

1. For any positive integer  $n$ , the  $n$ th **Fibonacci string**  $F_n$  is defined recursively as follows, where  $x \cdot y$  denotes the concatenation of strings  $x$  and  $y$ :

$$F_1 := 0$$

$$F_2 := 1$$

$$F_n := F_{n-1} \cdot F_{n-2} \quad \text{for all } n \geq 3$$

For example,  $F_3 = 10$  and  $F_4 = 101$ .  $F_5 = 10110$   $F_6 = 10110101$

$$F_7 = 101101011010$$

(a) What is  $F_8$ ?

(b) **Prove** that every Fibonacci string except  $F_1$  starts with **1**.

(c) **Prove** that no Fibonacci string contains the substring **00**.

(a)  $F_8 = 101101011011010110$

(b) Let  $n$  be an arb. integer s.t.  $n > 1$

Assume for all int  $k < n$  s.t.  $k > 1$  that  $F_k$  starts with **1**

• If  $n=2$ , then  $F_2=1$  by def ✓

• If  $n > 2$  then  $F_n = \underbrace{F_{n-1}} \cdot F_{n-2}$

So  $F_n$  starts with **1**. ↑ starts with **1** by IH

(c) Let  $n$  be an arb. positive integer

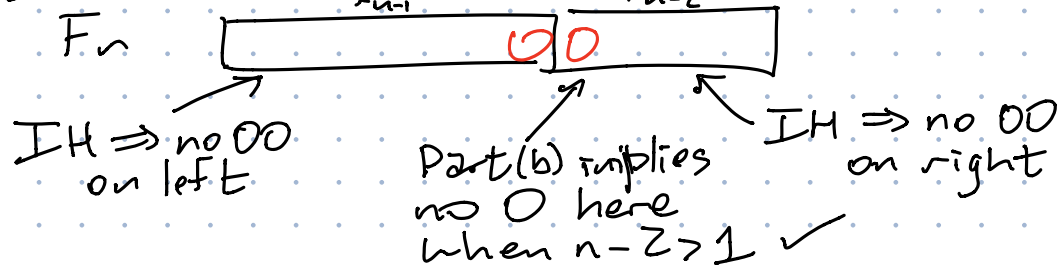
Assume, for all pos int  $k < n$ , that  $F_k$  has no **00**

• If  $n=1$   $F_1=0$  by def no **00** ✓

• If  $n=2$   $F_2=1$  by def no **00** ✓

• If  $n=3$   $F_3=10$  no **00** ✓

• If  $n > 3$   $F_n = F_{n-1} \cdot F_{n-2}$



4. The new swap-puzzle game *Candy Swap Saga XIII* involves  $n$  cute animals numbered 1 through  $n$ . Each animal holds one of three types of candy: circus peanuts, Heath bars, and Cioccolateria Gardini chocolate truffles. You also have a candy in your hand; at the start of the game, you have a circus peanut.

To earn points, you visit each of the animals in order from 1 to  $n$ . For each animal, you can either keep the candy in your hand or exchange it with the candy the animal is holding.

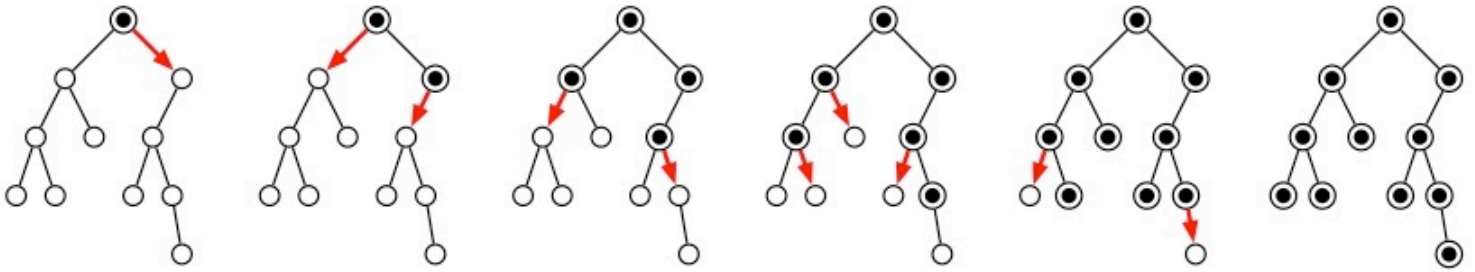
- If you swap your candy for another candy of the *same* type, you earn one point.
- If you swap your candy for a candy of a *different* type, you lose one point. (Yes, your score can be negative.)
- If you visit an animal and decide not to swap candy, your score does not change.

You *must* visit the animals in order, and once you visit an animal, you can never visit it again.

