

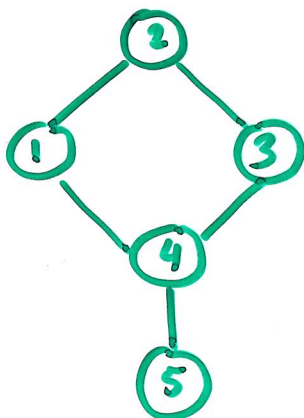
Graph Algorithms

①

Graphs are nodes connected with edges. Trees are graphs.

Adjacency Matrix representation

The vertices are labeled 1 through n . Each edge corresponds to one entry (or two) of the matrix.

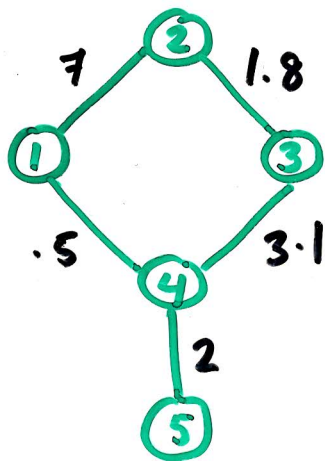


$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(V, E) is undirected if $E \subseteq \binom{V}{2}$ in which case $A = A^T$.

Each undirected edge is equivalent to two directed edges (u, v) and (v, u) .

A graph (V, E) together with a weight function $w: E \rightarrow \mathbb{R}$ is a weighted graph. ②



$$A = \begin{bmatrix} 0 & 7 & \infty & .5 & \infty \\ 7 & 0 & 1.8 & \infty & \infty \\ \infty & 1.8 & 0 & 3.1 & \infty \\ .5 & \infty & 3.1 & 0 & 2 \\ \infty & \infty & \infty & 2 & 0 \end{bmatrix}$$

Non-existing edges are identified with weight ∞ . The diagonal entries are put as 0. This is suitable when the weights are thought as some kind of distances. But, other variations are possible.

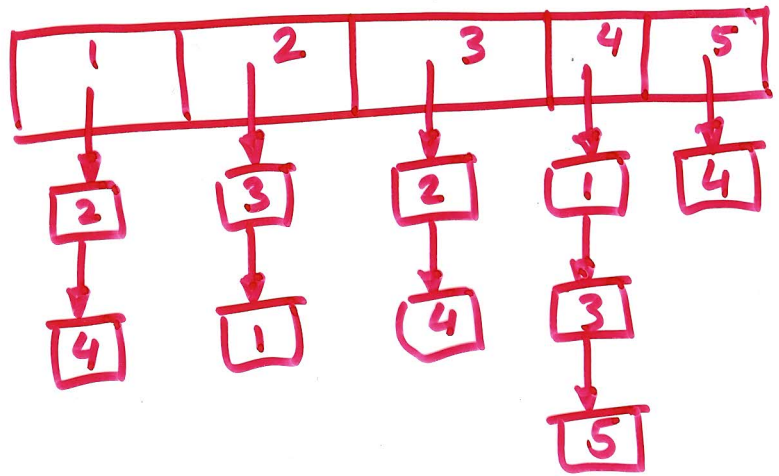
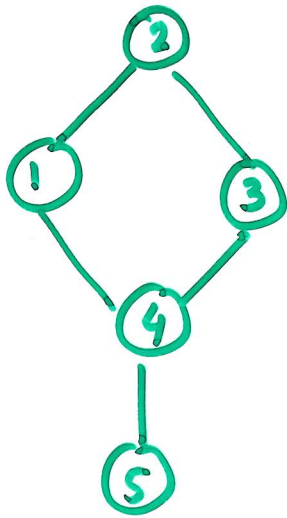
Adjacency Structure

A disadvantage of the adjacency matrix is that it occupies $O(n^2)$ space where $n = |V|$, the number of vertices. A more economic representation is a linked list representation.

③

The adjacency structure of (V, E) consists of a linear array for V plus a linked list of neighbors for each vertex i (i 's adjacency list)

Ex.



It takes $O(n+m)$ space when
 $n = |V|$, $m = |E|$.

Variations

- (1) If (V, E) is undirected, each edge is represented twice. It might be convenient to link the two corresponding nodes.
- (2) If (V, E) is directed, then i 's adjacency list stores only its successors.
- (3) If (V, E) is weighted, then the weight of an edge is stored in the node representing the edge.

Depth First Search

(4)

We use the following data structure

Vertex = record d, f : integer
 π : index

end;

Edge = record v : index;
next : \uparrow Edge

end;

Adj-struct = array $[1 \dots n]$ of Vertex;

V : Adj-struct;

Depth-first-Search (DFS): Traverses all nodes exactly once. The algorithm time stamps each vertex using d (when it is first visited) and f (when all neighbors are finished)

5

```
for i:=1 to n do v[i].d:=0 endfor ; time:=0;
for i:=1 to n do if v[i].d=0 then
    v[i].π:=nil;
    visit(i)
endif
endfor
```

```
procedure Visit(i);
time:=time+1; v[i].d:=time;
t:=v[i].adj;
while t≠nil do
    if v[t.v].d=0 then v[t.v].π:=i;
        visit(t.v)
    endif;
    t:=t.next;
endwhile
time:=time+1; v[i].f:=time;
```

(V and ~~time~~ time are global variable)
 $T(n,m) = O(n+m)$

