A Tour of O(N^2) Sorting Algorithms

Today's Class

touching lives

- $O(N^2)$ Sorting Algorithms
  - Bubble Sort
  - Selection Sort
  - Insertion Sort

- Animations
  - http://math.hws.edu/TMCM/java/xSortLab/
  - http://www.sorting-algorithms.com/
  - http://www.youtube.com/watch?v=kPRA0W1kECg

Bubble Sort

touching lives

- "Like bubbles floating to the surface"

```
void BubbleSort(int arr[], int size) {
   int i,j,tmp;
   for(j=1; j<size; j++) {
     for (i = 0; i < size - j; i++)
        if (arr[i] > arr[i + 1]) {
           tmp = arr[i]; arr[i] = arr[i + 1]; arr[i + 1] = tmp;
        }
   }
}
```

- Animation
  - http://en.wikipedia.org/wiki/Bubblesort
  - http://math.hws.edu/TMCM/java/xSortLab/

Bubble Sort: improved version

touching lives

- "Like bubbles floating to the surface"

```
void BubbleSort(int arr[], int size) {
   int i,j,tmp;
   for(j=1; j<size; j++) {
     bool flag = false;
     for (i = 0; i < size - j; i++)
        if (arr[i] > arr[i + 1]) {
           tmp = arr[i]; arr[i] = arr[i + 1]; arr[i + 1] = tmp;  flag = true;
        }
     if (! flag) break;
   }
}
```

- Animation
  - http://en.wikipedia.org/wiki/Bubblesort
  - http://math.hws.edu/TMCM/java/xSortLab/

Bubble Sort Analysis

touching lives

- How many steps in the worst case?

```
void BubbleSort(int arr[], int size) {
   int i,j,tmp;
   for(j=1; j<size; j++) {
     bool flag = false;
     for (i = 0; i < size - j; i++)
        if (arr[i] > arr[i + 1]) {
           tmp = arr[i]; arr[i] = arr[i + 1]; arr[i + 1] = tmp;  flag = true;
        }
     if (! flag) break;
   }
}
```

- (N-1) + (N-2) + (N-3) … + 1 = $\sum_{i=1}^{N-1} i$
- $= \frac{N \cdot (N-1)}{2} = \mathrm{O}(\mathrm{N}^2)$

Summations Sidebar

touching lives

- $\sum_{i=1}^{100} i =$
- 5050
  - 1 + 100 = 101
  - 2 + 99 = 101
  - 3 + 98 = 101
  - 4 + 97 = 101
  - … 50 times
- $(N+1) \cdot \frac{N}{2}$
- $= \frac{N^2+N}{2} = O(N^2)$

Selection Sort

touching lives

- Repeatedly select the maximum value and swap to the back

```
void SelectionSort(int num[], int size) {
      int i, j, first, temp;
      for (i = size - 1; i > 0; i--) {
            first = 0;
            for (j=1; j<=i; j++)   // locate largest value in positions 0 through i
                  if (num[j] > num[first]) first = j;
            temp = num[first];   // swap largest found with element at position i
            num[first] = num[i];
            num[i] = temp;
      }
}
```

- Animation
  - http://en.wikipedia.org/wiki/Selection_sort
  - http://math.hws.edu/TMCM/java/xSortLab/

Selection Sort

touching lives

- Another version: repeatedly select the minimum value and swap to the front

```
void SelectionSort(int num[], int size) {
      int i, j, first, temp;
      for (i = 0; i < size; i++) {
            first = i;
            for (j=i+1; j<size; j++)   // locate smallest value in positions i thru size-1
                  if (num[j] < num[first]) first = j;
            temp = num[first];        // swap smallest found with element at position i
            num[first] = num[i];
            num[i] = temp;
      }
}
```

- Animation
  - http://en.wikipedia.org/wiki/Selection_sort
  - http://math.hws.edu/TMCM/java/xSortLab/

Insertion Sort

touching lives

- Grow a sorted list at the front of the array by inserting elements one at a time.

```
void insertion_sort (int arr[], int size){
   int j, temp;
   for (int i = 1; i < size; i++){
       j = i;
       while (j > 0 && arr[j] < arr[j-1]){
          temp = arr[j]; arr[j] = arr[j-1]; arr[j-1] = temp; j--;
       }
    }
}
```

- Animation
  - http://en.wikipedia.org/wiki/Insertion_sort
  - http://math.hws.edu/TMCM/java/xSortLab/