Describe and analyze an efficient algorithm to compute your maximum possible score. Your input is an array  $C[1..n]$ , where  $C[i]$  is the type of candy that the  $i$ th animal is holding.



2. Suppose we need to distribute a message to all the nodes in a given binary tree. Initially, only the root node knows the message. In a single round, each node that knows the message is allowed (but not required) forward it to at most one of its children. Describe and analyze an algorithm to compute the minimum number of rounds required for the message to be delivered to all nodes in the tree.



A message being distributed through a binary tree in five rounds.

Let  $\text{MinRounds}(v) = \text{min \# rounds to distribute the message from } v \text{ to all of } v\text{'s descendants.}$

$$\text{MinRounds}(v) = \begin{cases} 0 & v \text{ is a leaf} \\ 1 + \text{MinRounds}(w) & v \text{ has one child } w \\ \min \left\{ \begin{array}{l} \max \left\{ \begin{array}{l} 1 + \text{MinRounds}(u) \\ 2 + \text{MinRounds}(w) \end{array} \right\} \\ \max \left\{ \begin{array}{l} 1 + \text{MinRounds}(w) \\ 2 + \text{MinRounds}(u) \end{array} \right\} \end{array} \right\} & v \text{ has two children } u, w \end{cases}$$

Memoize into tree itself  $v_0 \text{ MinRounds.}$

Eval. in postorder

$O(V)$  time

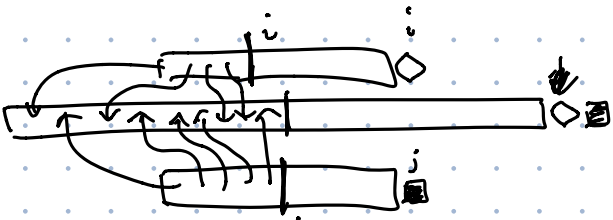
5. A shuffle of two strings  $X$  and  $Y$  is formed by interspersing the characters into a new string, keeping the characters of  $X$  and  $Y$  in the same order. For example, the string **BANANAANANAS** is a shuffle of the strings **BANANA** and **ANANAS** in several different ways.

**BANANA**ANANAS

BAN**ANA**ANANAS

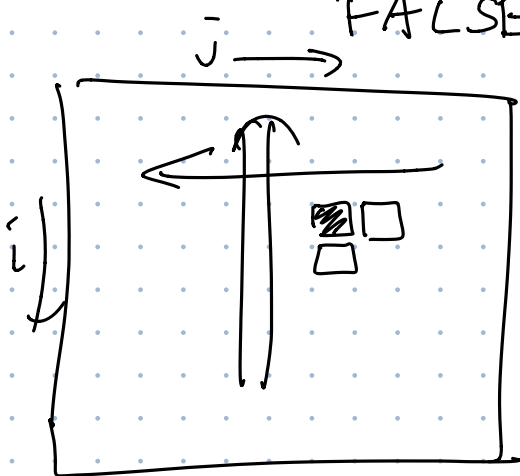
BANANA**ANANAS**

Given three strings  $A[1..m]$ ,  $B[1..n]$ , and  $C[1..m+n]$ , describe and analyze an algorithm to determine whether  $C$  is a shuffle of  $A$  and  $B$ .



Shuffle( $i, j$ ) = True iff  $C[i+j-1..m+n]$  is a shuffle of  $A[i..m]$  and  $B[j..n]$

$$\text{Shuffle}(i, j) = \begin{cases} \text{True} & i=m+1 \quad j=n+1 \\ \text{Shuffle}(i+1, j) \vee \text{Shuffle}(i, j+1) & \text{if } C[i+j-1] = A[i] \text{ and } C[i+j-1] = B[j] \\ \text{Shuffle}(i+1, j) & \text{if } C[i+j-1] = A[i] \neq B[j] \\ \text{Shuffle}(i, j+1) & \text{if } C[i+j-1] = B[j] \neq A[i] \\ \text{FALSE} & \text{else} \end{cases}$$




$O(n^2)$  time

2. A *shuffle* of two strings  $X$  and  $Y$  is formed by interspersing the characters into a new string, keeping the characters of  $X$  and  $Y$  in the same order. A *smooth* shuffle of  $X$  and  $Y$  is a shuffle of  $X$  and  $Y$  that never uses more than two consecutive symbols of either string. For example,

- `prDoyGNArAmmmIcng` is a smooth shuffle of `DYNAMIC` and `programming`.
- `DYprNogrAammIcing` is a shuffle of `DYNAMIC` and `programming`, but it is not a smooth shuffle (because of the substrings `ogr` and `ing`).

