CSForAll Holmen
January 11, 2020

- Algorithms
- Data Representation
- Teacher Lesson Presentations
- Wisconsin CS Standards
- Next Meeting

Algorithms Characteristics

- Well-Order, Step-by-step
- Unambiguous operations
- Effectively computable operations
- Input transformed into output
- Finite number of steps, Halts in a finite amount of time

Algorithm Building Blocks

- Variables and expressions
- **Instruction sequences**
- **Selection instructions**
- **Iterative instructions**
- Functions

Sorting

- Selection Sort
- Insertion Sort

Compare Two Items

- Items in a collection of values to be sorted must be comparable
- For example given items x and y it must be possible to determine if x < y, x = y or x > y

Compare Examples

- Integers
- Single Letters
    - Case sensitive
- Single Characters
    - Character codes
- Strings
- Multi-key sorts

Ordering

- Ascending order
- Descending order

Swap Operation

- Many sorting algorithms use a swap operation to swap values
- For example if x has the value 10 and y has the value 15 then after the operation swap(x,y) x has the value 15 and y has the value 10

Swap Code
Swap the values of x and y

```
temp = x
x = y
y = temp
```

Selection Sort Ascending Order

Initial	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5
20	1	1	1	1	1
7	7	3	3	3	3
14	14	14	7	7	7
3	3	7	14	10	10
1	20	20	20	20	14
10	10	10	10	14	20

Selection Sort

Suppose x is a list or array of n integers that can be indexed by position.
Positions begin at 0 so there are elements in x at positions 0 through n -1

```
i = 0
Repeat the follow steps n -1 times
    find the location of the smallest value in positions i through n -1
    call the position of the smallest value s
    swap the values at position i and s
    i = i + 1
```

Selection Sort Unplugged Activity

Selection Sort

Let x be a list or array of n integers and let x[k] references the k-th integer in the array. Legal values for k are 0 through n-1.

```
for (p = 0; p < n-1; p++) {
    s = p;
    for (j = p+1; j < n; j++) {
        if (x[j] < x[s])
            s = j;
    }
    temp = x[p];
    x[p] = x[s];
    x[s] = temp;
}
```

Insertion Sort Ascending Order

Initial	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5
20	7	7	3	1	1
7	20	14	7	3	3
14	14	20	14	7	7
3	3	3	20	14	10
1	1	1	1	20	14
10	10	10	10	10	20

Insertion Sort

Suppose x is a list or array of n integers that can be indexed by position.
Positions begin at 0 so there are elements in x with positions 0 through n-1

```
i = 1
Repeat the following process n-1 times
    copy the element in position i into a variable named temp
    j = i -1
    repeat the following process until j < 0 or
                                temp is larger than the element at position j

        copy the element at position j to position j+1
        j = j - 1

    copy the value of temp into position j+1
    i = i + 1
```

Insertion Sort Unplugged Activity

Insertion Sort

Let x be a list or array of n integers
x[i] references the i-th integer in the array
Legal values for i are 0 through n-1

```
for (i = 1; i < n; i++) {
        j = i-1;
        temp = x[i];
        while (j>=0 && x[j] > temp) {
                x[j+1] = x[j];
                j = j - 1;
        }
        x[j+1] = temp;
}
```