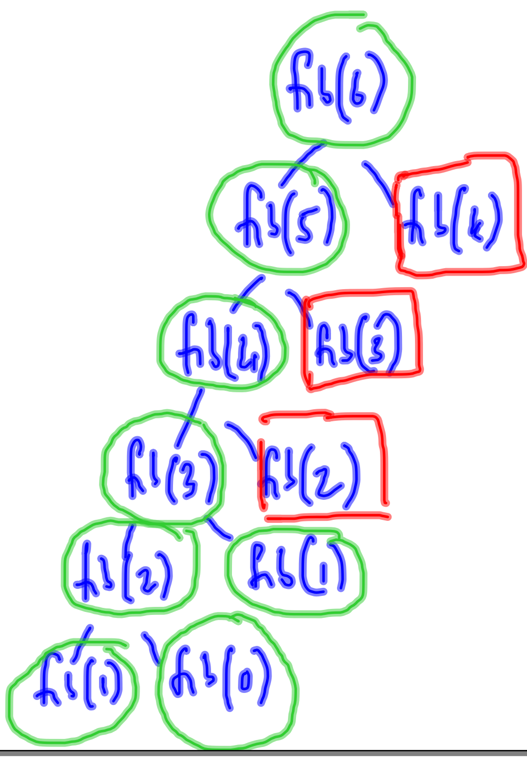
Evaluating recursively defined functions efficiently

$fib(n) = fib(n-1) + fib(n-2)$, $fib(0) = 0$, $fib(1) = 1$




Memoization

Store f(k) in a table

look up table before recomputing

fib(n) depends on fib(0), . . . , fib(n-1)

Table

| | |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | ~~-1~~ 1 |
| ⋮ | |
| n-1 | -1 |

First time fib(k) is evaluated,

use recursive defn,

store answer in table

If table(k) $\neq$ -1,

table(k) = fib(k)

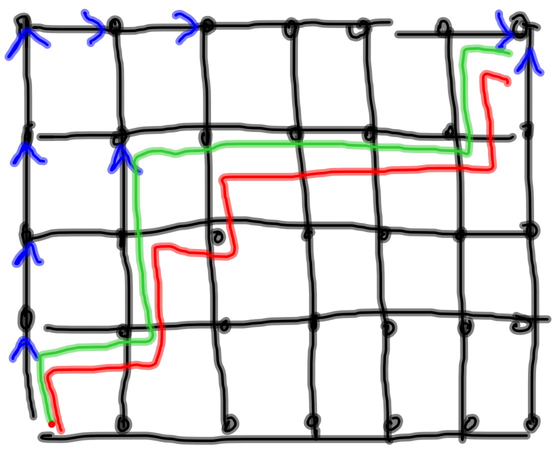
Use table value directly

Directly fill in table without using recursive
defn explicitly

Start with fib(0), fib(1)
Successively compute fib(2), fib(3) . . fib(n)

Memoization – Recursion with memory
⇓
Dynamic Programming – Iteratively fill up the memo table

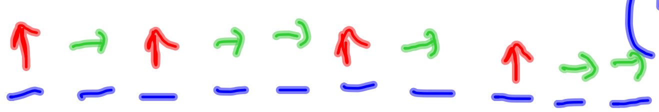
Destination



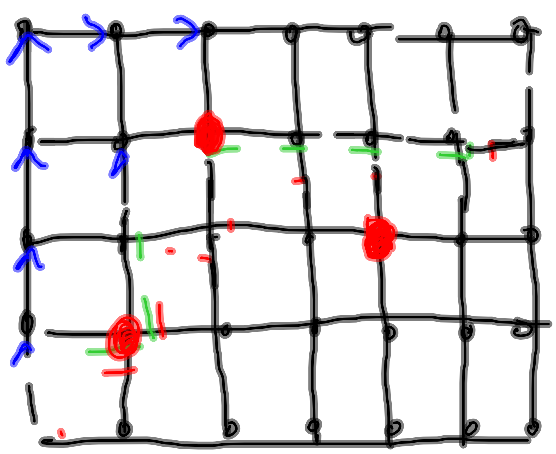
How many different paths are there from Source to Destination?

