comparison
or +=O(n log n)

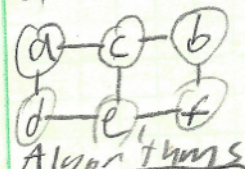
Euclid Alg
$gcd(m,n) = gcd(n, m \bmod n)$ until $n=0$

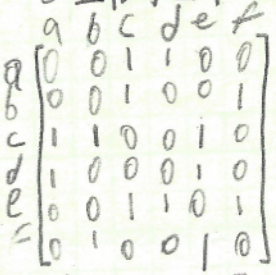
BST starts with middle num

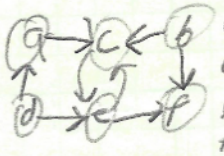
TSP
Find all combinations, calculate each

Graph Def
- A graph $G = \langle V, E \rangle$ = Finite, non-empty set V
Set E = (sets cannot have dupes)

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$, $0 \le |E| \le |V|(|V|-1)/2$







Algorithms
$1 > \log n > n > n \log n > n^2 > n^3 > 2^n > n!$

Big Oh - Set of all functions with lower/same order of growth n ($n \in O(n^2)$)
Big omega - Set of all func with higher/same ($n^3 \in \Omega(n^2)$)
Big theta - Set of all with same ($n^3 \notin \Theta(n^2)$ & $n \notin \Theta(n^2)$)
($2n^2 \in \Theta(n^2)$)

Recursion:
$M(n)$: Number of multiplications executed for the input number n.

Algorithm F(n)
if n=0
return 1
else return F(n-1)·n

$M(0) = 0$ (initial)
$M(n) = M(n-1) + 1$ (recur. relation)

$M(n) = M(n-1) + 1$
$= [M(n-2+1)] + 1$
$= M(n-2) + 2$
$M(n) = M(n-2) + 2$
$[M(n-3+1)] + 2$
$M(n-3) + 3$
$= M(n-i) + i \quad | i \in \mathbb{Z}$
$= \dots$
$= M(0) + n$
$= 0 + n$
$= n \in \Theta(n)$

$M(n)$ is the amount of multiplications for each n.

if n=1
return (2 mults)
else
return $S(n-1) + n \cdot n \cdot n$

$M(n-1) + 2$
$= M(n-i) + 2i \quad (i = n-1)$
$= M(1) + 2 \cdot (n-1)$
$= 0 + 2(n-1)$
$\in \Theta(n)$

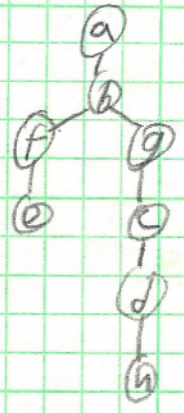
Uses Stack

Mark array to track visited vertices

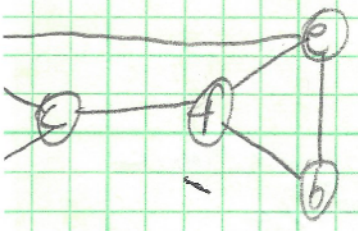
Mark Array

a	1
b	2
c	6
d	7
e	4
f	3
g	5
h	8

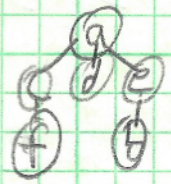
DFS Tree

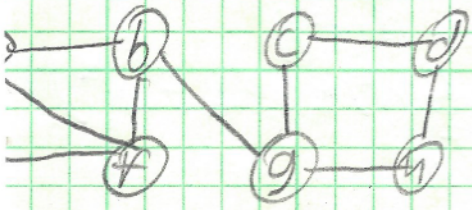


Uses Queue $\Theta(|V|^2)$

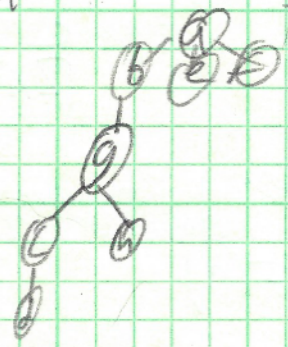


a → c → d → e → f → b





a → b → e → f → g → c → h → d



te Force

ction sort

```
int i, j
int aLength;
for (i=0; i<aLength-1; i++) {
    int jMin = i;
    for (j = i+1; j<aLength; j++) {
        if (a[j] < a[jMin]) {
            jMin = j;
        }
    }
    if (jMin != i) {
        swap(a[i], a[jMin])
    }
}
```

