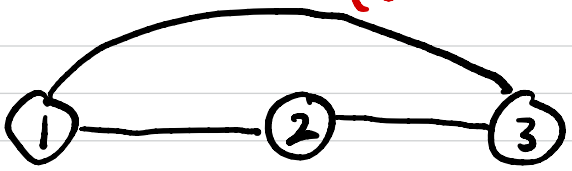
Graphs!

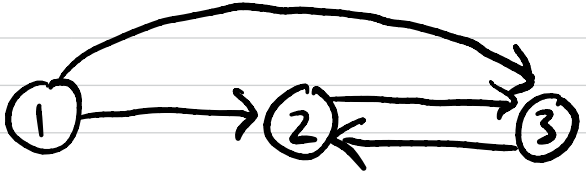
Graphs

(undirected)



$G = (V, E)$

(directed)



$(u, v) \in E$ if $u \to v$

Parameters: $n = |V|$

$m = |E|$ $\quad (m \leq n^2)$

$deg(u) = 3$

$in\text{-}deg(u) = 1$

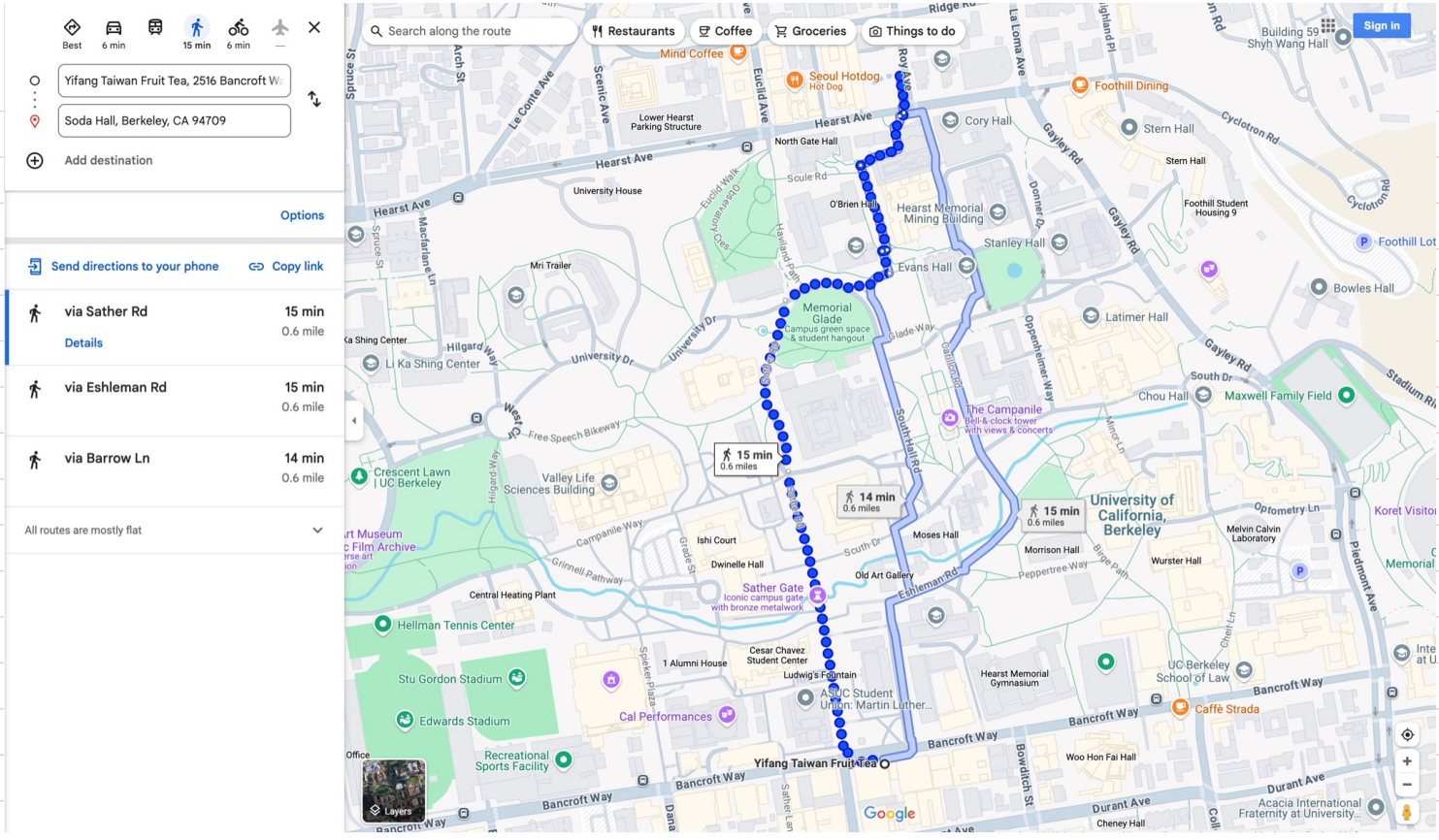
$out\text{-}deg(u) = 2$

aka $deg(u)$

Facebook friend graph (undirected)

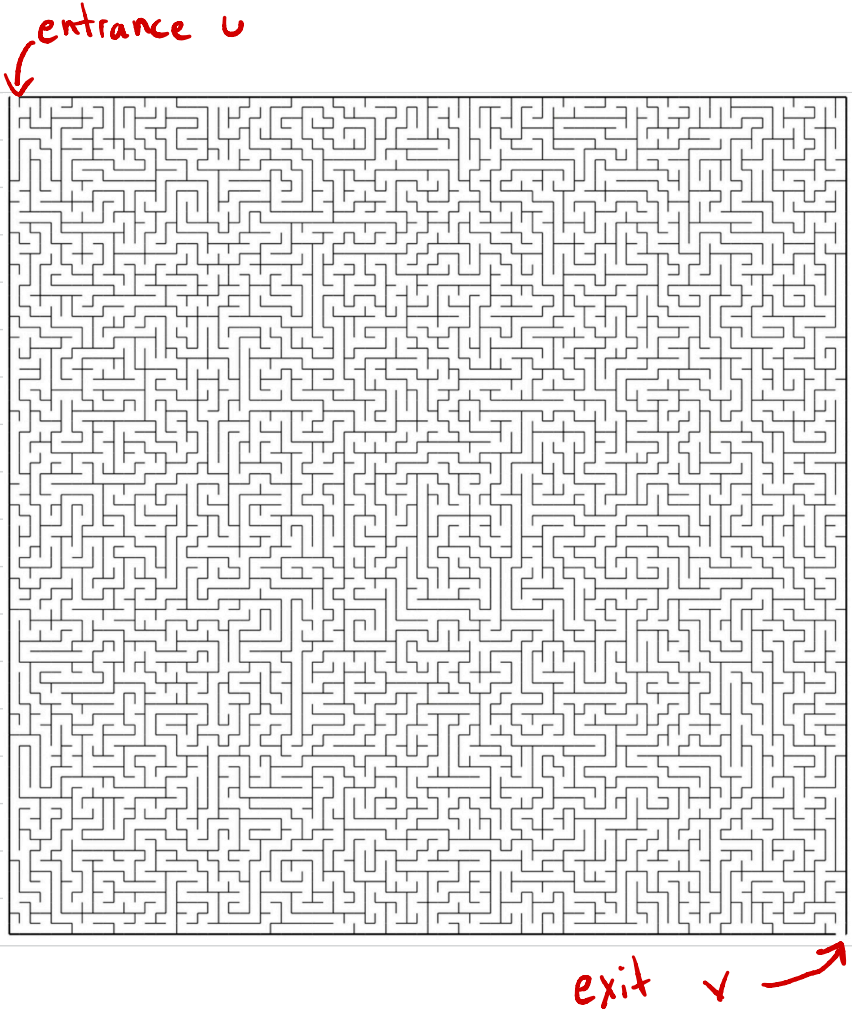
- $n = 2.91$ billion users
- avg user has 338 friends
  $m =$ 500 billion edges

Question: Who are my friends?



