

QuickSort

- **Divide:** Partition the set into 2 subsets such that each element in one subset is smaller than or equal to each element in second subset.
- **Conquer:** Sort each subset using QuickSort recursively.
- **Combine:** In QuickSort: Do nothing!

QuickSort Algorithm

```
QUICKSORT(A, p, r)
```

```
if p < r then
```

```
    q = PARTITION(A, p, r);
```

```
    QUICKSORT(A, p, q-1);
```

```
    QUICKSORT(A, q+1, r);
```

QuickSort - Partition

15	45	7	13	5	42	2	99	8	32
----	----	---	----	---	----	---	----	---	----

i j

15	45	7	13	5	42	2	99	8	32
----	----	---	----	---	----	---	----	---	----

i j

15	45	7	13	5	42	2	99	8	32
----	----	---	----	---	----	---	----	---	----

i j

15	45	7	13	5	42	2	99	8	32
----	----	---	----	---	----	---	----	---	----

i j

15	7	45	13	5	42	2	99	8	32
----	---	----	----	---	----	---	----	---	----

i j

15	7	13	45	5	42	2	99	8	32
----	---	----	----	---	----	---	----	---	----

i j

15	7	13	5	45	42	2	99	8	32
----	---	----	---	----	----	---	----	---	----

i j

15	7	13	5	45	42	2	99	8	32
----	---	----	---	----	----	---	----	---	----

i j

15	7	13	5	2	42	45	99	8	32
----	---	----	---	---	----	----	----	---	----

i j

15	7	13	5	2	42	45	99	8	32
----	---	----	---	---	----	----	----	---	----

i j

15	7	13	5	2	8	45	99	42	32
----	---	----	---	---	---	----	----	----	----

i j

15	7	13	5	2	8	32	99	42	45
----	---	----	---	---	---	----	----	----	----

QuickSort – Best Case

- Even partitioning.

$$T(n) = \begin{cases} O(1), & n = 0 \\ 2T(n/2) + O(n), & n > 0 \end{cases}$$

- $T(n)$ is $O(n \log n)$ - Follows directly from the Master Theorem or by using substitution method.

QuickSort – Worst Case

- $0/n-1$ partitions.
- $T(0)$ is $O(1)$
- $T(n) \leq T(n-1) + T(0) + O(n) = T(n-1) + O(n)$
- $T(n)$ is $O(n^2)$
- Is $0/n-1$ partition the worst possible?

QuickSort – Worst Case

$$T(n) = \max\{T(q) + T(n-1-q)\} + O(n) \mid 0 \leq q \leq n-1\}$$

- **Guess:** $T(n) \leq cn^2$ for some constant c and $n \geq 1$.
- **Proof** (by induction):
- **Basis:** $T(0)$ is constant. Hence $T(0)$ is $O(1)$.
- **Step:** $T(n) \leq \max \{ cq^2 + c(n-1-q)^2 \} + O(n) =$

