THINK BIG WE DO™

CSC 212
Data Structures & Algorithms

Fall 2022 | Jonathan Schrader

Recurrences

Factorial of n (formula)

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

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

Use Cases

Permutation

- gives the number of ways to select r elements from n elements when *order matters*

$${}^nP_r = \frac{n!}{(n-r)!}$$

Combination

- gives the number of ways to select r elements from n elements where *order does not matter*

$${}^nC_r = \frac{n!}{r!(n-r)!}$$

Analysis of Binary Search

```
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);
}
```

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

```
bsearch(A,0,12,48)
  lo 0   mid 6   hi 12
  because A[mid] < k
    return bsearch(A,7,12,48)

bsearch(A,7,12,48)
  lo 7   mid 9   hi 12
  because A[mid] < k
    return bsearch(A,10,12,48)

bsearch(A,10,12,48)
  lo 10   mid 11   hi 12
  because A[mid] == k
    return 11
```

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

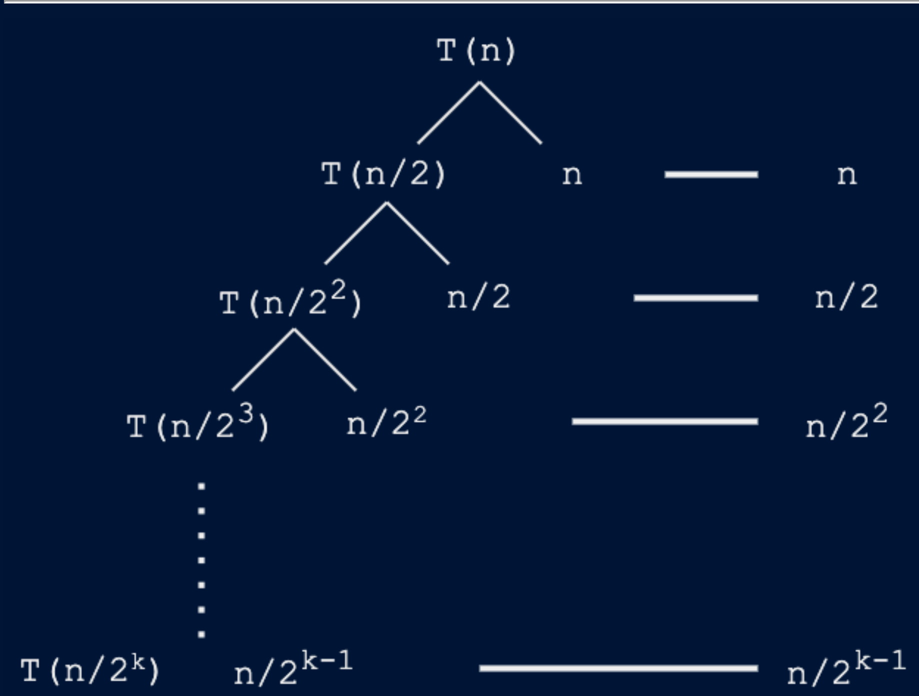
- then use the base case

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

- may need induction to prove correctness

Unrolling the binary search recurrence

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



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

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

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

$$n * 1$$

$$T(n) = n$$

$$\Theta(n)$$

Recurrence Relation: Linear

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

Can you write (and solve) the recurrence?