Source

$6 \times 4$ grid : 10 steps



$\binom{10}{6}$ right moves $= \binom{10}{4}$ up moves



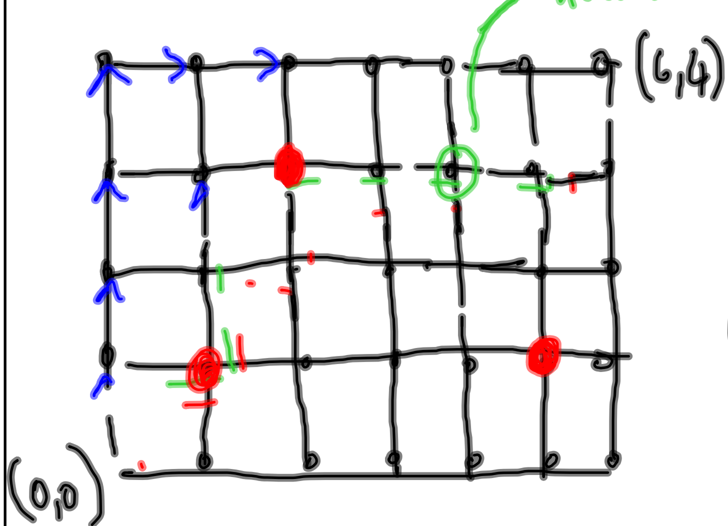
Potholes - intersections are blocked!

How many paths?

Inclusion - Exclusion

- Tedious
- Depends a lot on # & placement of potholes

how do I reach here?




Instead

Paths to $(i,j)$ come from left, or below

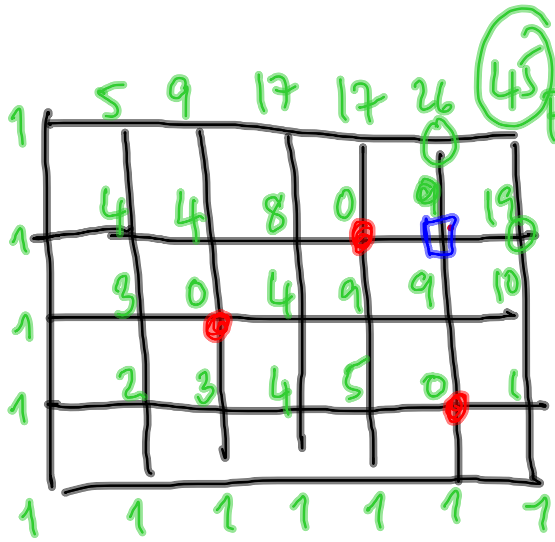
$(i-1,j) \longrightarrow (i,j)$

$\uparrow$

$(i,j-1)$

$\# \text{Paths}(i,j) = \# \text{Paths}(i-1,j) + \# \text{Paths}(i,j-1)$

$\# \text{Paths}(0,0) = 1$



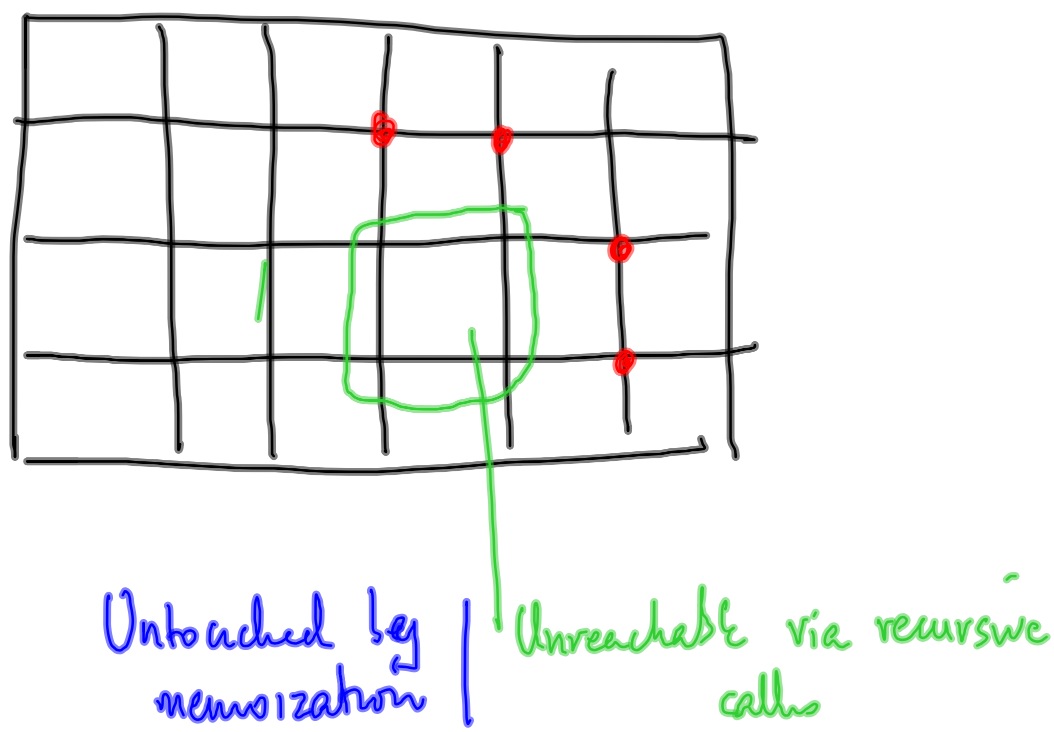

$$P(6,4) = P(5,4) + P(6,3)$$

$$P(4,4) \quad \underline{P(5,3)} \quad \underline{P(5,3)} \quad P(6,2)$$

Memoize $P(i,j)$, $i \le 6$, $j \le 4$

Or, do Dynamic Programming!

