Sorting in Linear Time

Comparison Sorts — O(nlgn), $\Omega(nlgn)$ for some input
The best we can do for comparison sorts is $\Omega(nlgn)$.

Other techniques for sorting exist, such as Linear Sorting which is not based on comparisons. Linear Sorting techniques include:

_____

_____

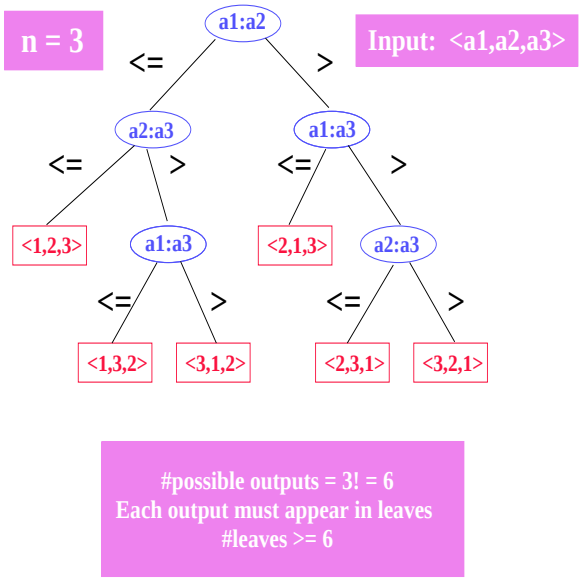
_____

Lower Bounds For Worst-Case Comparison Sorts

In general, assuming unique inputs, comparison sorts are expressed in terms of $a_i \leq a_j$ comparisons.

What is the best we can do on the worst case type of input?
What is the best worst-case running time?

Decision Tree Model



Analysis Of Decision Tree Model

Worst Case Comparisons is equal to height of decision tree

Lower bound on the worst case running time is the lower bound on the height of the decision tree.

Note that the number of leaves in the decision tree $\geq$ n!, where n = number elements in the input sequence.

A binary tree of height h has $\leq 2^h$ leaves.

$$n! \leq 2^h$$

By Stirling Approximation $n! = \sqrt{2\pi n}(n/e)^n(1+\Theta(1/n))$

Thus $n! > (n/e)^n$, $h \geq lg(n/e)^n$ = nlgn - nlge = $\Theta(nlgn)$

$$h \geq \lg(n!) \geq \Theta(nlgn)$$
$$h = \Omega(nlgn)$$

The best possible worst case running time for comparison sorts is thus _____.

Heapsort and Mergesort, both of which are _____, are asymptotically optimal.

Counting Sort

Assume algorithm is input n elements ranging from 1 to k.

Find the number of elements $j \leq A[i]$ and put A[i] in B[j].

If k = O(n), then T(n) for counting sort = $\Theta(n)$.

Example

Let array A be the following array (n=8, k=7).

$1 \ldots n$

5	7	3	5	2	1	4	3

Compute C[j] as the number of elements in A[] $\leq$ j. Array C is thus the following array.

Click mouse to advance to next frame.

Array B contains the final sorted array.

1	2	3	3	4	5	5	7

Pseudocode

```
CountingSort(A,B,k)                    Time
   for i = 1 to k                       k+1
      C[i] = 0                          k
   for j = 1 to length(A)               n+1
      C[A[j]] = C[A[j]] + 1             n
   for i = 2 to k                       k
      C[i] = C[i] + C[i-1]              k-1
   for j = length(A) downto 1           n+1
      B[C[A[j]]] = A[j]                 n
      C[A[j]] = C[A[j]] - 1             n
```

Analysis

$$\begin{aligned} \mathrm{T(n)} &= \mathrm{5n + 4k} \\ &= \Theta(n + k) \\ &= \Theta(n) \end{aligned}$$

$\Omega(nlgn)$ does not apply because CountingSort is not a comparison sort.

Stable Sort

Two elements having the same value appear in the same order in the sorted sequence as they did in the input sequence.

Counting Sort is a Stable Sort.

Array A:

5 (1)	7	3 (1)	5 (2)	2	1	4	3 (2)

Array B:

1	2	3 (1)	3 (2)	4	5 (1)	5 (2)	7

If we change line from

for j = length(A) downto 1

to

for j = 1 to length(A)

Would the algorithm still sort? ____

Would the algorithm still be a stable sort? ___

Radix Sort

This sort was originally used to sort computer punch-card decks.

