CSE 326: Data Structures
Topic 11: Sorting by Comparison

Luke McDowell
Summer Quarter 2003

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

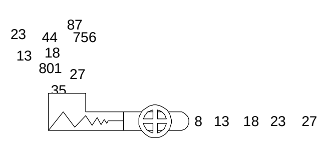
```
void SelectionSort (Array a[1..n]) {
      for (i=0, i<n; ++i) {
            j = Find index of smallest entry in Array.
            Swap(a[i],a[j])
      }

      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?

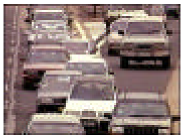
1. Insert into tree (N * logN)
2. In-order traversal [O(n)]

Running time?

Worst, bestNlogN

MergeSort

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



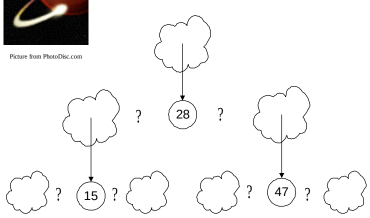
```
Merge (a1[1..n],a2[1..n])
i1=1, i2=1
While (i1<n, i2<n) {
      if (a1[i1] < a2[i2]) {
            Next is a1[i1]
            i1++
      } else {
            Next is a2[i2]
            i2++
      }
}
Now throw in the dregs…
```

MergeSort Running Time

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

QuickSort

Picture from PhotoDisc.com



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

QuickSort Example

7	2	8	3	5	9	6

Must swap pivot at end!

QuickSort Worst case

T(n) = N + T(n-1)
…
T(n) = O(N^2)

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)?

QuickSelect (Array A, int k)

Pick pivot:

1	2	3	4	5	6	7
7	2	8	3	5	9	6

Partition array:

1	2	3	4	5	6	7
5	2	6	3	7	9	8
				pindex		

- *k* == pindex?
- k < pindex?
- k > pindex?

Return pivot!

QuickSelect(left, k)

QuickSelect(right, k-pindex-1)

Running time?

O(n) – **one** recursive call!

StrawSort (aka, LukeSort)

"StrawMan"

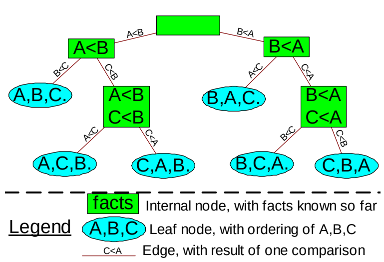
Can we do any better?

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.

Lower Bound – new analysis for us!!
True of **any algorithm**

Decision tree to sort list A,B,C



Leaves are final states.
Any alg will have same leaves,
in different places

Max depth of the decision tree

1. Max # leaves for binary tree of height *h?*

   2^h (complete, or induction)

2. Shallowest tree with *L* leaves?

   Ceil [log L]

3. A decision tree to sort N elements must have N! leaves.

   permutations

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:

O(N logN)