CSC 212

Data Structures & Algorithms

Fall 2022 | Jonathan Schrader

Recurrences

Housekeeping

Review Project [MEC]

- Due October 24, 11:59pm
- Walkthrough video

Factorial of n (formula)

```
int fact(int num) {
  if (num == 0)
    return 1;
  else
    return num * fact(num - 1);
}
```

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

$$\sum_{i=1}^{n} i = 1 \qquad \textit{for all } n >= 1$$

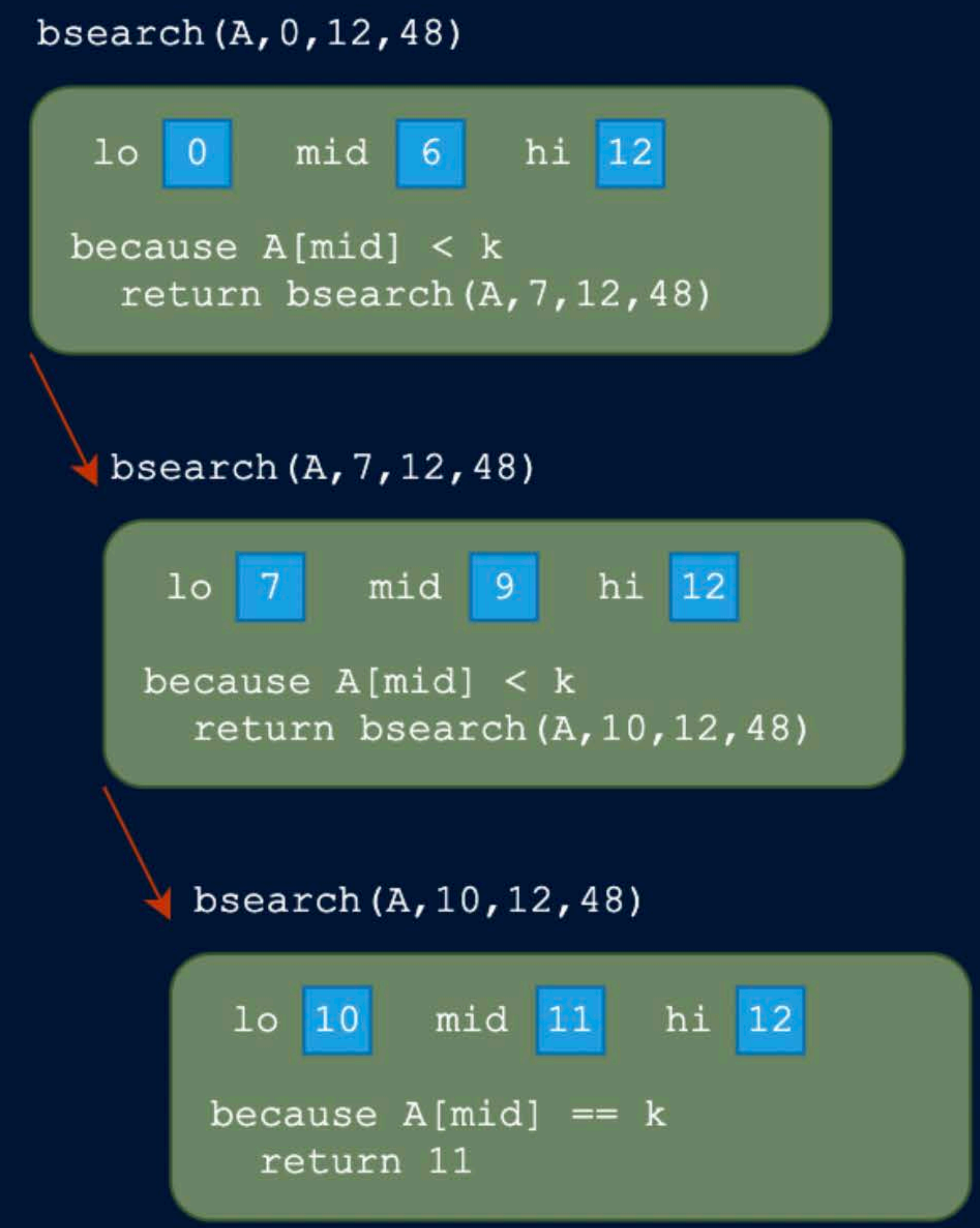
$$n! = n * (n-1)!$$

Analysis of Binary Search

```c
int bsearch(int *A, int lo, int hi, int k) {
  //base case
  if (hi < lo)
    return NOT_FOUND;

  // calculate mid point index
  int mid = lo + ( (hi - lo) / 2);
  // key found?
  if (A[mid] == k)
    return mid;
  // key in upper subarray?
  if (A[mid] < k)
    return bsearch(A, mid + 1, hi, k);
  // key is in lower subarray?
  return bsearch(A, lo, mid - 1, k);
}
```