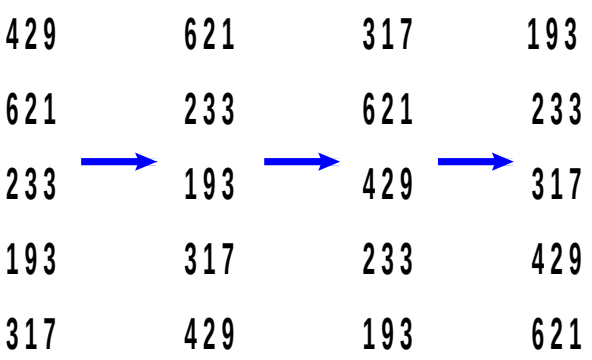
It is currently used for multi-key sorts (eg., year/month/day)

Consider each digit of the number as a separate key

Idea 1: Sort on most significant digit n, then sort on digit n-1, etc.
Problem: For old sorters, sort into 10 bins, but subsequent recursive sorts require all 10 bins. Operator must store the other 9 piles.

Idea 2: Sort on least significant digit, then sort on next least significant digit, etc.

Example




```
RadixSort(A,d)
   for i = 1 to d
      StableSort(A, digit(i))
```

Would the array always be sorted if we iterate from i=d down to 1? ___

Analysis

- If each digit is in the range 1 to k, use _____
- Each pass over a digit is $\Theta(n + k)$
- For d digits $\Theta(dn + dk)$
- If d is a constant and k = O(n), T(n) = _____

Radix-n implies each digit can differentiate among n different symbols. For example, in the previous case we assumed radix-10. This is why the name **Radix Sort** is given.

Accuracy

Why do later passes not mess up earlier sorts?

Prove that after pass p, sorted for digit p+1 to last digit

Prove by induction over p

- True for p=1. Apply Stable sort to digit 1, sorted after pass
- Assume true for p=i, prove true for p=i+1
- After pass i, compare two numbers x and y
- If x appears before y then one of following conditions must be true
  1. $x < y$ for digit i, then x belongs before y in sorted order
  2. $x = y$ for digit i, then $x < y$ for rest of number, placed in that order in earlier pass, not swapped because use stable sort

---

Example

Show how n integers in the range 1 to n^2 can be sorted in O(n) time.

Use a Radix-n sort.

Each digit requires n symbols, and $log_n n^2$ digits are needed (d=2, k=n).

$$\begin{aligned} \mathrm{T(n)} &= \Theta(2n + 2k) \\ &= \Theta(2n + 2n) \\ &= \Theta(n) \end{aligned}$$

Bucket Sort

Bucket Sort assumes that the input values are _____ distributed over the range [0,1), $0 \leq x < 1$.

Procedure:

- Divide inputs into n equal-sized subintervals (buckets) over the range [0,1).

- Sort buckets and concatenate the buckets.

- T(n) = _____

Example

```
BucketSort(A)                              COST*TIMES
   n = length(A)                            1
   for i = 1 to n                           n+1
      insert(A[i], B[floor(nA[i])]          n
   for i = 0 to n-1                         n+1
      InsertionSort(B[i])                   n*T(InsertionSort)
   return(B[0].B[1].....B[n-1])             n
```

Analysis

$T_{BS} = 4\text{n} + T_{IS}\text{n} + 3$

Let n_i = number of elements in bucket B[i].
The expected time to sort elements in B[i] = $E(O(n_i^2)) = O(E(n_i^2))$.

Note that the **expected value** of a random variable X as described in Chapter 6, is

$$E(X) \ = \ \sum_x xPr(X = x)$$

The variable x here represents the value of X, and Pr(X=x) represents the probability that the value of X is x.

The _____ of a random variable X with mean E[X] is notated Var[X].

- E[X^2] = Var[X] + E^2[X]
- P(X=k) = $\binom{n}{k} p^k q^{n-k}$ This is the binomial distribution for n trials.

Analysis

Thus, for all buckets,

$$\sum_{i=0}^{n-1} O(E(n_i^2)) \ = \ O(\sum_{i=0}^{n-1} E(n_i^2)).$$

Given:

- n elements, uniformly distributed over all possible values
- n buckets

What is the probability that an element will be inserted into some bucket i?

Answer: _____

Given n trials consisting of putting elements into buckets, how many elements will be inserted into bucket i?

$$n_i = \text{Binomial(n,p)}$$
$$\text{with mean } E(n_i) = \text{np} = 1$$
$$\text{and variance } Var(n_i) = \text{np(1-p)} = 1 - 1/\text{n}$$
$$E(n_i^2) = 1 - 1/\text{n} + 1^2$$
$$= 2 - 1/\text{n}$$
$$= \Theta(1)$$

Therefore, $T_{IS} = \Theta(1)$
and $T_{BS} = 4\text{n} + \Theta(1)\text{n} + 3 = \text{O(n)}$ for the average case. In the worst case, the run time is n^2.

Medians and Order Statistics

The selection problem (assume distinct elements)

Input: Set A of n numbers and integer i such that $1 \leq \text{i} \leq \text{n}$.

