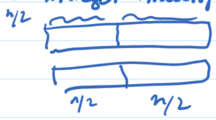
DIVIDE - CONQUER

Tuesday, October 9, 2018 5:10 PM

- Brute force (exhaustive search)
- Transform : (simplification, representation, reduction)
- Decrease : (recursion)
- DIVIDE : solve (multiple) simpler instances of size n/b and recombine the subsolutions

Ex: Fast integer multiplication.



input : 2 n-bit integers A, B

output : AB

$M(1) = 0$

$M(n) = 3M(\frac{n}{2}) + \Theta(n)$, $n = 2^m$

$M(n) = \Theta(n^{\log_2 3 = 1.5\ldots})$

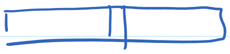
$(A_1 + B_1) \times 10^n$ $\quad A_2 + B_2$

$\left[(A_1 + B_2) + (B_1 B_2) - A_1 B_1 - A_2 B_2 \right] \times 10^{n/2}$

Ex: Binary Search

INPUT : sorted array, x

OUTPUT : true if x is in array

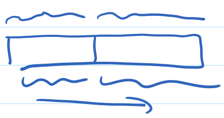


$B(1) = 1$

$B(n) = 1 + B(\frac{n}{2})$, $n = 2^m$

$B(n) = \Theta(n^0 \lg n = \lg n)$

Ex: Merge Sort



$M(1) = 0$

$M(n) = 2M(\frac{n}{2}) + \Theta(n)$ $\quad M(n) = \Theta(n^1 \lg n)$

DIVIDE - CONQUER TEMPLATE

- divide
- recurse (conquer)
- combine

POWER MOD (revisited)

13597

INPUT : nonnegative integers b, e, m ; input size

OUTPUT : $(b^e) \% m$

naive :

```
ans = 1
for (i = 1; i <= e; ++i)
{
    ans = (ans * b) % m;
}
return ans;
```

$\Theta(e) = \Theta(10^{\log_{10} e})$ input size

divide conquer : $b^e \% m = (b^{e/2})^2 \% m$

```
pm (b, e, m)
{
    if (e == 0)
        return 1;
    if (e % 2 == 0)
    {
```

$e = 4$

$b^4 = (b^2)^2$

$b^5 = b \cdot b^4$

return 1;

if (e % 2 == 0)
{
→ temp = pm(b, e/2, m)
return (temp * temp) % m;
}
else
{
temp = pm(b, e/2, m);
return (((temp * temp) % m) * b) % m;
}
}

$b^5 = b \cdot b^4$
$= b \cdot (b^2)^2$

$b^{14} = (b^7)^2$
$b \cdot b^6 = b(b^3)^2$
$b^3 = b \cdot b^2$
$b = b(b^0)^2$

$M(0) = -$
$M(1) = -$
$M(e) = 2 + 1M(\frac{e}{2})$, $e = 2^m$

$a = 1$ $\quad 1 = 2^0$ $\quad M(e) = \theta(e^0 \lg e) = \theta(\lg e)$
$b = 2$
$d = 0$

$\theta(\lg e)$ — e = input size

Ex. QUICK SORT (TONY HOARE)

Divide-conquer:	divide /	conquer /	combine
MERGE	$\theta(1)$	$2M(\frac{n}{2})$	$\theta(n)$
QUICK	$\theta(n)$	$2M(\frac{n}{2})$	0

3 1 4 8 7 6 2 5 — pivot

< pivot | pivot | > pivot

4 2 1 3 | 5 | 7 6 8

1 2 3 4 5 6 7 8

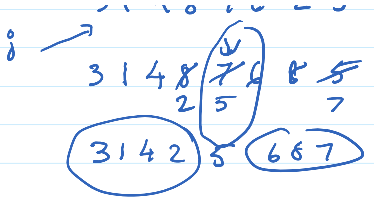
```
qs(A[lo..hi])          n = hi-lo+1
{
    if (lo < hi)
    {
        k = partition(A[lo..hi]);  // k = final POSITION of pivot
        { qs(A[lo..k-1]);           Q(l)
        { qs(A[k+1..hi]);           Q(n-1-l)
    }
}
```

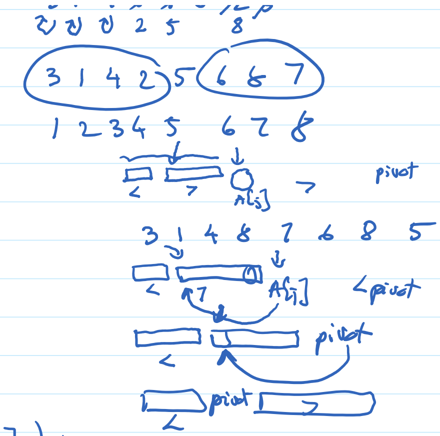
$n-1$, $Q(\ell)$, $Q(n-1-\ell)$

HOW PARTITION WORKS.