# Memoization vs DP



# Sequence alignment

diff command to compare text files

$$s = \overbrace{a_1 \; a_2}^{w_1} \; a_3 \; \overbrace{\cdots \quad}^{w_2} \; \overbrace{\;}^{w_3} \; a_m$$

$$t = \underbrace{b_1 \; b_2}_{w_1} \; \underbrace{b_3 \quad}_{w_2} \; \underbrace{- \; -}_{w_3} \; b_n$$

Maximum overlap between s and t

Other applications

Comparing DNA

Define what it means to match

A block which matches exactly to another block

Subsequence

✓ × ✓ × ✓ × ✓ × ✓

$$a_1\ \boxed{a_2\ a_3}\ a_4\ \boxed{\cdot}\ \cdots\ \boxed{a_{i-1}}\ a_i\ \boxed{a_{i+1}\ \cdots}\ a_m$$

Drop some letters and keep the rest

$$a_1\, a_4 \cdots a_i \cdots a_m$$

Given s & t

Want subsequences u of s, v of t

u & v match ⇒ u = v

|u|, |v| is maximized

Not unique in general

abced     abcd     abed
abecd     abcd     abed

Longest common subsequence problem

length of the

$s = s_1 \, s_2 \, s_3 \, \ldots \, s_m$

$t = t_1 \, t_2 \, t_3 \, \ldots \, t_n$

Alphabet of symbols need not be finite

Need to be able to check $s_i == t_j$ ?

Base Case?

Either s or t is empty $llcs(s,t) = 0$

Typical subproblem

$s_i\ s_{i+1} \ldots s_m$

$llcs(i,j)$

$t_j\ t_{j+1} \ldots t_n$

Original problem: $llcs(1,1)$

Base cases: $llcs(m+1, j)$, $llcs(i, n+1)$

llcs(i, j)

$\ldots \mid s_i \; s_{i+1} \; s_{i+2} \ldots s_m$

$\ldots \mid t_j \; t_{j+1} \; t_{j+2} \ldots t_n$

$s_i == t_j$ Match $(s_i, t_j)$

$\underbrace{1}_{\text{this match}} + \underbrace{\text{llcs}(i+1, j+1)}_{\text{rest}}$

