Lecture 5: Dynamic programming

```
fib(n):                    //computing fib numbers
  if n=0  return 0
  if n=1  return 1
  return fib(n-1)+fib(n-2).
```

Memorization

0 1
| 0 | 1 | 1 | 2 | 3 | 5 | 8 | . . . |
→

$f(x_1, x_2, \ldots, x_t)$ recursive + memorization

Analyze running time?

# of distinct recursive calls. $\prod_{i=1}^{t} |\#\text{values of } x_i| \le N_1$

Compute amount of work to compute value of func ignoring recursive calls. N_2

recursive function with memorization

$O(N_1 \cdot N_2)$ Total running time.

G: Graph n vertices ≡ monstrously large

Implicit access to G:

- access vertices n vertex(i)
- given i,j is_edge(i,j). returns if there is an edge between i and j

---

Connectivity

Given, s,t vertices

- Is s connected to t in G.
- What is the len of the shortest path?

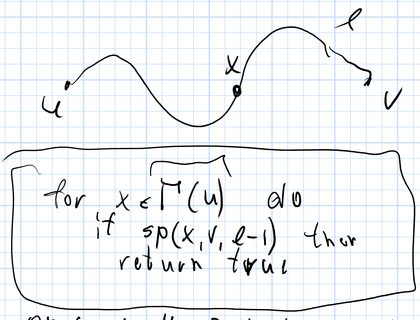
BFS requires $\Theta(n)$ space.

$O(\ell \log n)$

Alg

- guess length of shortest path between u and v. $\ell = 1, \ldots, n$

```
sp(u,v,ℓ):
  if ℓ=0 and u≠v return false
  if ℓ=1 and is_edge(u,v) then true
  for x=1 to n do
      if sp(u,x,⌊ℓ/2⌋) and sp(x,v,⌈ℓ/2⌉) then
          return true
  return false.
```

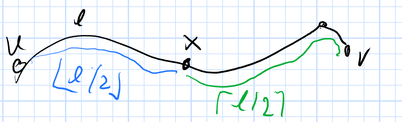


Depth of recursion is $O(\log n)$

Run in $O(\log n)$ words

Runs in $O(\log^2 n)$ bits

Oops: Depth $\Omega(n)$ in worst case



Savitch theorem

$|SL| = L^2$?