Google Maps

Q: Shortest path to boba?



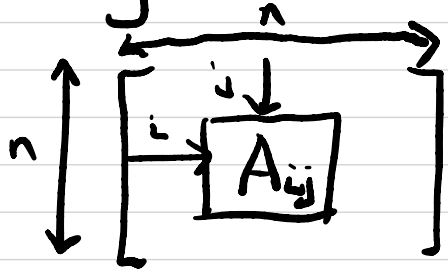
Maze solving

Q: Is u connected to v?

Representing graphs on computers

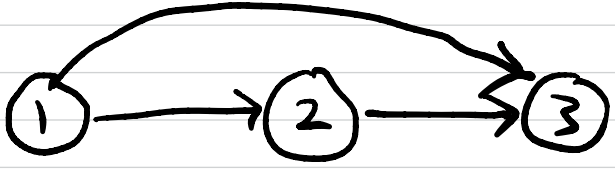
$V = \{1, \ldots, n\}$

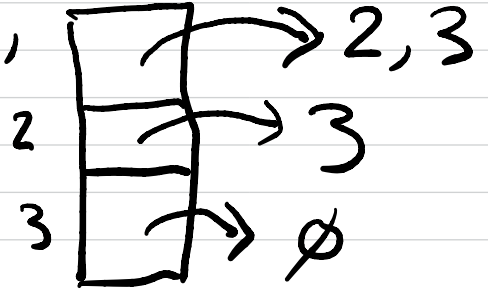
(1) adjacency matrix representation



$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{o.w.} \end{cases}$$

(2) adjacency list representation (unordered)





(linked list)

		Adjacency matrix	Adjacency list
	size	$O(n^2)$ space	$O(n+m)$
time to:	answer is $(u,v) \in E$?	$O(1)$	$O(n)$ $O(\deg(u))$
	enumerate u's neighbors	$O(n)$	$O(\deg(u))$

Facebook graph w/ adjacency matrix

size n^2 = $(2.91 \text{ billion})^2$ space

$= 8.5 \times 10^{18} \approx 1$ million TB

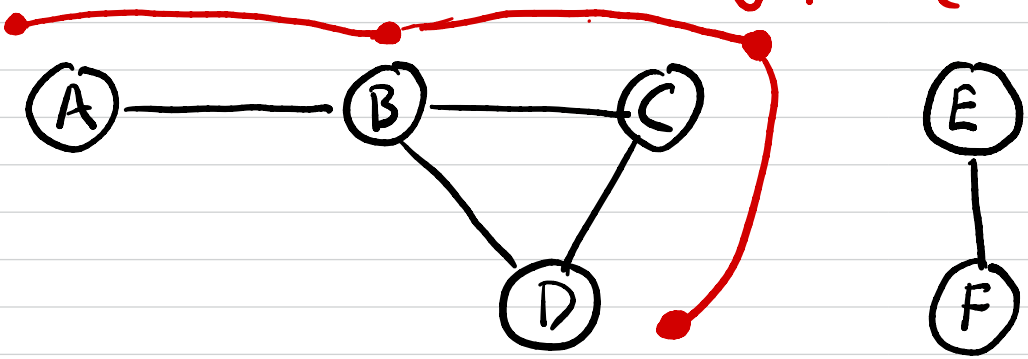
computing friends list = 2.91 billion steps

Graph Exploration

Useful for:

1. Is there a path from u to v?
2. Is G connected?
3. What are G's connected components?

Connectivity for undirected graphs (∃ path from u to v?)




```
explore(G, v)
  visited[v] = true

  for v s.t. (v, v) ∈ E
    if visited[v] = false
      explore(G, v)
```