i → lo 3 1 4 8 7 6 2 5 hi
j ↗ 3 1 4 8 7 6 8 5

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





partition (A [lo..hi])
{
  i = lo;
  for (j = lo; j < hi; ++j)
  {
    if (A[j] < A[hi])
      swap (A[j], A[i++]);
  }
  swap (A[hi], A[i]);
  return i;
}

ANALYSIS OF PARTITION

P(n): Count # of comparisons of array elements

$$P(n) = \sum_{j=lo}^{hi-1} (1) = hi - 1 - lo + 1 = hi - lo = n - 1, \quad n = hi - lo + 1$$

ANALYSIS OF QUICKSORT

Q(n) = WORST-CASE # of comparisons of array elements

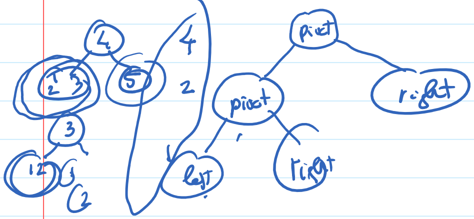
$$Q(1) = 0$$

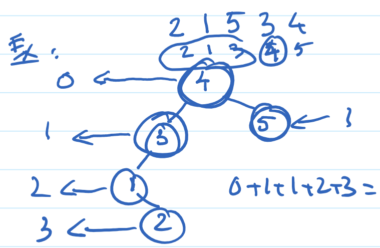
$$Q(n) = \max_{0 \le \ell \le n-1} \left\{ Q(\ell) + Q(n-1-\ell) + (n-1) \right\}, \quad n > 1.$$

How to solve this recurrence ??

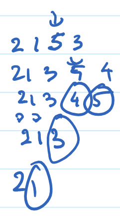
a) Use calculus (messy)

b) Associate each run of QS with a UNIQUE binary tree

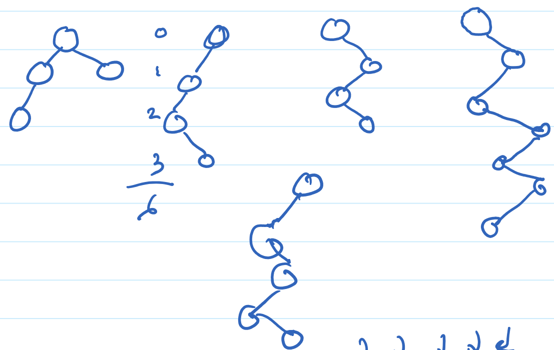




0 + 1 + 1 + 2 + 3 = 7 comparisons

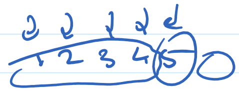


KEY IDEA : # of comparisons made by a run of QS = total sum of depths of the corresponding binary tree



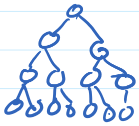
$$0 + 1 + 2 + \cdots + (n-1) = \frac{(n-1)n}{2}$$

$$Q(n) = \frac{n(n-1)}{2}$$



1 2 3 4 = $\underset{\sim}{5(4)}$.

Best case $\Theta(n\lg n)$

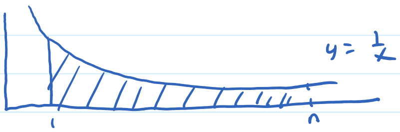


Average Case : (assuming each permutation is equally likely) $\Theta(n\lg n)$

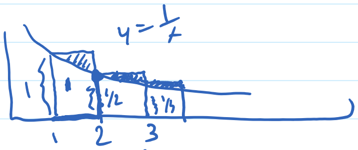
$$\boxed{\text{Avg}(n) = 2(n+1)\,H(n) - 4n}$$

$H(n)$ — n^{th} harmonic number $= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}$

$\ln(n) =$

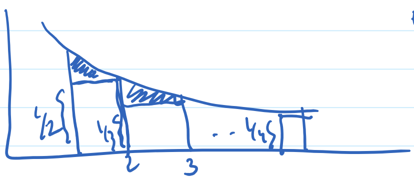


$$\ln(n) = \int_1^n \frac{1}{x}\,dx$$



$\ln(x) \le H(n-1)$

$$\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n-1} = H(n-1)$$



$H(n) - 1 \le \ln n \le H(n-1)$

$H(n) \le 1 + \ln n$

$H(n) \ge \ln(n+1)$

$$H(n) - 1 = \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \le \ln(n)$$

$$\boxed{\ln(n+1) \le H(n) \le 1 + \ln n}$$

BRILLIANT IDEA: use median as pivot

this will lead to

$$Q(n) = 2Q\left(\frac{n}{2}\right) + \Theta(n)$$

if we can find median in linear time

$= \Theta(n\lg n)$!!!

WORST-CASE

smallest	$n-1$
2^{nd} smallest	$n-1+n-2 = 2n-3$
3^{rd} smallest	$n-1+n-2+n-3 = 3n-6$

k^{th} smallest (k is an input): $k(n) - \cdots$

$k = \frac{n}{2}$ $\qquad \frac{n}{2} \cdot n = \frac{n^2}{2}$