DAA UNIT - 2

DECREASE & CONQUER

inductive / incremental approach.

Insertion Sort

- decrease by one
- stable, in place

Worst, avg: $\Theta(n^2)$
Best: $\Theta(n)$ [nearly sorted]

Compare cur ele to max sorted (rightmost). If smaller, shift max sorted by 1 to right and repeat.

```
for i ← 0 to n-1
    v ← A[i]                    (store cur. element)
    j ← i-1                     (find prev. element (greatest sorted))
    while j ≥ 0 and A[j] > v
        A[j+1] ← A[j]           (right shift)
        j ← j-1
    j ← j+1                     (undo the last subtraction)
    A[j] ← v                    (put v in sorted position)
```

Topological Sort (in-order)

topological sort ⇄ dag

↳ for any given vertex, no back edges.

DFS

Do DFS and pop off from stack.
↓
Reverse pop off to get topo sort

Source removal

Look for any vertex with no incoming edges
↓
Remove it and all its edges, record order and repeat

```
DFS(v):
    mark v → visited
    for neighbour u of v
        if not_visited(u)
            DFS(u)
    push v to stk.

Top Sort(G):
    all nodes ← unvisited
    stk = empty
    for each v in G:
        if v not visited:
            DFS(v)
    return reverse(stk)
```

```
L ← list for sorted vertices
S ← set of source vertices
while S is non-empty:
    remove v from S
    add v to tail of L
    for neighbours m with an edge v→m:
        remove edge e
        if m has no other incoming edge
            insert m into S
if graph has edges
    return Error: not a dag
else
    return L
```

GENERATING PERMUTATIONS

(1) Johnson-Trotter's Algorithm

1. put arrows pointing to the left

$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3}$

2. mobile: if no. points to a number less than itself
3. find greatest mobile no.
4. swap it with whatever it points to
5. flip arrows of all elements greater than this mobile element
6. save permutation, repeat as long as last permutation has mobile element

1. $\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3}$
2. $\overleftarrow{1}\ \overleftarrow{3}\ \overleftarrow{2}$
3. $\overleftarrow{3}\ \overleftarrow{1}\ \overleftarrow{2}$
4. $\overleftarrow{3}\ \overleftarrow{2}\ \overleftarrow{1}$
5. $\overleftarrow{2}\ \overrightarrow{3}\ \overleftarrow{1}$
6. $\overleftarrow{2}\ \overleftarrow{1}\ \overrightarrow{3}$

Lexicographic Permute

DECREASE BY CONSTANT FACTOR → solve only one of the subproblems

1. Fake coin problem
   Divide coins into two piles → weigh (unlike divide & conquer)
   $\log_2(n)$ [what if you don't know heavy/light?]

2. Russian Peasant Multiplication
   n, m → input

   n : even
   $$nm = \left(\frac{n}{2}\right)(2m)$$

   n : odd
   $$nm = \left(\frac{n-1}{2}\right)(2m) + m$$

   1 · m = m trivial

n	m	
50	65	
25	130	
12	260	(+130)
6	520	
3	1040	
1	2080	(+1040)

Product = 2080 + 1040 + 130
= 3250

3. Josephus Problem
   Given n,
   Express n as some $2^a + l$
   Then, $S(n) = 2l+1$.

Binary Search.

$l \leftarrow 0$, $r \leftarrow n-1$
while $l \leq r$:
  $m = \lfloor (l+r)/2 \rfloor$
  if $k == A[m]$ return m
  else if $k < A[m]$: $r \leftarrow m-1$
  else: $l \leftarrow m+1$; return -1

$C_{worst}(1) = 1$
$C_{worst}(n) = C\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 1$
$C_w(2^k) = C\left(\frac{2^k}{2}\right) + 1$
$= \left[C\left(\frac{2^k}{2^2}\right) + 1\right] + 1$
$= C\left(\frac{2^k}{2^i}\right) + i$