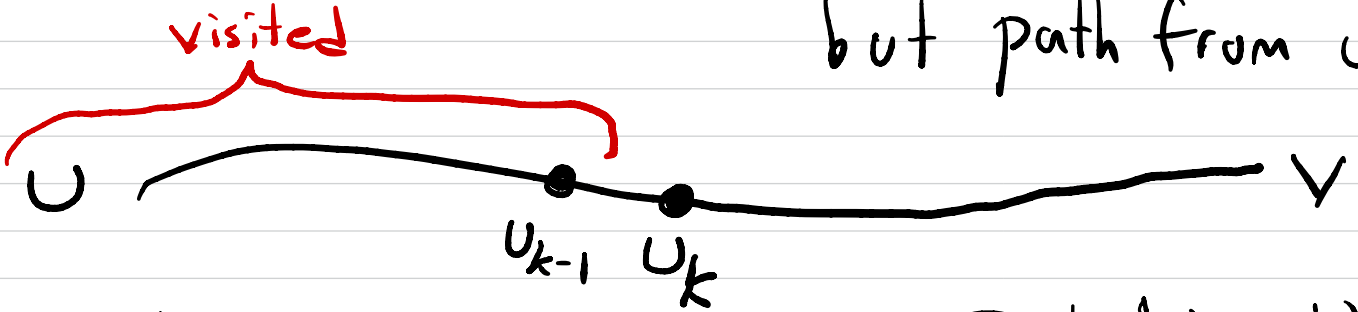
boolean array visited[n]
(init to all 0's)

Property: explore(G, v) visits exactly the vertices w s.t. G has path from v to w

Pf: 1. w explored $\Rightarrow$ path from v to w

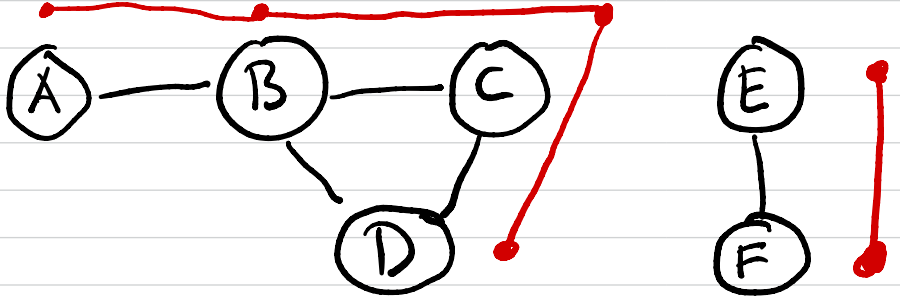
2. path from v to w $\Rightarrow$ w explored

Assume false, so w not explored but path from v to w



Must have called explore(G, u_k). But did not! Contradiction. □

Connectivity for undirected graphs (∃ path from u to v?)



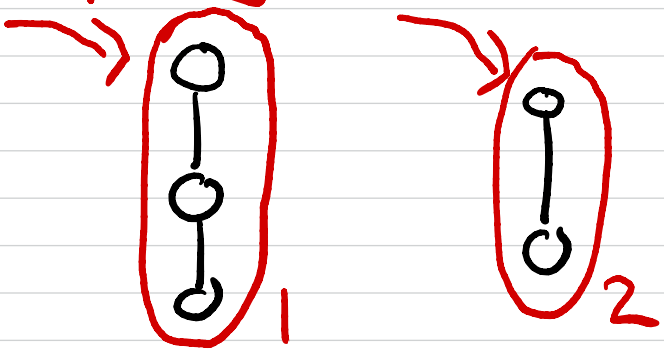

```
explore(G,u)
  visited[u] = true

  for v s.t. (u,v) ∈ E
      if visited[v] = false
          explore(G,v)
```

```
dfs(G)
  boolean array visited[n]
      (init to all 0's)

  for v ∈ V
      if visited[v] = false
          explore(G,v)
```

