COP 3502 9/19/23

Recursion!

- function that sometimes calls itself

$n! = 1 \times 2 \times 3 \times 4 \times \ldots \times n$, $0! = 1$

$n! = (1 \times 2 \times 3 \times \ldots \times (n-1)) \times n$

$n! = (n-1)! \times n$, $1! = 1$

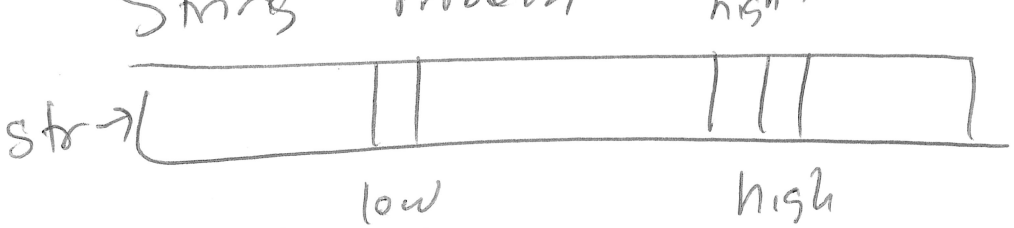
recursive def — base case

```
int fact(int n) {
    if (n<=1)
        return 1;
    return fact(n-1) * n;
}
```

$b^e = b^{e-1} \times b^1 = b^{e-1+1} = b^e$

just b

String Problem



question how many times does c appear in str[low..high]

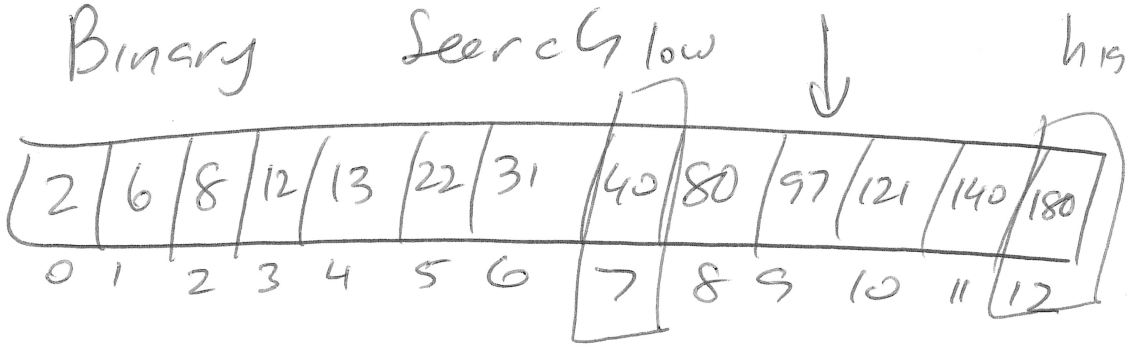
count(# c in low to high-1) +

0 (if str[high] != c) → ~~0~~

1 (if str[high] == c)

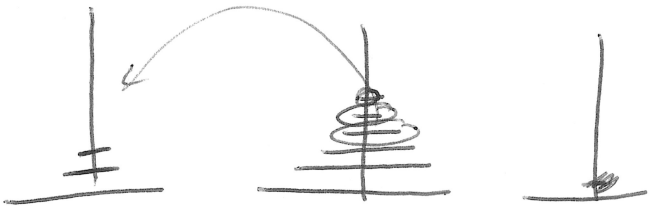
equals 1 when true
equals 0 when false

Binary Search




```
int binsearch(int * array, int low, int high, int searchval){
    if (low > high) return 0;
    ~~int~~ int mid = (low+high)/2;
    if ( searchval < array[mid])
        return binsearch(array, low, mid-1, searchval);
    else if ( searchval > array[mid])
        return binsearch(array, mid+1, high, searchval);
    else return 1;
}
```

Towers of Hanoi



1 2 3

n disks on one pole (largest bottom
move 1 disk at a time
top disk + place on either of the other sticks/poles, provided
(a) nothing on it
(b) it's smaller than top disk on that pole

Goal move all disks from tower start = 2 to tower end = 3.

→ Can't solve puzzle w/o moving bottom disk. ⟶ ONLY way to move bottom is to move all n-1 disks from start to "mid" tower

Towers (n, start, end)

1. Towers (n-1, start, mid)
2. Move Disk # n from start to end
3. Towers (n-1, mid, end)

Fast Mod Expo

Calculate b^e o/o m

fastmodexpo (long long base, long long exp,
long long mod)

---

slowmodexpo (~~~) {

long long res = 1;

for (long long i = 0; i < exp; i++)

res = (res * base) o/o mod;

return res;

}

Run time O(exp)

value of

$b^{1000000} = (b^{500000})^2$

tmp = $b^{500,000}$

return (tmp * tmp) o/o mod

```
if (expo%2 == 0) {
    long long tmp = fastmodexpo(base, expo/2, mod);
    return (tmp*tmp) % mod;
}
return (base * fastmodexpo(base, exp-1, mod)) % mod;
```

10^6
↓
500,000
↓
250,000
↓
125,000
↓
62,500
↓
31,250
↓
15,625
↓
15,624
↓
7812

2 steps or exp by 2

$$\frac{exp}{2^k} = 1$$

$$exp = 2^k$$

$$k = \log_2 exp$$

more steps

$\leq 2 \times \log_2 exp$

$O(\lg exp)$