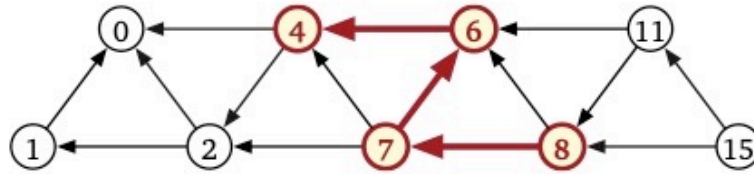
Describe and analyze an algorithm to decide, given three strings  $X$ ,  $Y$ , and  $Z$ , whether  $Z$  is a smooth shuffle of  $X$  and  $Y$ .



2. Let  $G$  be a **directed** graph, where every vertex  $v$  has an associated height  $h(v)$ , and for every edge  $u \rightarrow v$  we have the inequality  $h(u) > h(v)$ . Assume all heights are distinct. The **span** of a path from  $u$  to  $v$  is the height difference  $h(u) - h(v)$ .

*exactly* Describe and analyze an algorithm to find the **minimum span** of a path in  $G$  with ~~at least~~  $k$  edges. Your input consists of the graph  $G$ , the vertex heights  $h(\cdot)$ , and the integer  $k$ . Report the running time of your algorithm as a function of  $V$ ,  $E$ , and  $k$ .

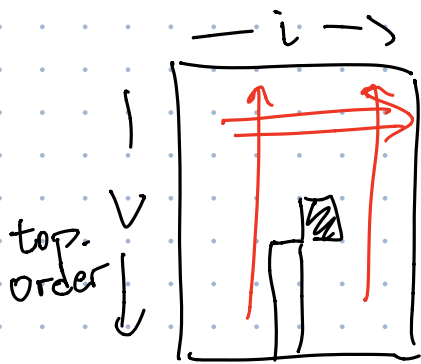
For example, given the following labeled graph and the integer  $k = 3$  as input, your algorithm should return the integer 4, which is the span of the path  $8 \rightarrow 7 \rightarrow 6 \rightarrow 4$ .



$Highest(v, i) =$  maximum height of a vertex reachable from  $v$  along a path with ~~at least~~ *exactly*  $i$  edges.

$$Highest(v, i) = \begin{cases} h(v) & i=0 \\ \max \{ Highest(w, i-1) \mid v \rightarrow w \} & i > 0 \\ (\max \emptyset = -\infty) \end{cases}$$

memoize into arrays  $v.Highest[0..k]$  at every node  $v$

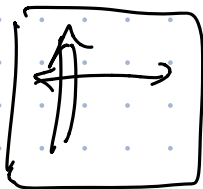


for  $i \in 0$  to  $k$   
for all  $v$  in whatever order you like

$$\left. \begin{array}{l} \text{ } \\ \text{ } \end{array} \right\} O(\deg(v))$$

$$\underline{\min \text{ span}(v) = h(v) - Highest(v, k)}$$

$$\text{Time: } O(k) \cdot \sum_v \deg(v) = \underline{O(kE)}$$



for all  $v$  in post  
for all  $i$  in whatever.

1. A **basic arithmetic expression** is composed of characters from the set  $\{1, +, \times\}$  and parentheses. Almost every integer can be represented by more than one basic arithmetic expression. For example, all of the following basic arithmetic expressions represent the integer 14:

$$\begin{aligned}
 &1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 \\
 &((1 + 1) \times (1 + 1 + 1 + 1 + 1)) + ((1 + 1) \times (1 + 1)) \\
 &(1 + 1) \times (1 + 1 + 1 + 1 + 1 + 1 + 1) \\
 &(1 + 1) \times (((1 + 1 + 1) \times (1 + 1)) + 1)
 \end{aligned}$$

Describe and analyze an algorithm to compute, given an integer  $n$  as input, the minimum number of 1's in a basic arithmetic expression whose value is equal to  $n$ . The number of parentheses doesn't matter, just the number of 1's. For example, when  $n = 14$ , your algorithm should return 8, for the final expression above. The running time of your algorithm should be bounded by a small polynomial function of  $n$ .

$\text{MinBAX}(n) = \text{min \#1's in a basic arith. exp. with value } n.$

$$\text{MinBAX}(n) = \begin{cases} \frac{1}{0} & n = 1 \\ \min \left\{ \begin{array}{l} \min \{ \text{MinBAX}(l) + \text{MinBAX}(n-l) \mid 1 \leq l \leq \frac{n}{2} \} \\ \min \{ \text{MinBAX}(l) + \text{MinBAX}(n/l) \mid 2 \leq l \leq \sqrt{n} \text{ and } n/l \text{ is int.} \} \end{array} \right. & n = 0 \end{cases} \quad \text{o/w}$$

Memoize into 1D array 

$O(n^2)$  time