Recurrences

Jurij Volčič
Institut for Matematiske Fag

Outline

- Recurrences in counting problems and recursive programs
- Solving recurrences
  - Backtracking method
  - Guess + Induction method
  - Linear homogeneous recurrences
  - Recursion tree method (Tuesday)
  - Master method (Tuesday)

Reading: KBR 3.5

Take a look at CLRS 4.4. and 4.5 before tomorrow's class

Recurrence relations

Reminder: We have already seen recurrence relations as a way to define sequences. For example,

$$1, 1, 2, 3, 5, 8, 13, 21, \ldots$$

can be defined via

$$f_0 = 1, \quad f_1 = 1 \qquad \text{(initial/boundary conditions)}$$

$$f_n = f_{n-1} + f_{n-2} \qquad \text{(recurrence relation)}$$

Today's goal: Express the n^{th} term as a function of n.
(Solve the recurrence)

How do recurrences come up?

- In **counting problems**.
  - Ex: Count the number of ways to tile an $2 \times n$ grid with 1×2 or 2×1 dominos

$$
\begin{aligned}
a_1 &= 1, a_2 = 2 \\
a_n &= a_{n-1} + a_{n-2} \qquad\qquad (n \geqslant 3)
\end{aligned}
$$

- When estimating the **runtime of recursive algorithms** (e.g. divide-and-conquer)
  - Merge sort:

$$
\begin{aligned}
T_1 &= \Theta(1) \\
T_n &= T_{\lfloor n/2 \rfloor} + T_{\lceil n/2 \rceil} + \Theta(n) \qquad (n \geqslant 2)
\end{aligned}
$$

Divide-and-Conquer

- Divide a problem of size n into a sub-problems each of size $\frac{n}{b}$
- $f(n)$: the cost of dividing and then re-combining the partial solutions into a full one

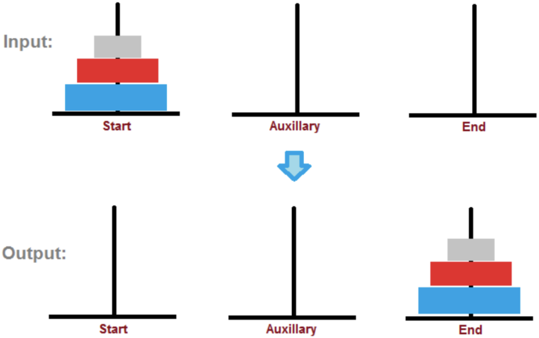
Runtime on input of size n:

$$T_n = aT_{n/b} + f(n)$$

Counting example: **Towers of Hanoi**

Goal: Move the disks to the end rod using as few steps as possible.

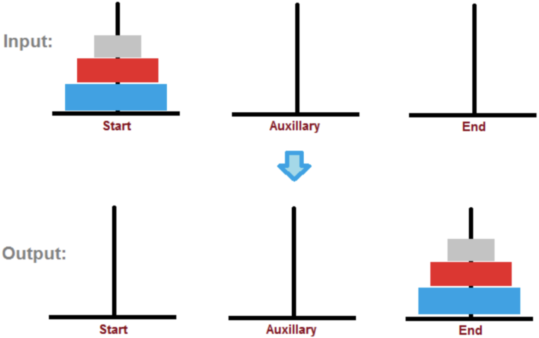
- One step: choose one of the topmost disks and place it on top of another stack or on an empty rod.
- No larger disk may be placed on top of a smaller disk.



Towers of Hanoi

Let T_n be the number of steps needed to move n disks.

- Compute T_1, T_2, T_3.
- Express T_n in terms of T_{n-1}



Solving the recurrences

$$T_1 = 1$$
$$T_n = 2T_{n-1} + 1$$

Method 1: Backtracking

$$T_1 = 1$$
$$T_n = 2T_{n-1} + 1 \qquad n \geqslant 2$$

$$\begin{aligned}
T_n &= 2T_{n-1} + 1 \\
&= 2\left(2T_{n-2} + 1\right) + 1 = 2^2 T_{n-2} + 2 + 1 \\
&= 2^2\left(2T_{n-3} + 1\right) + 2 + 1 = 2^3 T_{n-3} + 2^2 + 2 + 1 \\
&= 2^3\left(2T_{n-4} + 1\right) + 2 + 1 = 2^4 T_{n-4} + 2^3 + 2^2 + 2 + 1 \\
&\vdots \\
&= 2^{n-1} T_{n-(n-1)} + 2^{n-2} + \ldots + 2^2 + 2 + 1 \\
&= 2^{n-1} + 2^{n-2} + \ldots + 2^2 + 2^1 + 2^0 \\
&= 2^n - 1
\end{aligned}$$

