# CSC 212: Data Structures and Abstractions

## Analysis of Recursive Algorithms

Marco Alvarez

Department of Computer Science and Statistics
University of Rhode Island

Fall 2020

THINK BIG WE DO℠

# Analysis of Binary Search

```
int bsearch(int *A, int lo, int hi, int k) {
    if (hi < lo) {
        return NOT_FOUND;
    }
    int mid = lo + ((hi-lo)/2);
    if (A[mid] == k) return mid;
    if (A[mid] < k) return bsearch(A, mid+1, hi, k);
    return bsearch(A, lo, mid-1, k);
}
```

Base Case: $T(1) = c_0$

Recursive Case: $T(n) = T(n/2) + c_1$

# Recurrence relations

- By itself, a recurrence does not describe the running time of an algorithm
  - need a **closed-form solution** (non-recursive description)
  - exact closed-form solution may not exist, or may be too difficult to find
- For most recurrences, an asymptotic solution of the form $\Theta()$ is acceptable
  - … in the context of analysis of algorithms

# How to solve recurrences?

- By **unrolling** (expanding) the recurrence
  - a.k.a. **iteration method** or repeated substitution
- By **guessing** the answer and proving it correct **by induction**
- By using a **Recursion Tree**
- By applying the **Master Theorem**

# Unrolling a Recurrence

- Keep unrolling the recurrence until you identify a general case
  - then use the base case

- Not trivial in all cases but it is helpful to build an intuition
  - may need induction to prove correctness

# Unrolling the binary search recurrence

$$T(1) = c_0 \qquad\qquad T(n) = T(n/2) + c_1$$

# Applying the base case

We already know T(1) is equal to a constant $c_0$:

$$= T(n/2^k) + kc_1$$

# Example 1

```
int power(int b, int n) {
    if (n == 0) {
        return 1;
    }
    return b * power(b, n-1);
}
```

Can you write (and solve) the recurrence?

# Example 1

$$T(0) = 0$$

$$T(n) = T(n-1) + 1$$

```
int power(int b, int n) {
    if (n == 0) {
        return 1;
    }
    return b * power(b, n-1);
}
```

# Example 2

$$T(1) = a$$

$$T(n) = 2T(n/2) + n$$

# Example 3

$$T(0) = 1$$

$$T(n) = 2T(n-1) + 1$$

# Find the **max** (strongly unimodal)

- Running Time?

# Find the **max** (weakly unimodal)

- Running Time?

## Series

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

In general:

$$\sum_{i=1}^{n} i^m = \frac{1}{m+1}\left[(n+1)^{m+1} - 1 - \sum_{i=1}^{n}\left((i+1)^{m+1} - i^{m+1} - (m+1)i^m\right)\right]$$

$$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k} B_k n^{m+1-k}.$$

Geometric series:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad |c| < 1,$$

$$\sum_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad |c| < 1.$$

Harmonic series:

$$H_n = \sum_{i=1}^{n} \frac{1}{i}, \qquad \sum_{i=1}^{n} iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$$

$$\sum_{i=1}^{n} H_i = (n+1)H_n - n, \quad \sum_{i=1}^{n}\binom{i}{m}H_i = \binom{n+1}{m+1}\left(H_{n+1} - \frac{1}{m+1}\right).$$