$s_i \neq t_j$ $s_i \; s_{i+1} \; s_{i+2} \; \ldots \; s_m$

$t_j \; t_{j+1} \; t_{j+2} \; \ldots \; t_n$

Suppose $s_i$ is matched to some $t_k$ in lcs

$\Rightarrow k > j$, so $t_j$ is useless

Symmetrically, if $t_j$ matches $s_k$, $k > i$, $s_i$ useless

$= \max(\text{llcs}(i, j+1), \text{llcs}(i+1, j))$

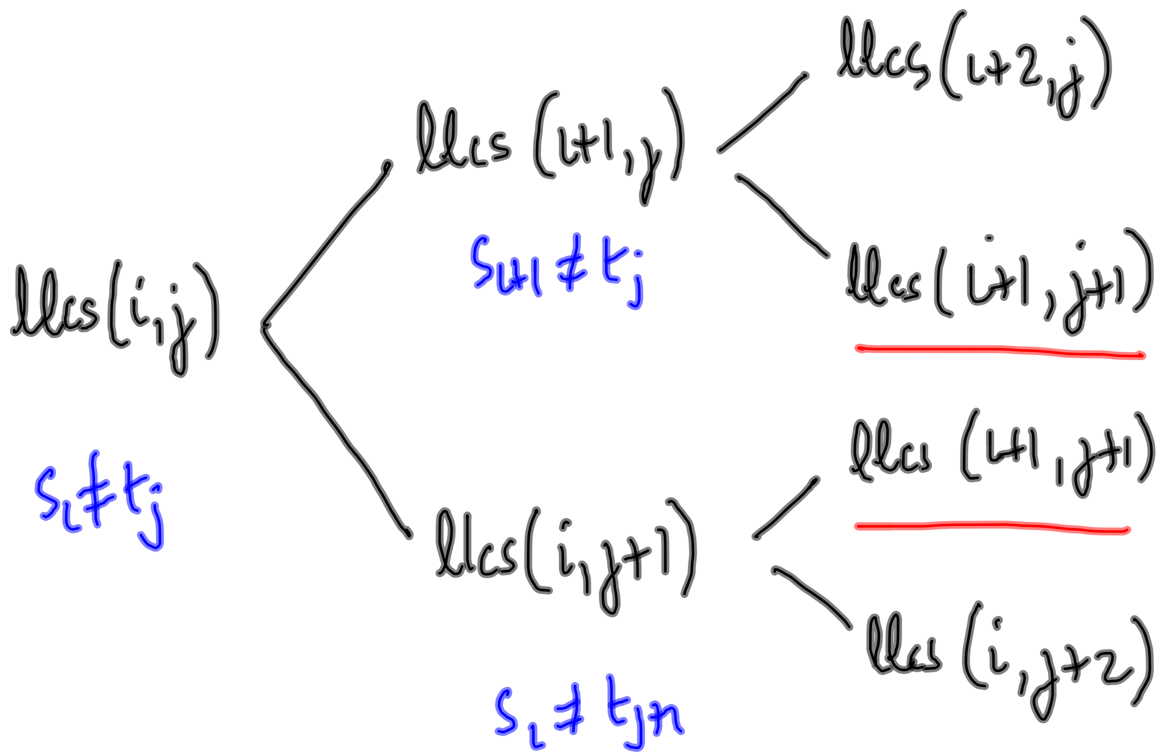
$llcs(i, j)$

if $s_i == t_j$ $\quad 1 + llcs(i+1, j+1)$

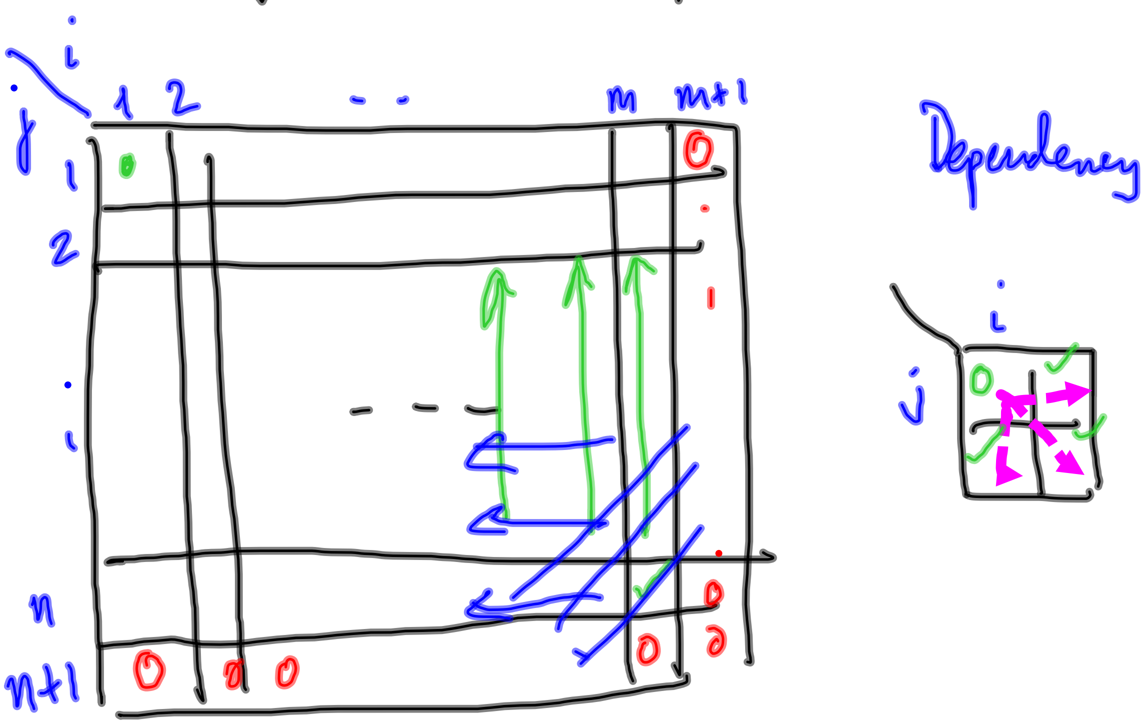
else $\quad \max(llcs(i, j+1), llcs(i+1, j))$

a b c d e f g

a b' c' a d' e f' g'


llcs(i,j)
$s_i \neq t_j$
llcs(i+1,j)
$s_{i+1} \neq t_j$
llcs(i,j+1)
$s_i \neq t_{j+1}$
llcs(i+2,j)
llcs(i+1,j+1)
llcs(i+1,j+1)
llcs(i,j+2)

Need llcs(i,j) $\quad i \le m+1,\ j \le n+1$



Extract a witness?

Given llcs(0,0) = k

Find an actual common subsequence of length k

Find all such.