Computing G's connected components (undirected)



(3 connected components)

```
explore(G,v)
  visited[v] = true
  ccnum[v] = count

  for w s.t. (v,w) ∈ E
      if visited[w] = false
          explore(G,w)
```

```
dfs(G)
  boolean array visited[n]
      (init to all 0's)
  count = 1
  int array ccnum[n]

  for v ∈ V
      if visited[v] = false
          explore(G,v)
          count = count + 1
```

Runtime of DFS

Only call explore(G,v) exactly once, for each v

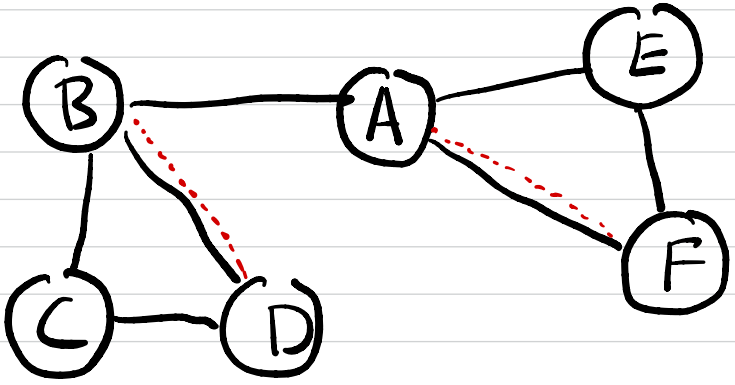
Runtime of explore(G,v)

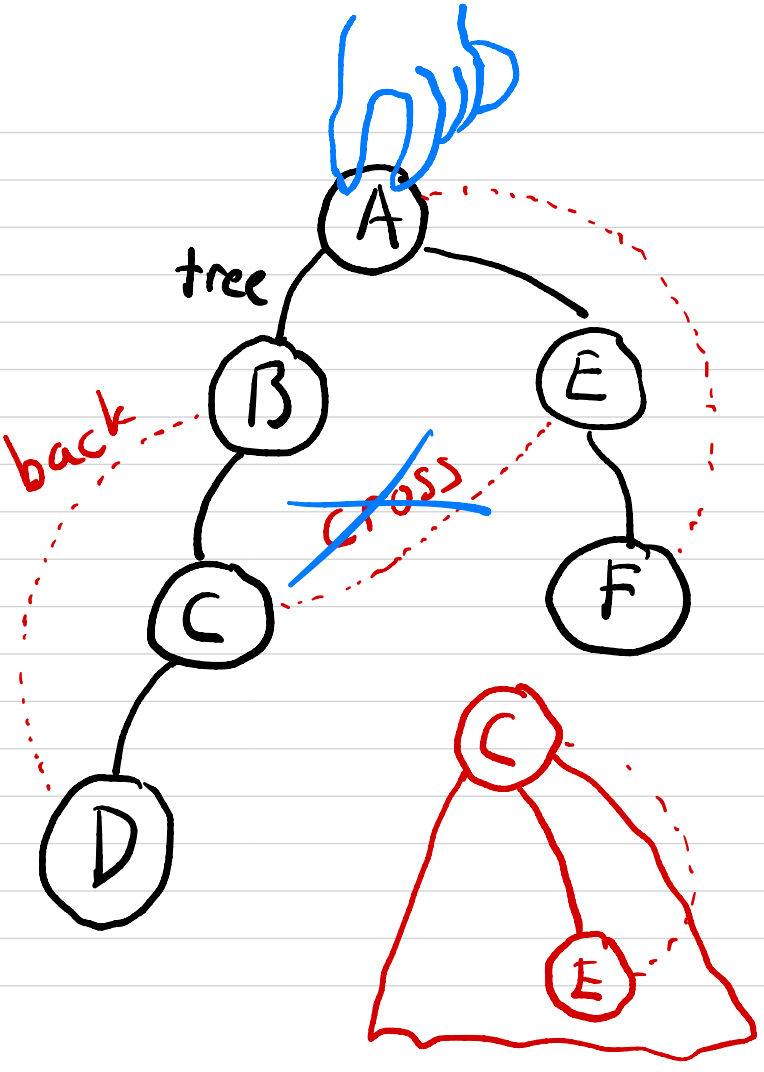
- set visited[v] = true — $O(1)$ time
- enumerate v's neighbors — $O(\deg(v))$ time (adjacency list)