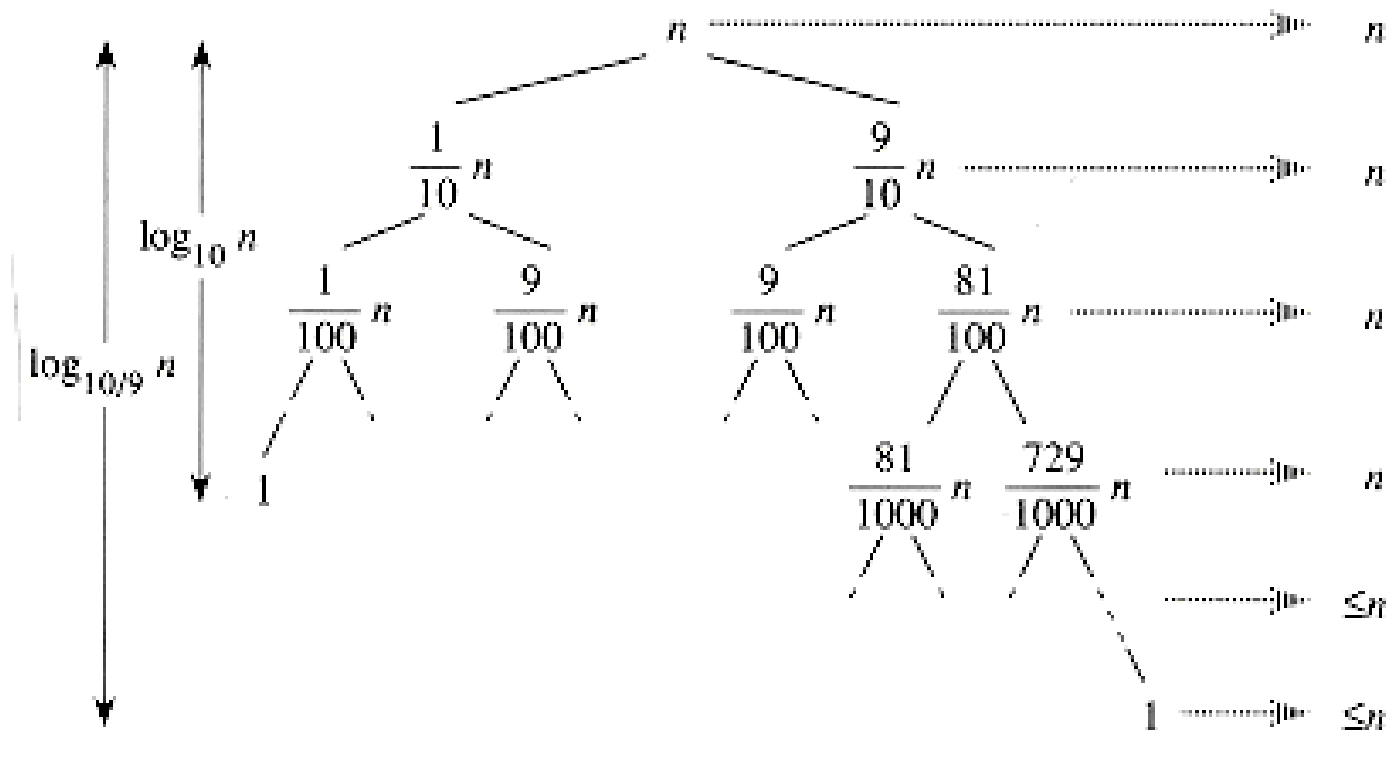
$$c \max \{ q^2 + (n-1-q)^2 \} + O(n)$$

is maximized when $q = 0$ or $q = n-1$ (exercise)

$$T(n) \leq c(n-1)^2 + O(n) = cn^2 - c(2n-1) + O(n) \leq cn^2$$

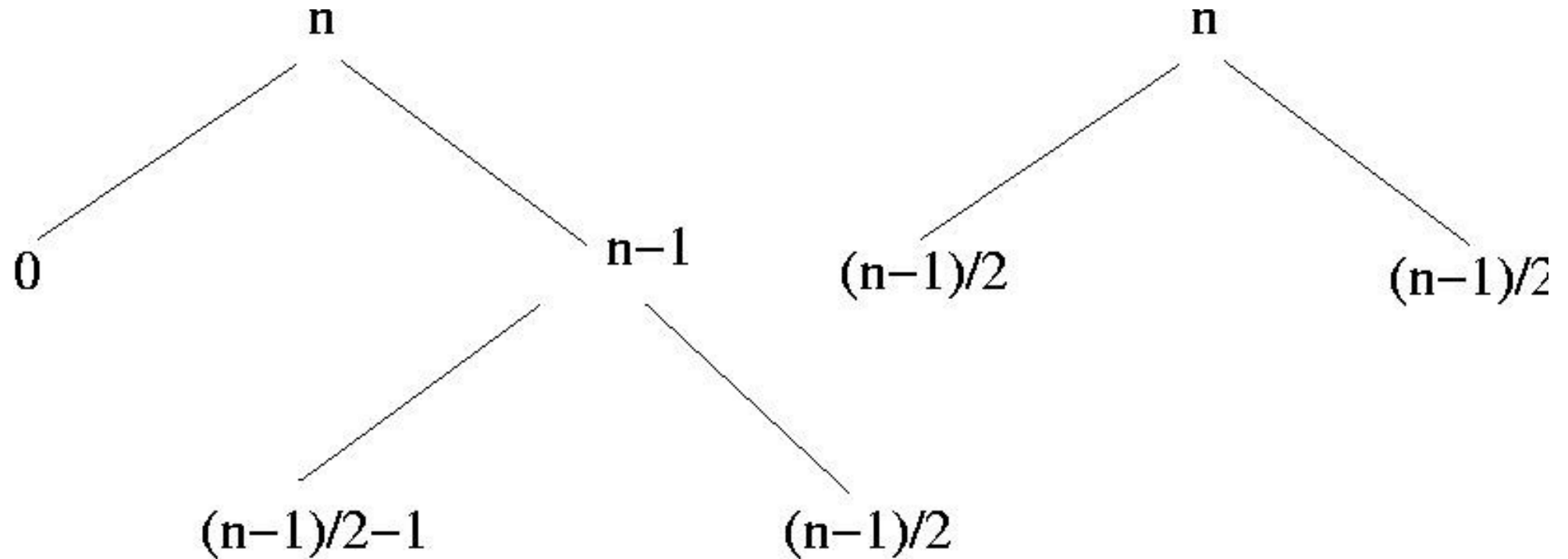
provided that c is selected so it dominates $O(n)$.

