# CS 170 Final

## Euclid's GCD: $O(n^3)$

```
def gcd(a,b):
    if b==0:
        return a
    return gcd(b, a mod b)
```

## Extended GCD: $O(n^3)$

```
def extended-gcd(a,b):
    if b==0:
        return (1, 0, a)
    (x', y', d) = extended-gcd(b, a mod b)
    return (y', x' - floor(a/b)*y', d)
```

if d divides a and b and $d = ax + by$ for some integers s and y, then d = gcd(a, b)

## Multiplicative Inverse

inverse of a,

$$ax \equiv 1 (\texttt{mod}\ N)$$

for any $a(\texttt{mod}\ N)$, a has a multiplicative inverse if and only if they are relatively prime, gcd(a,N) = 1

## Fermat's Little Theorem

given a prime (or carmichael) p,

$$a^{p-1} \equiv 1 (\texttt{mod}\ p$$

## RSA Euler's Theorem

$$m^{(p-a)(q-1)} = 1(\texttt{mod}\ p)$$

## Master's Theorem

If

$$T(n) = aT(\lceil n/b \rceil) + O(n^d) \texttt{ for } a > 0, b > 1, \texttt{ and } d \geq 0,$$

then,

$$T(n) = \begin{cases} O(n^d) & if d > log_b a \\ O(n^d logn) & if d = log_b a \\ O(n^{log_b a}) & if d < lob_b a \end{cases}$$

## Fast Fourier Transform

Todo Add FFT Information Here

## Depth First Search

```
def explore(G,v): #Where G = (V,E) of a Graph
    visited(v) = true
    previsit(v)
    for each edge(v,u) in E:
        if not visited(u):
            explore(u)
    postvisit(v)

def dfs(G):
    for all v in V:
        if not visited(v):
            explore(v)
```

Previsit = count till node added to the queue
Postvisit = count till you leave the given node
A directed Graph has a cycle if it has a back edge found during DFS

## Directed Acyclic Graphs

Every DAG has a source and sink
Todo add more properties

## Greedy Algorithms

### Kruskal's MST Algorithm

Repeatedly add the next lightest edge that doesn't produce a cycle.

### Properties of Trees (undirected acyclic graphs)

- A tree with n nodes has n-1 edges
- Any connected undirected graph G(V,E), with $|E| = |V| - 1$ is a tree
- An undirected graph is a tree if and only if there is a unique path between any pair of nodes.

### Cut Property

Suppose edges X are part of a minimum spanning tree of $G = (V, E)$. Pick any subset of nodes S for which X does not cross between S and V-S, and let e be the lightest edge across the partition. Then $X \cup e$ is part of some Minimum Spanning Tree.

### Prim's Algorithm

(an alternative to Kruskal's Algorithm and similar to Dijkstras)
On each iteration, the subtreedefined by x grows by one edge, the lightest between a vertex in S and a vertex outside S.

### Huffman Encoding