Let $i = k$
$C(1) + k$
$= k+1$
$= \log_2(n) + 1$

$C_{worst}(n) = \log_2 n + 1$
$C_{Avg}(n) \approx \log_2(n)$

Quick Select + Lomuto Partition : (k^{th} smallest element)

partition:

1. select pivot as rightmost ele (say)
2. initialise $(i = l-1)$, $(j = l)$
3. Compare A[j] with pivot.

   if $A[j] < A[r]$ (pivot)

   $i = i + 1$

   swap $A[j], A[i]$
4. Increment j until r-1
5. Swap pivot A[r] & A[i+1]

   A[l...i] (pivot) A[i+2...r]

   smaller than pivot — greater than pivot.
6. return i+1

Quick Select (A, k): (left, right)

pivotIndex = partition(left, right, A)

if (pivotIndex + 1) = k:

A[pivotInd] → k^{th} smallest

else if (pivotInd + 1) > k:

QuickSelect(A, l, pivotInd - 1, k)

else

QuickSelect(A, pivotInd + 1, r, k)

Best case: Always select median as pivot

$T(n) = T\left(\frac{n}{2}\right) + O(n) \rightarrow O(n)$

Worst case: Biggest/smallest picked.

Median of Medians: divide into groups → find median of each group → find median of medians

O(n) worst case guaranteed. (5)

DIVIDE & CONQUER

Split into subproblems ⟶ solve all subproblems ⟶ combine to get soln for og.

Generally, problem divided by some factor b and a such problems

⟹ Recurrence

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$f(n)$ → dividing prob, combining soln.

Master's Theorem

If $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

and $f(n) \in \Theta(n^d)$ where $d \geq 0$

then:

$a < b^d$	$\Rightarrow T(n) \in \Theta(n^d)$
$a = b^d$	$\Rightarrow T(n) \in \Theta(n^d \log n)$
$a > b^a$	$\Rightarrow T(n) \in \Theta(n^{\log_b a})$

Illy for O, Ω

Merge Sort

split into half recursively and then combine into sorted arrays.

→ copy to some B, C

```
mergeSort (A[0...n-1]):
    if n>1:
        copy A[0...⌊n/2⌋-1] to B[0...⌊n/2⌋-1]
        copy A[⌊n/2⌋...(n-1)] to C[⌊n/2⌋...(n-1)]
        mergeSort(B)
        mergeSort(C)
        return Merge(B, C, A)
    else:
        return A
```

Merge ($B[0 \dots p-1]$, $C[0 \dots q-1]$, $A[0 \dots (p+q-1)]$):

```
i ← 0, j ← 0, k ← 0:
while i < p and j < q:
    if B[i] < C[j]:
        A[k] = B[i]
        i += 1; k += 1;
    else
        A[k] = C[j]
        j += 1; k += 1;
while i < p:
    A[k] = B[i]
    i += 1; k += 1;
while j < q:
    A[k] = C[q];
    j += 1, k += 1;
return A
```

Time Complexity

$C(n) = 2C\left(\frac{n}{2}\right) + C_{merge}(n)$

$C(1) = 0$

$C_{merge}(n) = n-1$

$\Rightarrow C(n) = 2C\left(\frac{n}{2}\right) + n-1$

$d = 1$
$a = 2$ $\quad a = b^d$
$b = 2$

$\Rightarrow C(n) \in \Theta(n \log n)$

Quick Sort + Hoare's Partition

```
quicksort (A[0 ... n-1])
if (l <) Hoare Part (A)
    i = Hoare Part (A)
    quick Sort (A[0 ... i])
    quicksort (A[i+1 ... n-1])
```

Time Complexity.

