Analysis of Recursive Algorithms

Warmup: what is the runtime of the following algorithm?

for $i = 1$ to n: $\leftarrow n(c + \underline{\ ?\ })$
  if $3 | i$:
    foo(n) $\leftarrow n$

outside-in: $f(n) = n(c + \frac{1}{3}n) = nc + \frac{n^2}{3} = \Theta(n^2)$

inside-out: $\frac{n}{3} n + nc = \Theta(n^2)$

Assume foo(n) takes n primitive operations.

Remember, $3|i$ means "3 divides i"

Problem: factorial
  input: $n \in \mathbb{Z}^{>0}$
  output: $n! = n(n-1)(n-2)\cdots 3 \cdot 2 \cdot 1$

Solution:

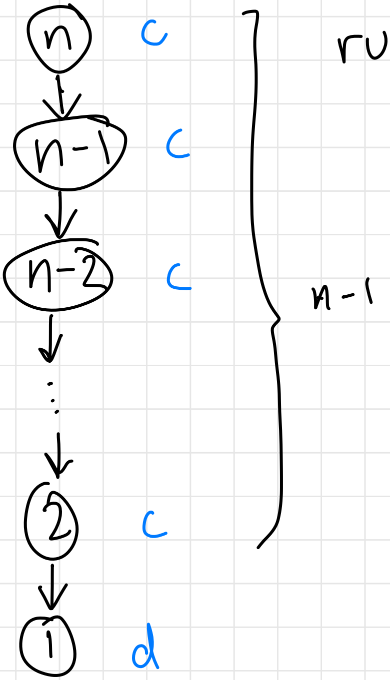
fact(n): ← name of alg
  if n = 1 then
    return 1 } d
  else
    return n · fact(n−1) } c

What is the runtime of fact?

idea #1: look at the recursion tree

Def The <u>recursion tree</u> for an alg. A is a tree that shows all of the recursive calls spawned by A on an input of size n.

recursion tree for fact(n):



runtime fact(n):
total runtime of all nodes

$$f(n) = (n-1)c + d$$
$$= cn - c + d$$
$$= \Theta(n)$$

idea #2: Use a recurrence relation.

Def A <u>recurrence relation</u> is a function $T(n)$ that is defined in terms of values of $T(k)$ for $k<n$.

ex for fact(n), the runtime $T(n)$ is

$T(1) = d$, $T(n) = \underline{c} + \underline{T(n-1)}$

base case, recursive case, $n>1$

fact(n):
  if $n=1$ then
      return 1 } d
  else
      return $n \cdot \underbrace{\text{fact}(n-1)}$ } c

But we need a closed-form solution for $T(n)$.

① Iterate the solution a few times
② make a guess about $T(n)$
③ try to prove

①
$T(1) = d$
$T(2) = c + T(1) = c + d$ $n=2$
$T(3) = c + T(2) = c + (c+d) = 2c + d$ $n=3$
$T(4) = c + T(3) = c + 2c + d = 3c + d$ $n=4$

② Guess: $T(n) = (n-1)c + d$

③ Prove

$P(n)$

<u>Claim</u> The recurrence relation ~~defined~~ by $T(1) = d$ and $\boxed{T(n) = c + T(n-1)}$ has closed-form solution $T(n) = (n-1)c + d$.

<u>Proof</u> We prove the claim using mathematical induction.

Base case: when $n=1$, $T(1) = d$ by def. of recurrence relation.
$T(1) = (1-1)c + d = d$ as well, so the

base case holds.

Inductive case: We WTS

$\forall n \geq 2$: $P(n-1) \Rightarrow P(n)$.

Assume $P(n-1)$; that is, $T(n-1) = c(n-2) + d$.

WTS $P(n)$; that is, $T(n) = c(n-1) + d$.

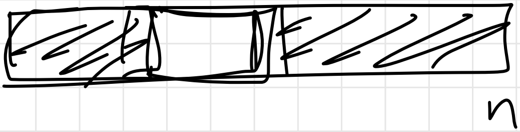
$$
\begin{aligned}
T(n) &= c + T(n-1) && \leftarrow \text{def. of our rec. rel.} \\
&= c + c(n-2) + d && \leftarrow \text{by IH} \\
&= c + cn - 2c + d \\
&= cn - c + d \\
&= (n-1)c + d
\end{aligned}
$$

So $P(n)$ holds.

Another recursive alg. assume A is sorted

Algorithm 1 binarySearch(A[1...n], x)

if $|A| = 0$ **then**
  return False
else
  $middle = \lfloor \frac{|A|}{2} \rfloor$
  if $A[middle] = x$ **then**
    return True
  else if $A[middle] > x$ **then**
    binarySearch(A[1..$middle - 1$], x)
  else
    binarySearch(A[$middle + 1$...1], x)



What is the runtime of recBinSearch?

we would need to ask: which runtime?

worst-case. So for binary search, assume x not in A.

- recursion tree
- recurrence relation