Output: $\text{x} \in \text{A}$ such that x is larger than i-1 elements of A (x is the ith smallest element).

i=1: _____

i=n: _____

i = $\frac{n+1}{2}$: _____ (actually $\lfloor \frac{n+1}{2} \rfloor$ and $\lceil \frac{n+1}{2} \rceil$).

Simple Solution: _____

return(A[i]).

This is _____.

Min and Max

Min(A): O(n) (____ comparisons)

Max(A): O(n) (____ comparisons)

Both: O(n) (_____ comparisons)

Selection in Expected Linear Time

Goal: Select ith smallest element from A[p..r].
Partition into A[p..q] and A[q+1..r]
If ith smallest element is in A[p..q]
then recurse on A[p..q]
else recurse on A[q+1..r]

```
Select(A, p, r, i)
1     if p = r
```

2 then return A[p]
3 else q = Partition(A, p, r)
4 k = q - p + 1
5 if i $\leq$ k
6 then return Select(A, p, q, i)
7 else return Select(A, q+1, r, i-k)

Example

Click mouse to advance to next frame.

Analysis

$$\text{T(n)} = \text{T(n-1)} + \Theta(n) \text{ in the worst case}$$
$$= \text{cn} + \text{c(n-1)} + \text{c(n-2)} + \ldots + \text{c}$$
$$= c\,\Sigma_{i=1}^{n}\, i = c\frac{n(n+1)}{2} = \Theta(n^2)$$
$$\text{T(n)} = \text{T(n/2)} + \Theta(n) \text{ in the best case}$$
$$\text{a=1, b=2, f(n)} = \Omega(n^{log_b a+\epsilon}),\ \epsilon = 1$$

Case 3: $\text{T(n)} = \Theta(n)$
Show: $1*\text{f(n/2)} \leq \text{cf(n)},\ \text{c} < 1$
$\text{f(n/2)} \leq \text{c*f(n)}$
$\text{c} >= 1/2$

Random-Partition

```
Random-Partition(A, p, r)
```

```
    i = Random(p, r)
    swap(A[p], A[i])
    return(Partition(A, p, r))


Random-Select(A, p, r, i)
    replace Partition with Random-Partition
    replace Select with Random-Select
```

These are still $\Theta(n^2)$ in the worst case
and O(n) in the average case.

However, we can get O(n) worst case performance with Select2.

Select2

O(n) Divide n elements into $\lfloor n/5 \rfloor$ groups of _ elements and one group of _____ elements

O(n) Find _____ of each group.

T($\lceil$(n/5)$\rceil$) Use Select2 recursively to find median of $\lceil \frac{n}{5} \rceil$ medians

O(n) Partition _ elements around median into ___ and ____ elements.

7n/10 + 6) If i $\leq$ k
then use Select2 to find ____ element in _____ lower elements
else use Select2 to find _____ element in _____ higher elements

Example
Click mouse to advance to next frame.

Analysis

$$\mathrm{T(n)} \leq T(\lceil \tfrac{n}{5} \rceil) + \mathrm{T(7n/10 + 6)} + \mathrm{O(n)}$$

To get T(7n/10 + 6), note that the smallest partition size would be

$$3(\lceil \tfrac{1}{2} \lceil \tfrac{n}{5} \rceil \rceil - 2) >= (\tfrac{3n}{10} - 6)$$

7n/10 + 6 denotes number of elements in larger partition

Assume $\mathrm{T(n)} \leq \mathrm{cn}$ for $\lceil n/5 \rceil$ and 7n/10+6

$$\mathrm{T(n)} = \mathrm{O(n)}$$

$$\mathrm{T(n)} \leq \mathrm{c(n/5)} + \mathrm{c(7n/10 + 6)} + \mathrm{O(n)}$$

$$= \mathrm{cn/5} + \mathrm{c7n/10} + \mathrm{c6} + \mathrm{O(n)}$$

$$= \mathrm{1/10(2cn} + \mathrm{7cn} + \mathrm{60c} + \mathrm{O(n))} \leq \mathrm{cn}$$

$$\mathrm{2cn} + \mathrm{7cn} + \mathrm{60c} + \mathrm{O(n)} \leq \mathrm{10cn}$$

$$\mathrm{60c} + \mathrm{O(n)} \leq \mathrm{cn}$$

$$\mathrm{c(n - 60)} \geq \mathrm{O(n)}$$

$$\mathrm{c} \geq \mathrm{O(n)} \ / \ \mathrm{(n - 60)}$$

c is a valid constant for large enough n.

Stopping condition: $\mathrm{T(n)} = \Theta(1)$, if $\mathrm{n} \leq 80$