$C_{best}(n) = 2C_b\left(\frac{n}{2}\right) + n$

$d = 1, \quad a = 2, \quad b = 2$

$a = b^d$

$\Rightarrow C_{best}(n) \in \Theta(n \log n)$

$C_{avg} \approx 2n \log n$
$\approx 1.38 n \log n$
$\approx \log n$

```
HoarPart (A[l ... r]):
    pivot ← A[l]
    i ← l; j ← r+1
    repeat
        repeat i ← i+1 until A[i] ≥ p
        repeat j ← j-1 until A[j] ≤ p
        swap A[i], A[j]
    until i ≥ j
    undo swap A[i], A[j] = A[j], A[i]
    swap pivot, A[j]
    return j
```

$C_{worst} = \frac{(n+1)(n+2)}{2} - 3 \in \Theta(n^2)$

$= \sum_{0}^{n+2} n - 3$

$C_{worst} \in \Theta(n^2)$

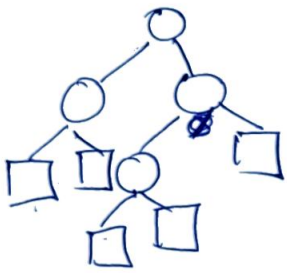
(7)

Binary trees

Height (T):

if $T = \phi$ return -1

return $\max\{\text{height}(T_L), \text{height}(T_R)\} + 1$



No of comparisons (empty or not) $= x + n = 2n + 1$

No of additions $= n$

□ → external nodes x

$x = n + 1$

Traversal

inorder: L V R

preorder: V L R

postorder: L R V

Multiplication of large numbers

Say we want $a.b$.

a can be split into a_1 (first half), a_0 (second half)

$$a = a_1 \times 10^{\frac{n}{2}} + a_0$$

$$b = b_1 \times 10^{\frac{n}{2}} + b_0$$

Now we have

$$c = (a_1 \times 10^{\frac{n}{2}} + a_0)(b_1 \times 10^{\frac{n}{2}} + b_0)$$

$$= (a_1 b_1) 10^n + (a_1 b_0 + a_0 b_1) 10^{\frac{n}{2}} + (a_0 b_0)$$

$c_0 = a_0 \times b_0$

$c_2 = a_1 \times b_1$

$c_1 = (a_1 + a_0)(b_1 + b_0) - (c_1 + c_0)$

$C = c_2 \times 10^n + c_1 \times 10^{\frac{n}{2}} + c_0$

$A(n) = 3A\left(\frac{n}{2}\right) + cn$

$A(n) \in \Theta(n^{\log_2 3})$

$M(n) = 3M\left(\frac{n}{2}\right)$ $\quad M(1) = 1$

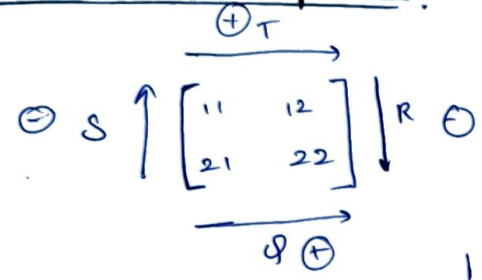
$M(2^k) = 3M(2^{k-1}) = 3^2 M(2^{k-2}) = 3^i M(2^{k-i})$

put $i = k$

$\Rightarrow 3^k = 3^{\log_2 n}$

$M(n) = 3^{\log_2 n} = n^{\log_2 3}$ $\quad M = n^{1.585}$

Strassen's Matrix Multiplication



time complexity

$n^{\log_2 7}$

$7M\left(\frac{n}{2}\right) \quad M(1)=1$

$7A\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2$

diag

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = B_{11}(A_{21} + A_{22})$

$R = A_{11}(B_{12} - B_{22})$

$S = A_{22}(B_{21} - B_{11})$

$T = B_{22}(A_{11} + A_{12})$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (B_{21} + B_{22})(A_{12} - A_{22})$

$B_{11}\ A_{11}\ A_{22}\ B_{22}$

C_{ij} :

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

P
R A T
V U