Comparison-based sorting algorithms

- **Simple**: Selection Sort
  - (Insertion Sort, Bubble Sort, Shell Sort)
- **Good worst case**: HeapSort, AVLSort, MergeSort
- **Quick**: QuickSort
- **Imaginary**: StrawSort (aka, LukeSort)
- Can we do better?

Selection Sort idea

- Find the smallest element, put it first
- Find the next smallest element, put it second
- Find the next smallest, put it next
- etc.

Selection Sort

```c
void SelectionSort (Array a[1..n]) {
  for (i=0, i<n; ++i) {
    j = Find index of smallest entry in Array.
    Swap(a[i],a[j])
  }
} while (other people are coding QuickSort/MergeSort)

Twiddle thumbs
}
```

Running time?

Worst, Avg, Best N^2

HeapSort: sorting with a priority queue ADT (heap)

- Shove everything into a queue, take them out smallest to largest.

Running time?

Worst, avg N logN

AVL Sort?

1. Insert into tree (N * logN)
2. In-order traversal [O(n)]

Running time?

Worst, bestN logN
MergeSort

MergeSort (Array [1..n])
Split Array in half
Recursively sort each half
Merge two halves together

Merge Sort Running Time

T(n) = 2 T(N/2) + N
...  
T(N) = O(N logN)  
(best, worst)
Discuss in section

QuickSort

QuickSort Example

Pick a “pivot”. Divide into less-than & greater-than pivot.
Sort each side recursively.

Dealing with Slow QuickSorts

- Randomly permute input
  - Bad cases more common than simple probability would suggest. So, make it truly random.
- Pick pivot cleverly
  - “Median-of-3” rule takes Median(first, middle, last) element.
  - Average running time: N log N
- Choose pivot point randomly!

With good choice, fastest in practice!!
QuickSelect

- What if we want to find the $k^{th}$ smallest element in an array?
- What if $k = N/2$ (i.e., we want to find the median)?

StrawSort (aka, LukeSort)

Can we do any better?

Worst case time Lower Bound

- How many comparisons does it take before we can be sure of the order?
- This is the minimum # of comparisons that any algorithm could do.

Decision tree to sort list A,B,C

Max depth of the decision tree

1. Max # leaves for binary tree of height $h$?
2. Shallowest tree with $L$ leaves?
3. A decision tree to sort $N$ elements must have $N!$ leaves.

Therefore:
- Any sorting algorithm that uses only comparisons between elements requires at least $\log(N!)$ comparisons in the worst case!
- This plus some algebra yields bound: