$G = (V, E)$ undirected

$DG = \langle V, E \rangle$ digraph

$E = \{(w, v) \mid w \in V \&\& v \in V\}$

$E = \{\langle w, v \rangle \mid w \in V \&\& v \in V\}$

Directed Graph DG

$V = \{1, 2, 3, 4, 5\}$

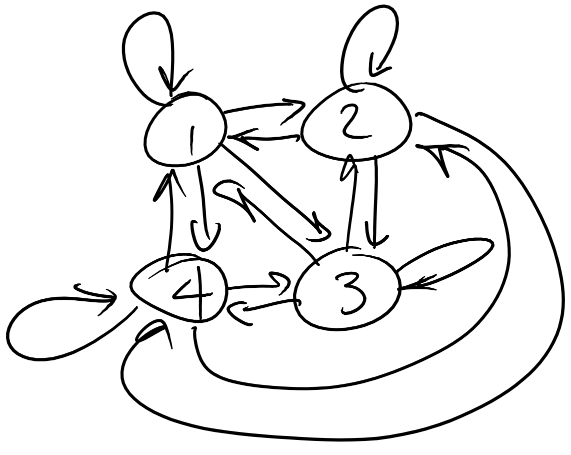
$O(|V|)$

$E = \{\langle 1,2\rangle, \langle 2,2\rangle, \langle 3,5\rangle, \langle 5,2\rangle\}$

$O(|E|)$

$N = |V|$

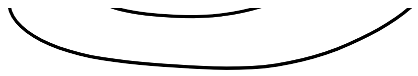
? $(N - 1)' = |E|$



$N = 4$

Adjacency Matrix

	1	2	3	4
1				
2				
3				
4				



3				
4				

$$\begin{tikzcd} \boxed{i} \arrow[r] & \boxed{j} \end{tikzcd}$$

j is adjacent from i

i is adjacent to j

<i, j> is adjacent

$$\begin{tikzcd} 1 \arrow[r, bend left] & 2 \arrow[l, bend left] \arrow[r] & 3 \arrow[loop above] \arrow[ll, bend left] \end{tikzcd}$$

$V = \{1, 2, 3\}$

$$E = \begin{array}{c|ccc} & 1\checkmark & 2\checkmark & 3 \\ \hline \textcircled{1} & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 3 & 1 & 0 & 1 \end{array}$$

Adj[2][3]

$V = 100$ Adjacency Matrix 100×100

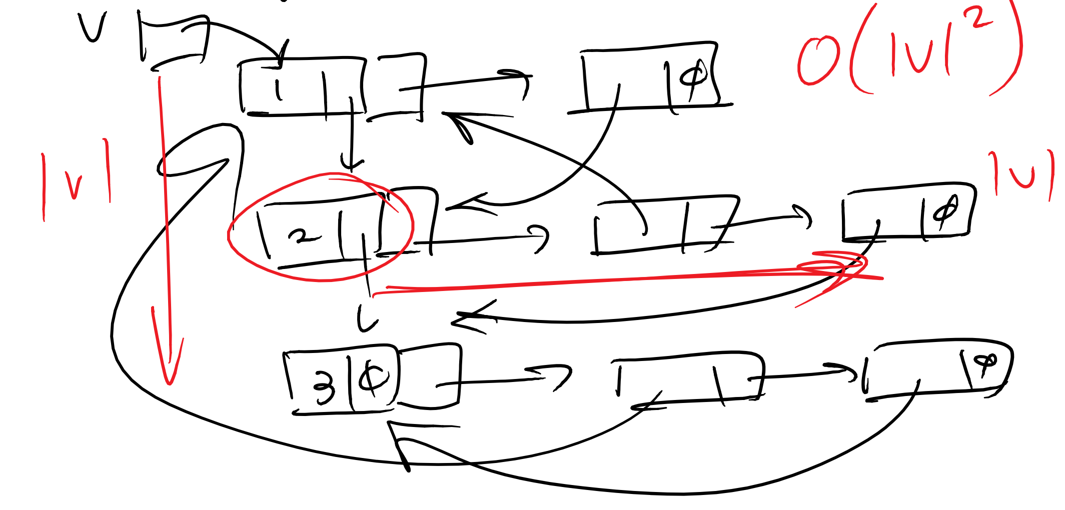
space $O(|V|^2)$

Adjacency List $O(|V|)$

$$\begin{tikzcd} 1 \arrow[r] & \boxed{2 \mid \emptyset} & \\ 2 \arrow[r] & \boxed{1 \mid \ } \arrow[r] & \boxed{3 \mid \emptyset} \\ 3 \arrow[r] & \boxed{1 \mid \ } \arrow[r] & \boxed{3 \mid \emptyset} \end{tikzcd}$$

Space $O(|E|)$ $|E| < |V|^2$

Dynamic Adjacency List



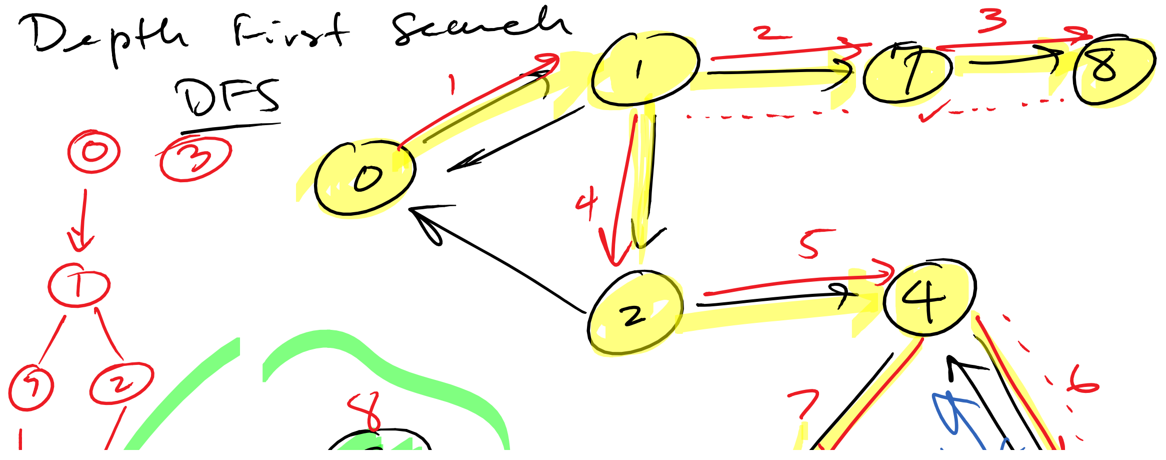
Vertices incident to edges

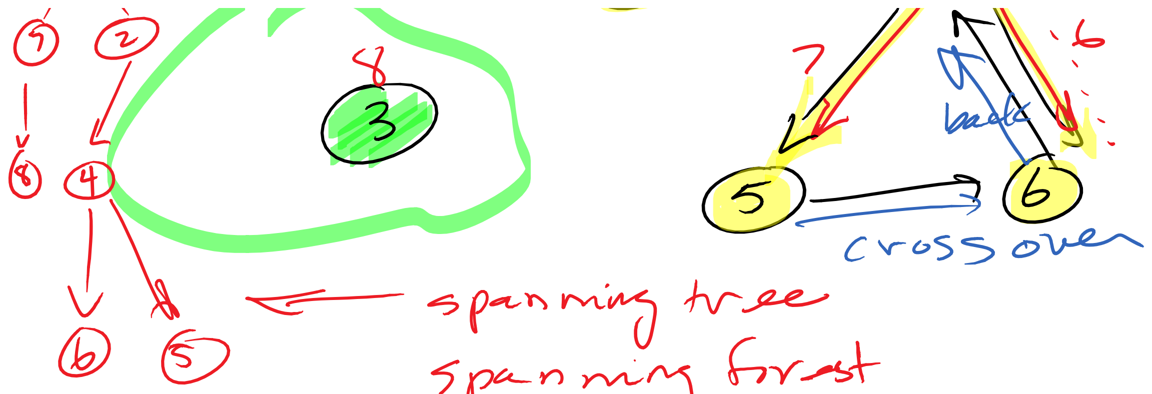
1	1	2
2	2	1
3	2	3
4	3	1
5	3	3

← edges

Depth First Search

DFS





spanning tree
spanning forest

tree edges

back edges
crossover edge

```
#include <vector>
using namespace std;
  void DFSsearch(const vector<vector<bool>> ~~&~~)


int main() {
      vector<vector<bool>> adj =
            {  { 0, 1, 0 },
                 { 1, 0, 1 },
                 { 1, 0, 1 }
            };
      DFSsearch(adj);

}
```

```
}
void DFSsearch (const vector<vector<bool>>& A)
{   vector<bool> visited ( A.size(), 0 );
    for (int v = 0; v < A.size(); v++)
    {   if (!visit[v])
        {   dfs (A, visited, v);
        }
    }
}

void dfs(
 const vector<vector<bool>>& A,
      vector<bool>& visited,
      int v );

      visit[v] = 1;
      for ( int w = 0; w < A.size(); w++)
      {   if ( A[v][w] && !visit[w])
                dfs(w);
      }

}
```