Recurrence Relation: Linear, con't

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

Breakdown

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

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

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

$$\vdots$$

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

Substitution $T(n-1)$

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

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

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

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

For k times

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

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

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

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

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

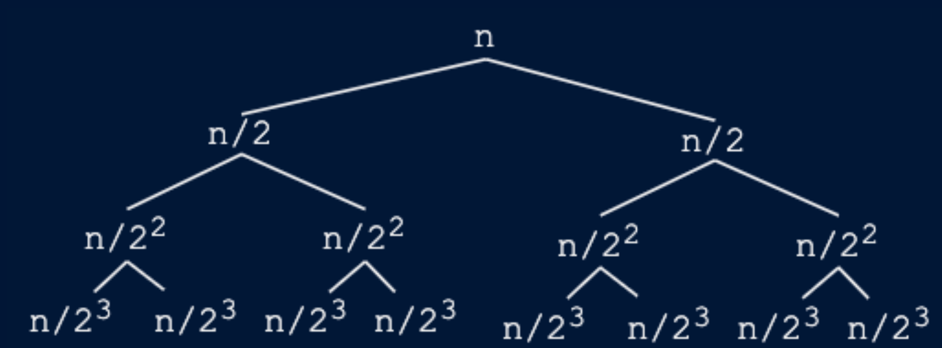
$$T(n) = \Theta(n)$$

$$\Theta(n)$$

Recurrence Relation: Divide & Conquer

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

Tree Method



$$For\ k\ times\ = \frac{n}{2^k}\ => \ n = 2^k \ldots Hence,\ k\ = log\ n$$

$$T(n) = O(n\ log\ n)$$

Recurrence Relation: Divide & Conquer, con't

Substitution

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

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

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

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

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

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

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

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

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

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

$$Therefore\ldots\ n = 2^k$$

$$k = log\, n$$

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

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

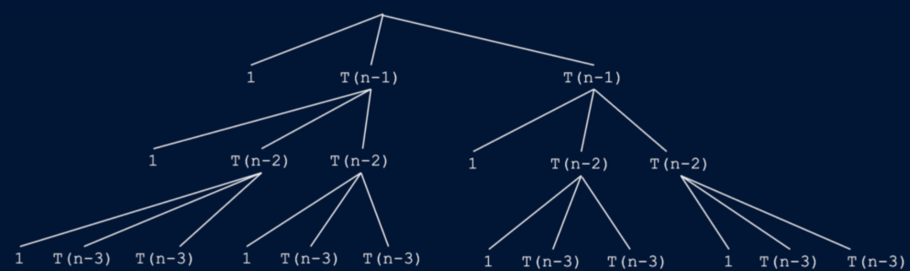
$$\Theta(n\, log\, n)$$

THINK BIG WE DO

Recurrence Relation: Tower of Hanoi

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

Tree Method



Breakdown

$$1 + 2 + 2^2 + 2^3 ... + 2^k = 2^{k+1} - 1$$

$$ar + ar + ar^2 + ar^3 ... + ar^k = \frac{a(r^{k+1} - 1)}{r - 1}$$

$$Where...\ a = 1,\ r = 2$$

$$\frac{1(2^{k+1}1)}{2-1} = 2^{k+1} - 1$$

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

$$n = k$$

$$= 2^{k+1} - 1 => 2^{n+1} - 1$$

$$= O(2^n)$$

Recurrence Relation: Tower of Hanoi, con't

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

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 $n-k=0$

$$n = k$$

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

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

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

$$2^n+1-1$$

$$\Theta(2^n)$$

Resources

Just in case the links embedded in the individual slides aren't working…

Slide - Subject - Link

3 - Factorial of n (formula) | https://byjus.com/maths/factorial/#formula)

5 - Recurrence relations | https://www.math.wichita.edu/discrete-book/ch_sequences.html

6 - How to solve recurrences? | https://courses.engr.illinois.edu/cs473/sp2010/notes/99-recurrences.pdf

6.1 - *guessing* | https://www.geeksforgeeks.org/method-of-guessing-and-confirming/

6.2 - *Master Theorem* | http://homepages.math.uic.edu/~leon/cs-mcs401-s08/handouts/master_theorem.pdf

7 - Unrolling a Recurrence | https://courses.cs.washington.edu/courses/cse332/18su/handouts/unrolling.pdf

8 - Unrolling the binary search recurrence | https://youtu.be/XcZw01FuH18

9 - Example 1: powers | https://youtu.be/4V30R3I1vLI

11 - Example 2: merge sort | https://youtu.be/1K9ebQJosvo

13 - Example 3: tower of hanoi | https://youtu.be/JvcqtZk2mng