Reminder of useful formulas

- **Arithmetic progression**

$$d+2d+3d+\ldots+nd = d(1+2+\ldots+n) = d\cdot\frac{(1+n)n}{2}$$

- **Finite geometric progression** $(q \neq 1)$

$$1+q+q^2+\ldots+q^n = \frac{q^{n+1}-1}{q-1}$$

- **Infinite geometric progression** with $|q|<1$

$$1+q+q^2+q^3+\ldots = \frac{1}{1-q}$$

Method 2: Guess + Induction

- Guess the solution (Ex: $T_n = 2^n - 1$)
- Prove that the solution is correct using induction.

Thm. $2^n - 1$ satisfies the recurrence

$$\begin{cases} T_1 = 1 \\ T_n = 2T_{n-1} + 1 \quad \text{for } n \geqslant 2 \end{cases}$$

Proof. (by induction on n)

Base case: $n = 1$

$$T_1 \stackrel{?}{=} 2^1 - 1 \checkmark$$

Induction step:

Assume $T_n = 2^n - 1$ for some $n \geqslant 1$. **(Induction Hypothesis)**

Need to show: $T_{n+1} = 2^{n+1} - 1$.

Method 3: Linear homogeneous recurrences

Linear homogeneous recurrence relation of degree k:

$$a_n = r_1 a_{n-1} + r_2 a_{n-2} + \ldots + r_k a_{n-k}$$

where, $r_1, \ldots, r_k$ are constants.

Which of the following are linear homogeneous recurrences with constant coefficients?

1. **Fibonacci:** $F_n = F_{n-1} + F_{n-2}$
2. **Hanoi:** $T_n = 2T_{n-1} + 1$
3. **Merge Sort:** $M_n = 2M_{\lfloor n/2 \rfloor} + \mathrm{const} \cdot n$
4. $a_n = a_{n-1} + 2a_{n-5}$
5. **Factorial:** $b_n = n b_{n-1}$

Method 3: Linear homogeneous recurrences

Linear homogeneous recurrence relation of degree k:

$$a_n = r_1 a_{n-1} + r_2 a_{n-2} + \ldots + r_k a_{n-k}$$

where, $r_1, \ldots, r_k$ are constants.

The recipe

1. Find the roots of the characteristic equation
$$x^k = r_1 x^{k-1} + \ldots + r_{k-1} x + r_k$$
2. For $k = 2$, we have two roots s_1 and s_2.
   - If $s_1 \neq s_2$
   $$a_n = u s_1^n + v s_2^n$$
   - If $s_1 = s_2$
   $$a_n = u s_1^n + v n s_1^n$$
3. Use the initial conditions to find constants u, v.

Why does the recipe work? $(k = 2)$

two distinct solutions $s_1 \neq s_2$

Suppose $s_1 \neq s_2$ are solutions of $x^2 = r_1 x + r_2$.

If for some $u, v \in \mathbb{R}$ we denote $a_n = u s_1^n + v s_2^n$ for every $n \in \mathbb{N}$, then

$$r_1 a_{n-1} + r_2 a_{n-2}$$

$$= r_1 (u s_1^{n-1} + v s_2^{n-1}) + r_2 (u s_1^{n-2} + v s_2^{n-2})$$

$$= u(r_1 s_1 + r_2) s_1^{n-2} + v(r_1 s_2 + r_2) s_s^{n-2}$$

$$= u s_1^2 s_1^{n-2} + v s_2^2 s_2^{n-2}$$

$$= a_n$$

Why does the recipe work? ($k = 2$)

one solution $s_1 = s_2 =: s$

Suppose s is the sole solution of $x^2 = r_1 x + r_2$.

Then $r_1 = 2s$ and $r_2 = -s^2$.

If for some $u, v \in \mathbb{R}$ we denote $a_n = us^n + vns^n$ for every $n \in \mathbb{N}$, then

$$r_1 a_{n-1} + r_2 a_{n-2}$$

$$= 2s(us^{n-1} + v(n-1)s^{n-1}) - s^2(us^{n-2} + v(n-2)s^{n-2})$$

$$= u(2s^n - s^n) + v(2(n-1)s^n - (n-2)s^n)$$

$$= a_n$$

Example: the Fibonacci sequence

$F_n = F_{n-1} + F_{n-2}, \quad F_1 = F_2 = 1$