$$t(n) = \begin{cases} 1 & if\, n = 1 \\ T(n/2) + n & if\, n > 1 \end{cases}$$



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

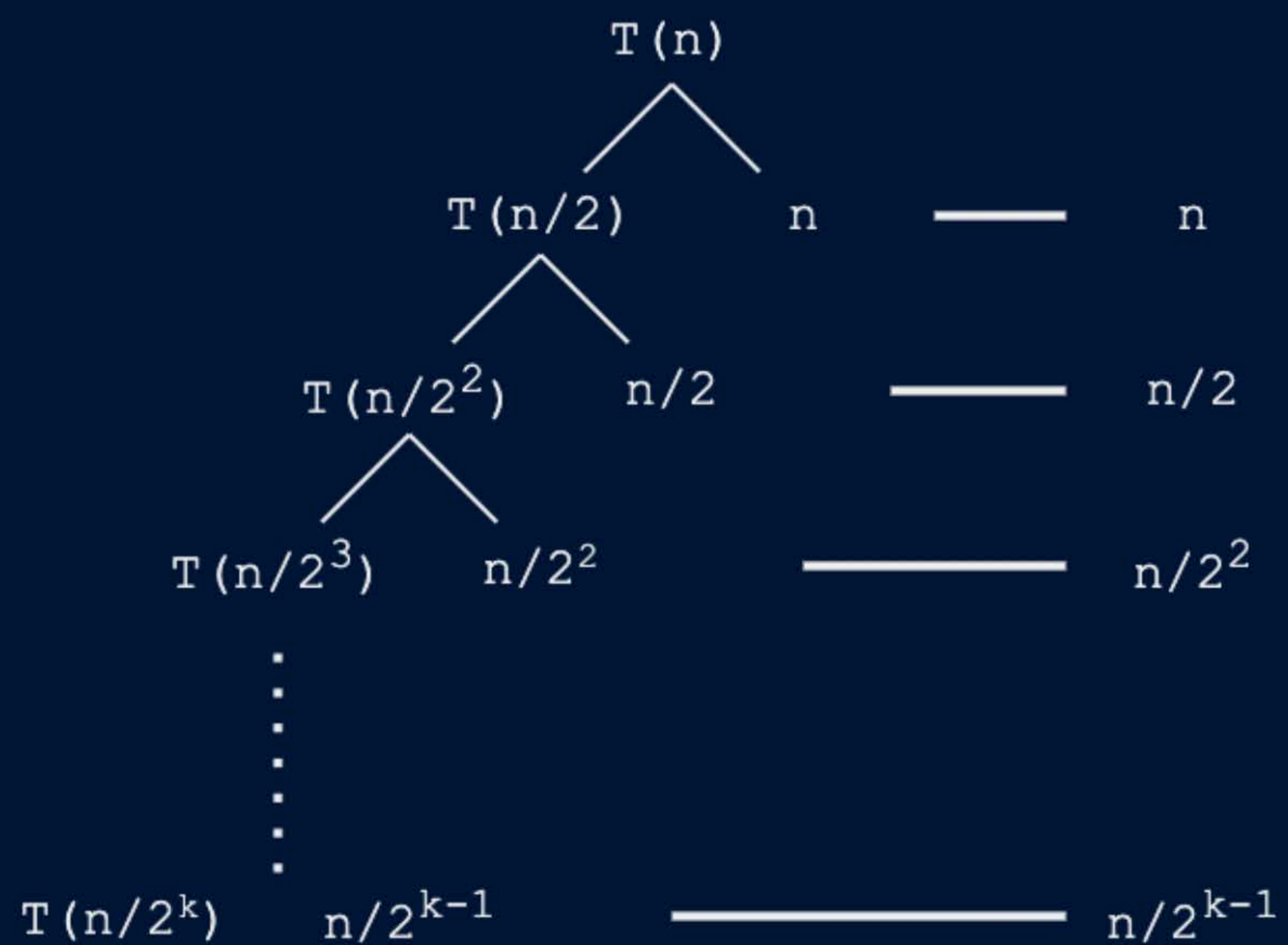
- then use the base case

Not trivial in all cases but it is helpful to build an intuition

- may need induction to prove correctness

Unrolling the binary search recurrence

$$t(n) = \begin{cases} 1 & if\, n = 1 \\ T(n/2) + n & if\, n > 1 \end{cases}$$



$$T(n) = n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \frac{n}{2^k}$$

$$T(n) = \left[1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{2^k}\right]$$

