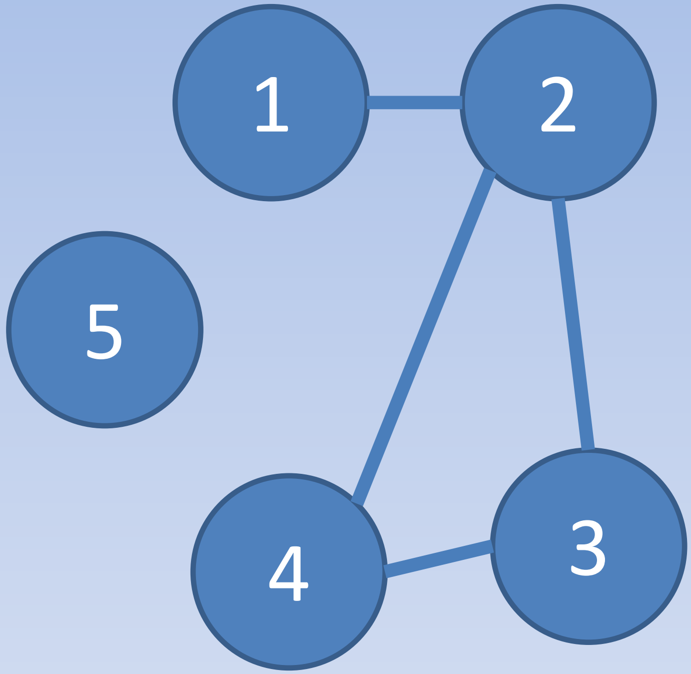
Discrete Mathematics

Spring 2025

Yuan Ze University

Representing graphs

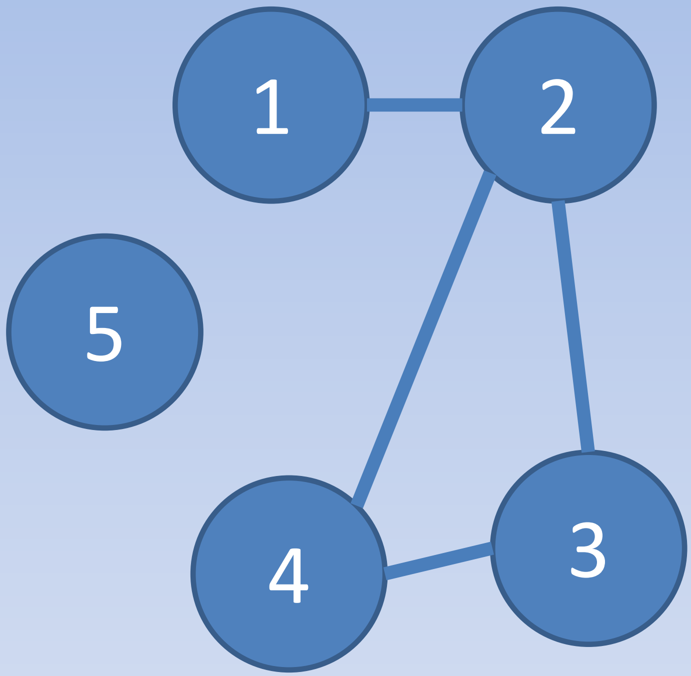
Adjacency list representation



- 1: 2 [redacted]
- 2: 1, 3, 4 [redacted]
- 3: 2, 4 [redacted]
- 4: 2, 3 [redacted]
- 5: [redacted]

Adjacency matrix

	1	2	3	4	5
1	0	1	0	0	0
2	1	0	1	1	0
3	0	1	0	1	0
4	0	1	1	0	0
5	0	0	0	0	0



Our default choice

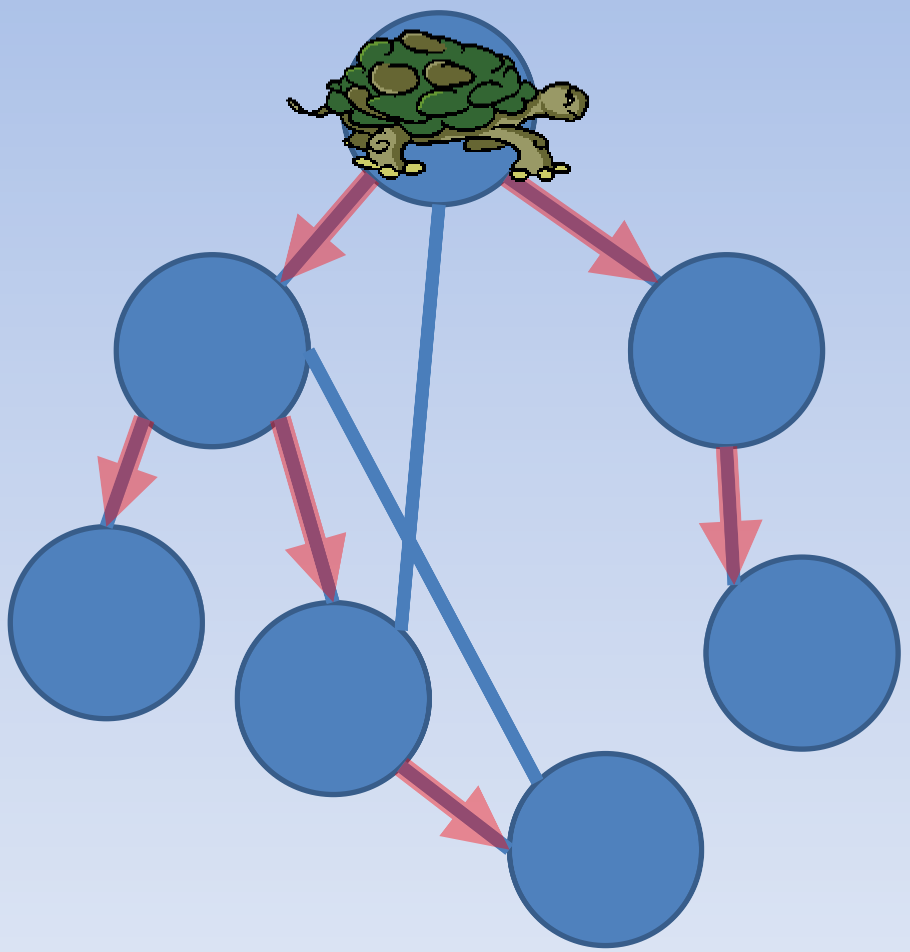
- Adjacency list representation

Depth-first search (DFS)

