Lecture 17

Queries

$$[25,\ 13,\ -100,\ -12,\ 60,\ 31,\ -76,\ 2]$$

1. What does the list look like after one pass of selection sort (i.e., one 'selection')?
2. What does the list look like after another pass of selection sort?
3. What does the list look like after another pass of selection sort?
4. What does the list look like after another pass of selection sort?

1 Sorting

Not to overdo the whole sorting thing, but let's take a look at how long we should expect this sort to take. What exactly do I mean by this? When we in computer science talk about "efficient" a piece of code is, we are generally referring to how well it scales. That is, as the size of the input gets larger, by how much does the time change? If I sort a list of size 10,000 and it takes 4.3 seconds, about how long will it take to sort a list of size 20,000? Let's look at the code and try to figure it out!

```
def selection_sort(lst):
    for index in range(len(lst) - 1):
        small_index = index_of_smallest(lst, index)
        swap(lst, index, small_index)


def index_of_smallest(lst, start):
    min_index = start

    for index in range(start + 1, len(lst)):
        if lst[index] < lst[min_index]:
            min_index = index

    return min_index


def swap(lst, i, j):
    temp = lst[i]
    lst[i] = lst[j]
    lst[j] = temp
```

Looking at the code, how many times does the loop in `index_of_smallest` run? So if we count the total number of times the loop happens, we'll get a vague idea of how efficient the code is.

$$(n-1) + (n-2) + \cdots + 2 + 1 = \frac{n(n-1)}{2}$$

So, for a list of size 100, we do 4,950 comparisons; while for a list of size 200, we do 19,900 comparisons. As the size of the list continues to increase, how does the number of comparisons performed increase?

Size (n)	Comparisons	Factor Increase
100	4,950	
200	19,900	4.02
400	79,800	4.01
800	319,600	4.01
8,000	31,996,000	100.1

As we'll now observe, this increase in the number of comparisons directly correlates with the amount of time needed to perform the sort. The sizes above make for really boring examples, so we'll start with a list of size 5,000.

Size (n)	Time (s)		Factor Increase
5,000	1.05		
10,000	4.23		4.03
20,000	17.55		4.15
40,000	70.2		
80,000	280.8	4m 41s	
1,000,000	43,875	12h 11m 15s	
100,000,000	438,750,000	13.9 years	

This is what I meant by "selection sort isn't a very good sort". There are better general purpose sorting algorithms that you will spend time understanding next quarter.

Truth Tables!

Start simple: p or q
Get harder: not ((not p or q) and (not q or r)) or (not p or r)

Methods?