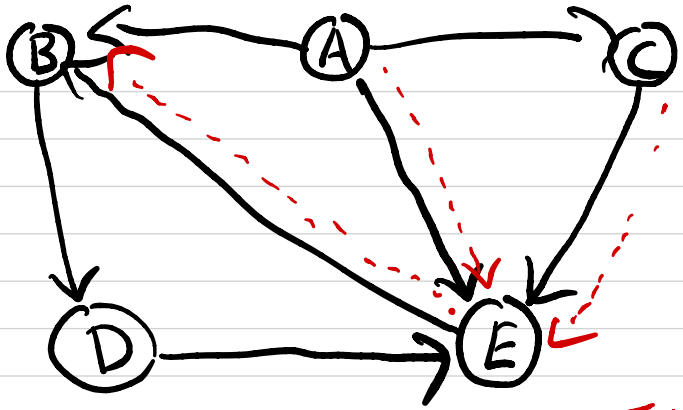
$$\text{total} = \sum_{v} O(1 + \deg(v))$$

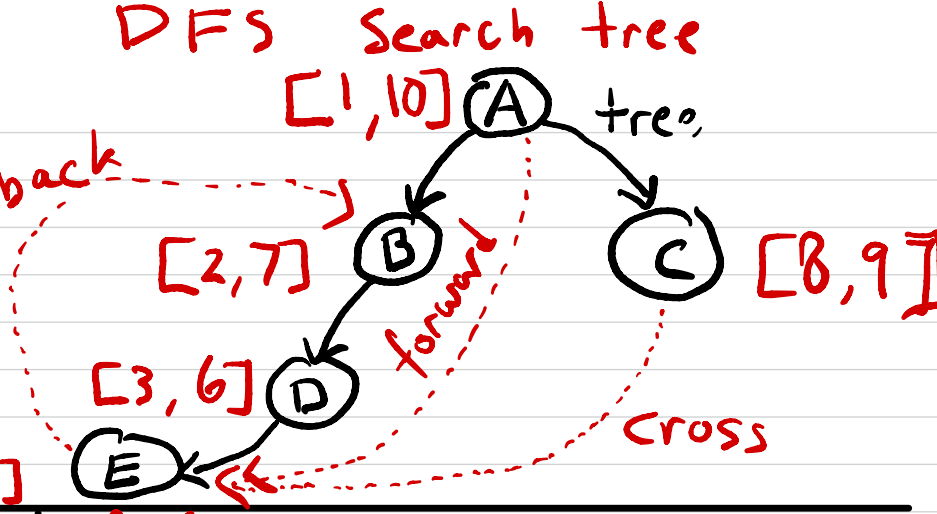
$$= O(n + m) \text{ time}$$

The DFS tree










```
explore(G,v)
  visited[v] = true
  pre[v] = clock
  clock = clock+1
  for x s.t. (v,x) ∈ E
      if visited[x] = false
          explore(G,x)
  post[v] = clock
  clock = clock+1
```

```
dfs(G)
  boolean array visited[n]
      (init to all 0's)
  clock = 1
  int array pre[n], post[n]
  for v ∈ V
      if visited[v] = false
          explore(G,v)
```