Bubble Sort

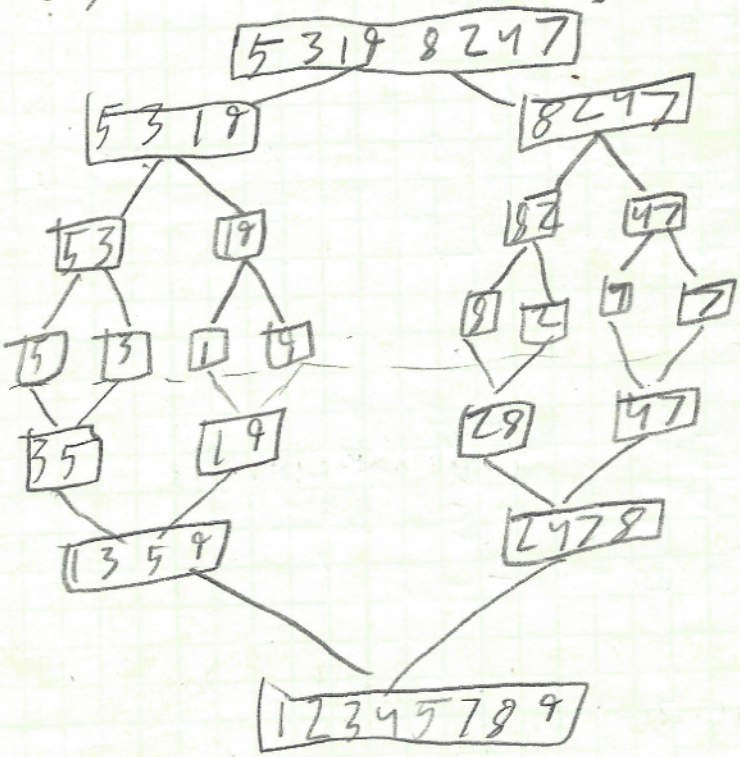
```
func BubbleSort(A[])
    n = length(A)
    while(swapped == false)
        for (i = 1 to n-1)
            if A[i-1] > A[i]
                Swap(A[i-1], A[i])
                swapped = true
        n = n-1
```

OK

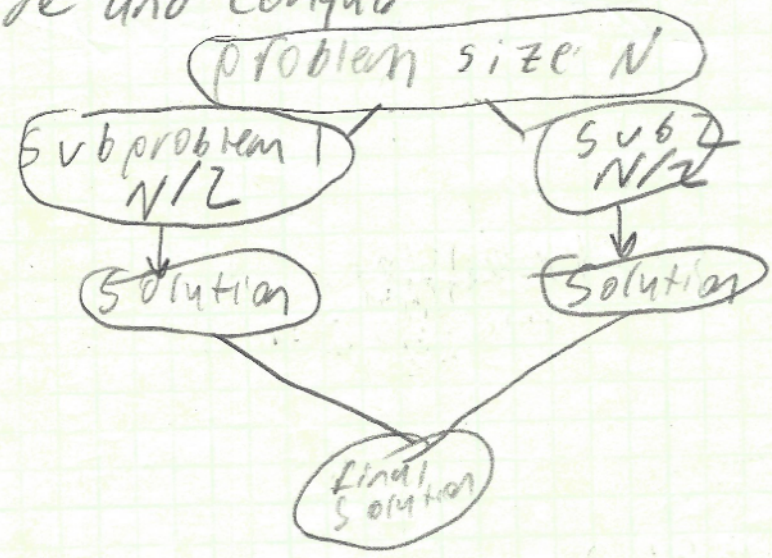
```
for i=0 to n-2 do
    for j=0 to n-2 do
        if A[j+1] < A[j]
            Swap(A[j], A[j+1])
```

Text NOBODY
NOT B ≠ T
NOT O ≠ N
NOT B ≥ N

Merge Sort - O(n log(n))



Divide and Conquer



ster theorem
b = number of subproblems
a = number of subproblems to be solved
d = is the number in $\theta(n^d)$ in which $\theta(n^d)$ is the time to divide & conquer the original problem of size n.

```
int sum(A[], start, end)
    if (start == end)
        return A[start]
    else
        int sum1 = sum(A, start, (start+end)/2)
            sum2 = sum(A, (start+end)/2, end)
        return(sum1+sum2)
```

b = 2
a = 2
d = 0

Case 1 if $a < b^d$ then $\theta(n^d)$
case 2 if $a = b^d$ then $\theta(n^d \cdot \log n)$
case 3 if $a > b^d$ then $\theta(n^{\log_b a})$

$T(n) = 2 \cdot T(n/2) + 5n^3 + 7n$

a = 2
b = 2
d = 3 ($5n^3$)

$2 < 2^3$, therefore $\theta(n^3)$