QuickSort – Average Case



$$\log_b n = \frac{\log_c n}{\log_c b}$$

QuickSort – Average Case



Randomized QuickSort

R-PARTITION(A,p,r)

$i = \text{RANDOM}(p, r)$

swap $A[r]$ and $A[i]$

$\text{PARTITION}(A, p, r)$

R-QUICKSORT(A,p,r)

if $p < r$ then

$q = \text{R-PARTITION}(A, p, r)$

$\text{R-QUICKSORT}(A, p, q-1)$

$\text{R-QUICKSORT}(A, q+1, r)$

Analysis of Randomized QuickSort

- What is the probability that z_i is compared to z_j ?
- z_i and z_j are compared at most once.
- z_i and z_j are compared iff z_i or z_j is the first pivot element chosen from $[z_i, z_j]$
- $P(z_i \text{ compared with } z_j) = 2/(j - i + 1)$
- Expected number of comparisons = sum of probabilities over all pairs i and j .

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{j=1}^{n-i} \frac{2}{j+i-i+1} < \sum_{i=1}^{n-1} \sum_{j=1}^n \frac{2}{j} = \sum_{i=1}^{n-1} O(\lg n) = O(n \lg n)$$

Counting Sort

- Elements to sort are integers between 1 and k .
- $A(i)$ = number of occurrences of i .
- $B(i) = A(1) + A(2) + \dots + A(i)$.
- Element i is inserted in $C(B(i)), B(i) = B(i)-1$
- Requires $O(k)$ working space.
- $O(k + n)$ time.
- Stable.

APPLET

Radix Sort

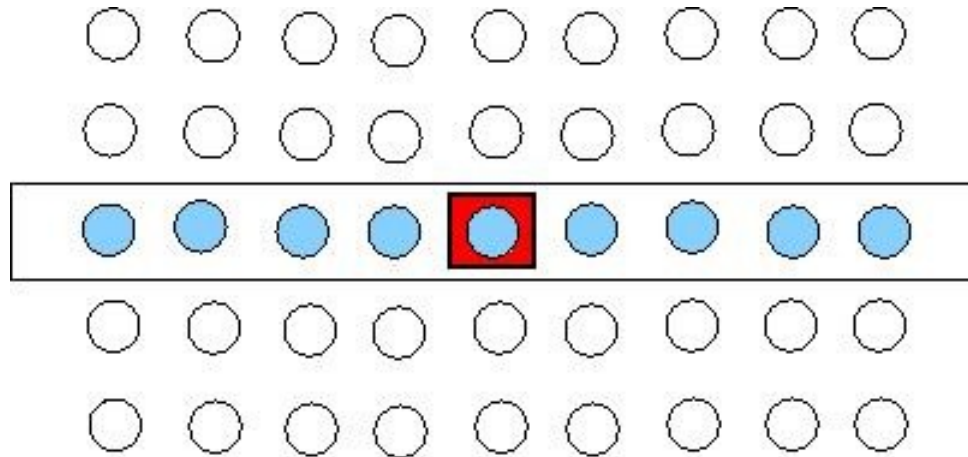
- Sort digit for digit.
- Most significant digit first?
- Least significant digit first?

APPLET

Bucket Sort

- Rational numbers in the interval $[0, 1]$.
- Divide $[0, 1]$ into n equal size subintervals.
- Distribute the elements into these subintervals.
- Use insertion sort in each of these buckets.
- Expected running time for uniformly distributed inputs: $O(n)$.

Median in Linear Time



$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 140 \\ T(\lceil \frac{n}{5} \rceil) + T(\frac{7n}{10} + 6) + O(n) & \text{if } n > 140 \end{cases}$$

Median in Linear Time

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 140 \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) & \text{if } n > 140 \end{cases}$$

We will show by induction that $T(n)$ is $O(n)$

Basis: $n=1, 2, 3, \dots, 140$

Step: Assume true for all $n < k$, k . Prove for $n=k$

$$\begin{aligned} T(n) &\leq c\lceil n/5 \rceil + c(7n/10 + 6) + an \\ &\leq cn/5 + c + 7cn/10 + 6c + an \\ &= 9cn/10 + 7c + an \\ &= cn + (-cn/10 + 7c + an) \end{aligned}$$

$$-cn/10 + 7c + an \leq 0 \Leftrightarrow 10a \frac{n}{n-70} \leq c$$