (n-2) + \cdots + 2 + 1 = \frac{n(n-1)}{2} \in \text{Big-}\Theta(n^2)$$

Remark. If $S_n \leq \frac{n(n-1)}{2}$, then we say $S_n \in \mathcal{O}(n^2)$. If $S_n \geq \frac{n(n-1)}{2}$, then we say $S_n \in \Omega(n^2)$.

Practice 26.4. Consider the same algorithm in Example 25.3.

```
 $t \leftarrow 0$   
for  $i \in \{1, 2, 3, \dots, n-1\}$  do  
  for  $j \in \{i+1, \dots, n\}$  do  
    if  $x_i = x_j$  then  $t \leftarrow t + 1$ 
```

Let $x_1 = 10, x_2 = 20, x_3 = 30, x_4 = 20, x_5 = 10$. When tracing the algorithm, choose the right sequence of t in order.

Sorting

A **sort** is an algorithm that guarantees that

$$x_1 \leq x_2 \leq x_3 \leq \cdots \leq x_n$$

after the algorithm finishes.

Sorting

A **sort** is an algorithm that guarantees that

$$x_1 \leq x_2 \leq x_3 \leq \cdots \leq x_n$$

after the algorithm finishes.

Example 26.5. Let x_1, x_2, \dots, x_n be an array whose elements can be compared by \leq . The following algorithm is called a **bubble sort**. Step through this algorithm for the list $x_1 = 9, x_2 = 4, x_3 = 7$, and $x_4 = 1$. How many times does it make the “>” comparison when sorting a list of n elements?

```
for  $i \in \{1, 2, \dots, n-1\}$  do
  for  $j \in \{1, 2, \dots, n-i\}$  do
    if  $x_j > x_{j+1}$  then swap  $x_j$  and  $x_{j+1}$ 
```

```
def bubbleSort(alist):
    for i in range(0, len(alist)-1):
        for j in range(0, len(alist)-1-i):
            if alist[j] > alist[j+1]:
                alist[j], alist[j+1] = alist[j+1], alist[j]

alist = [54,26,93,17,77,31,44,55,20]
bubbleSort(alist)
print(alist)
```

Solution: The following table shows how the program variables change during execution. Each row of the table gives the value of the program variables before the if statement is executed. The last two columns show the result of each “>” comparison.

x_1	x_2	x_3	x_4	i	j	Comparison	Result
9	4	7	1	1	1	$x_1 \stackrel{?}{>} x_2$	yes
4	9	7	1	1	2	$x_2 \stackrel{?}{>} x_3$	yes
4	7	9	1	1	3	$x_3 \stackrel{?}{>} x_4$	yes
4	7	1	9	2	1	$x_1 \stackrel{?}{>} x_2$	no
4	7	1	9	2	2	$x_2 \stackrel{?}{>} x_3$	yes
4	1	7	9	3	1	$x_1 \stackrel{?}{>} x_2$	yes
1	4	7	9				

After the last comparison and swap, the loops are finished, and the list is in order. ♦

The bubble sort is so named because the largest elements tend to “bubble” up to the end of the list, like the bubbles in a soft drink. Look back at the above trace and notice that the 9 bubbled to the end of the list when i was 1, then the 7 bubbled up next to it when i was 2, and so on.

The ‘if’ statement requires one comparison. The outside loop makes the inside loop execute $n - 1$ times, but each time the size of the index set for j gets smaller. Thus, the total number of comparisons is $(n - 1) + (n - 2) + \cdots + 2 + 1 = \frac{n(n-1)}{2} \in \text{Big-}\Theta(n^2)$.

Practice 26.6. Trace through the bubble sort algorithm for the following data set.

$$x_1 = 5, x_2 = 4, x_3 = 2, x_4 = 1, x_5 = 3.$$

```
for  $i \in \{1, 2, \dots, n-1\}$  do
  ⌈ for  $j \in \{1, 2, \dots, n-i\}$  do
    ⌊   if  $x_j > x_{j+1}$  then swap  $x_j$  and  $x_{j+1}$ 
```