CSCI 262
Data Structures

17 – Selection Sort

Sorting

- Input: a list of elements, e.g. integers
- Output: a list of the input elements in sorted order

Why do we study this problem?

- Teaching example
  - Algorithm design
  - Algorithm analysis
- Sorting is also useful for all sorts of applications

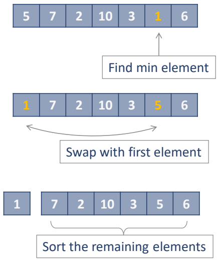
Selection Sort

- Input: a list of elements, e.g. integers
- Output: a list of the input elements in sorted order

- A simple solution:
  - Find the minimum element in the list
  - Swap it with the first element in the list
  - Sort the sublist after the first element

- This sorting algorithm is named **selection sort**.

Selection Sort Illustrated



Find min element

Swap with first element

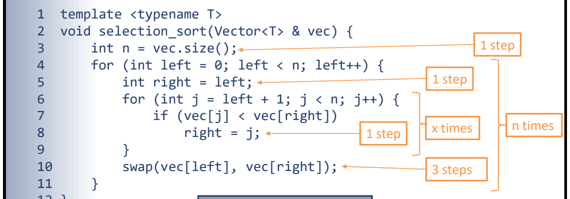
Sort the remaining elements

Selection Sort Code

```
template <typename T>
void selection_sort(vector<T> & vec) {
    int n = vec.size();
    for (int left = 0; left < n; left++) {
        int right = left;
        for (int j = left + 1; j < n; j++) {
            if (vec[j] < vec[right]) right = j;
        }
        swap(vec[left], vec[right]);
    }
}
```

Analyzing Selection Sort

Recall we want to count *basic computer steps*…



```
template <typename T>
void selection_sort(Vector<T> & vec) {
    int n = vec.size();
    for (int left = 0; left < n; left++) {
        int right = left;
        for (int j = left + 1; j < n; j++) {
            if (vec[j] < vec[right])
                right = j;
        }
        swap(vec[left], vec[right]);
    }
}
```

What is x? Ans: n – left – 1.
How do we add these up?

Analyzing Selection Sort

Things we can easily count:

1 step (line 3)

4n steps (lines 5 and 10)

Things that are trickier:

n – left – 1 (different value of left each time)

Analyzing Selection Sort

Just have to count carefully:

1^{st} time through:

left = 0, so n – left – 1 = n – 1

2^{nd} time through:

left = 1, so n – left – 1 = n – 2

…

Last time through:

left = n – 1, so n – left – 1 = 0

Analyzing Selection Sort

Putting it all together, we have:
Cost of selection sort is

$$1 + 4n + \underbrace{n - 1 + n - 2 + \dots + 0}_{\text{We know this!}}$$

= 1 + 4n + n(n – 1)/2

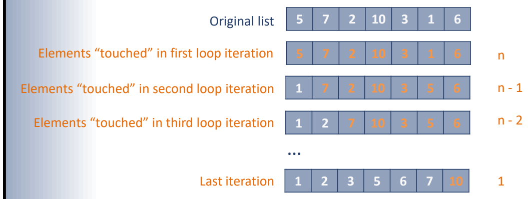
= $n^2/2 + 7n/2 + 1$

What is the "big-O" of this expression?

Visual Analysis

Preceding pages were very rigorous in counting

Sometimes, a visual approach is simpler:



Original list	5	7	2	10	3	1	6	
Elements "touched" in first loop iteration	5	7	2	10	3	1	6	n
Elements "touched" in second loop iteration	1	7	2	10	3	5	6	n - 1
Elements "touched" in third loop iteration	1	2	7	10	3	5	6	n - 2
	…							
Last iteration	1	2	3	5	6	7	10	1

Cost: O(1 + 2 + … + n)

Analysis Complete

Selection sort is $O(n^2)$

Can we do better?

(Yes, to be continued)

Up Next

- Friday, March 23
  - Lab 9 (continued)
  - Extra credit APTs assigned

-SPRING BREAK!-

- Monday, April 2
  - Lab 9 due