$$\sum_{i=0}^{k} \frac{1}{2^i} = 1$$

$$n * 1$$

$$T(n) = n$$

$$\Theta(n)$$

Example 1: powers

```
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: powers, con't

$$t(n) = \begin{cases} 1 & if\ n = 0 \\ T(n-1)+1 & if\ n > 0 \end{cases}$$

Breakdown

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

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

$$T(n-2) = T(n-3)+1$$

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

Substitution $T(n-1)$

$$T(n) = \Big[T(n-2)+1\Big]+1$$

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

$$T(n) = \Big[T(n-3)+1\Big]+2$$

$$T(n) = T(n-3)+3$$

For k times…

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

Assume $n-k=0 \ldots n=k$

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

$$T(n) = T(0)+n$$

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

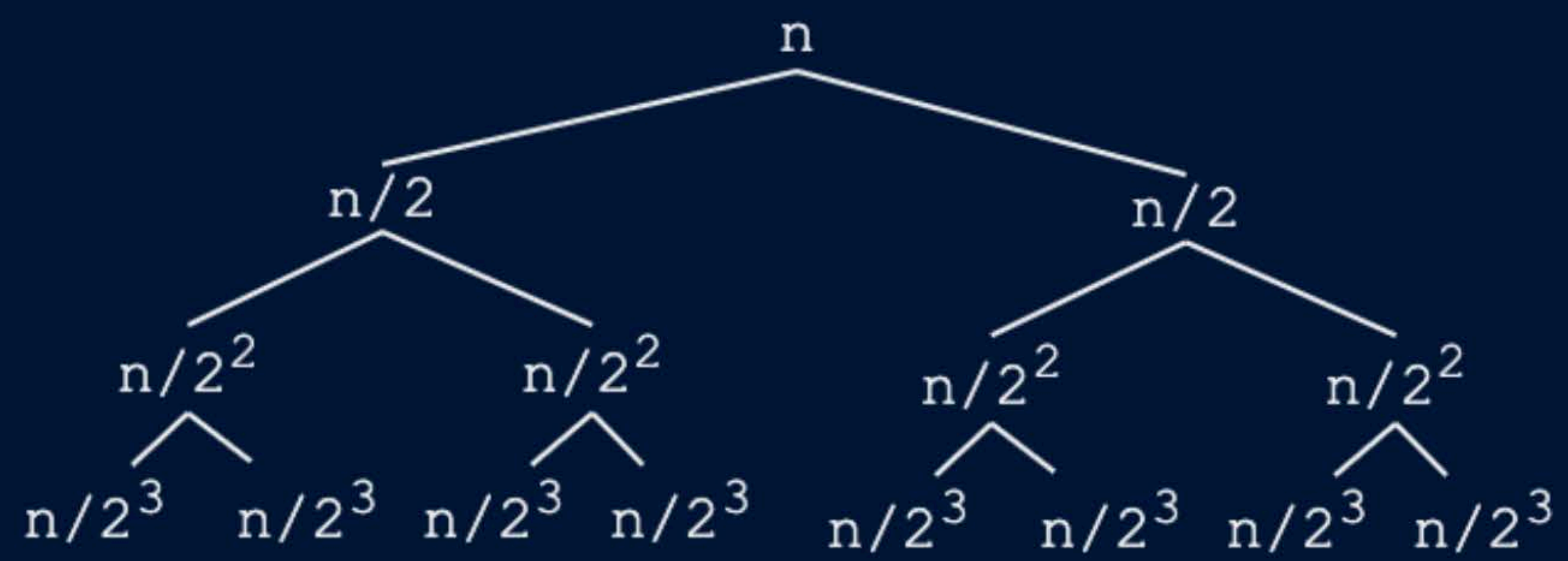
$$T(n) = \Theta(n)$$

$$\Theta(n)$$

Example 2: merge sort

$$t(n) = \begin{cases} 1 & if\, n = 0 \\ 2T(\frac{n}{2}) + n & if\, n > 0 \end{cases}$$

Tree Method



Continues for k times

Therefore, n^k

O(n log n)

Example 2: merge sort, con't

Substitution

$$2T(\frac{n}{2}) + n$$

$$2T(\frac{n}{2^2}) + \frac{n}{2}$$

$$2\left[2T(\frac{n}{2^2}) + \frac{n}{2}\right] + n$$

$$2^2T(\frac{n}{2^2} + n + n)$$

$$T(\frac{n}{2^2} = 2T(\frac{n}{2^3}) + \frac{n}{2^2})$$

$$2\left[2T(\frac{n}{2^3}) + \frac{n}{2^2}\right] + 2n$$

$$T(n) = 2^3T(\frac{n}{2^3}) + 3n$$

$$T(n) = 2^kT(\frac{n}{2^k}) + kn$$

Assume

$$T(\frac{n}{2^k}) = T(1)$$

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

Therefore, $n = 2^k$

$$k = log\, n$$

$$T(n) = 2^kT(1) + kn$$

$$= n * 1 + n\, log\, n$$

$$\Theta(n\, log\, n)$$

Example 3: tower of hanoi

$$t(n) = \begin{cases} 1 & if\ n = 0 \\ 2T(n-1)+1 & if\ n > 0 \end{cases}$$

Substitution

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

$$= 2\left[2T(n-2)+1\right]+1$$

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

$$= 2^2\left[2T(n-3)+1\right]+2+1$$

$$T(n) = 2^3T(n-3)+2^2+2+1$$

For k times…

$$T(n) = 2^kT(n-k)+2^k-1+2^k-2+\ldots 2^2+2+1$$

Assume $n-k=0$

$$n = k$$

$$2^nT(0)+1+2+2^2+\ldots 2^k-1$$

$$2^n-1+2^k-1$$

$$2^n+2^n-1$$

$$2^n+1-1$$

$$\Theta(2^n)$$