Our problems

- Given an undirected graph $G = (V, E)$, determine whether G is connected.
- Given an undirected graph $G = (V, E)$ and s, $t \in V$, determine whether there exists an s-t path.

The depth-first search



What does the turtle do?

- Whenever there is a neighbor not yet visited, visit it.
- Go back when all the neighbors are visited.

烏龜心法

- 只要旁邊還有一個沒走過的點，就走過去
- 只要旁邊每個點都走過了，就回頭

Pseudocode

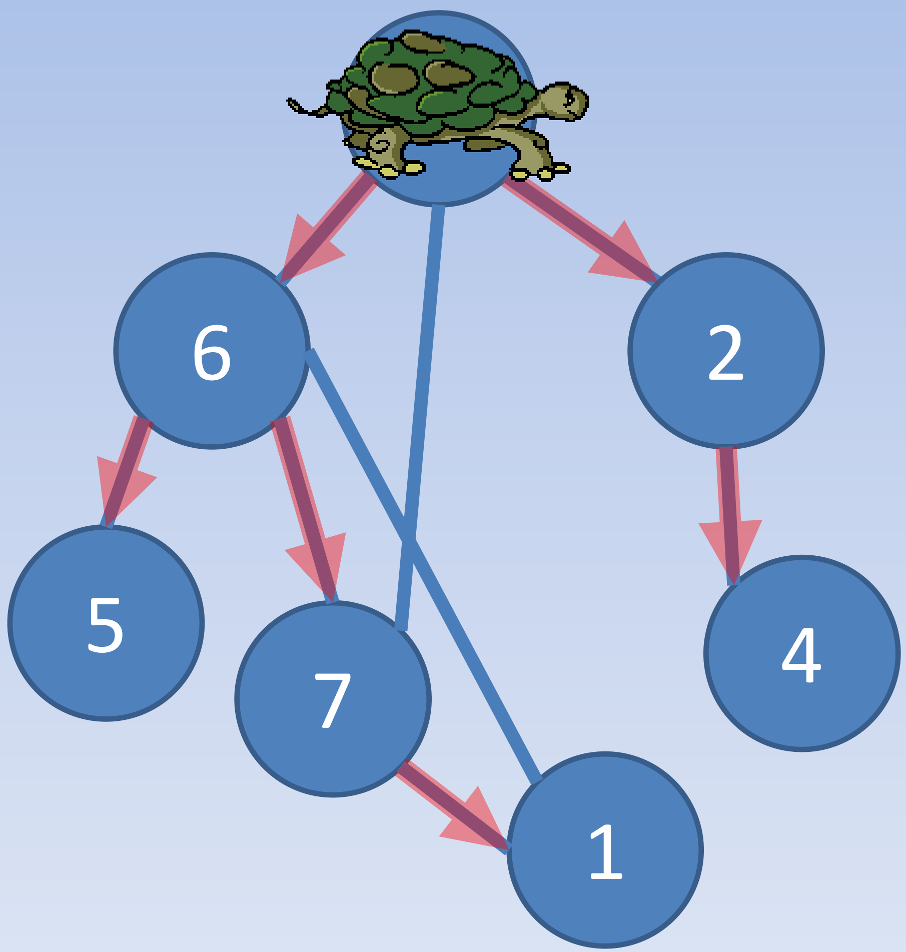
Algorithm DFS

Input: Undirected graph $G = (V, E)$ and $v \in V$

Label v as visited;
for all $u \in N(v)$ **do**
  if u is not labeled as visited **then**
    DFS(G, u);
  end if
end for

- G can be a global variable…
- Or a pointer to the actual graph

Function calls ☺



Calling DFS$(G, 3)\ldots$

  Calling DFS$(G, 6)\ldots$

    Calling DFS$(G, 5)\ldots$

    Returning...

    Calling DFS$(G, 7)\ldots$

      Calling DFS$(G, 1)\ldots$

      Returning...

    Returning...

  Returning...

  Calling DFS$(G, 2)\ldots$

    Calling DFS$(G, 4)\ldots$

    Returning...

  Returning...

Returning...

Running time ☺

Algorithm DFS

Input: Undirected graph $G = (V, E)$ and $v \in V$

Label v as visited;
for all $u \in N(v)$ **do**
    if u is not labeled as visited **then**
        DFS(G, u);
    end if
end for

- Calling DFS(G, v) takes $\leq 3\deg(v) + 2$ time, excluding the recursive calls...
- DFS(G, v) is called at most once for each $v \in V$ (Why?)...
- So the running time is $O(\sum_{v \in V} (3\deg(v)+2)) = O(|V|+|E|)$.

A short explanation… ☺

Algorithm DFS

Input: Undirected graph $G = (V, E)$ and $v \in V$

```
1: Label v as visited;
2: for all u ∈ N(v) do
3:     if u is not labeled as visited then
4:         DFS(G, u);
5:     end if
6: end for
```

Why is $\mathrm{DFS}(G, v)$ called at most once for each $v \in V$?

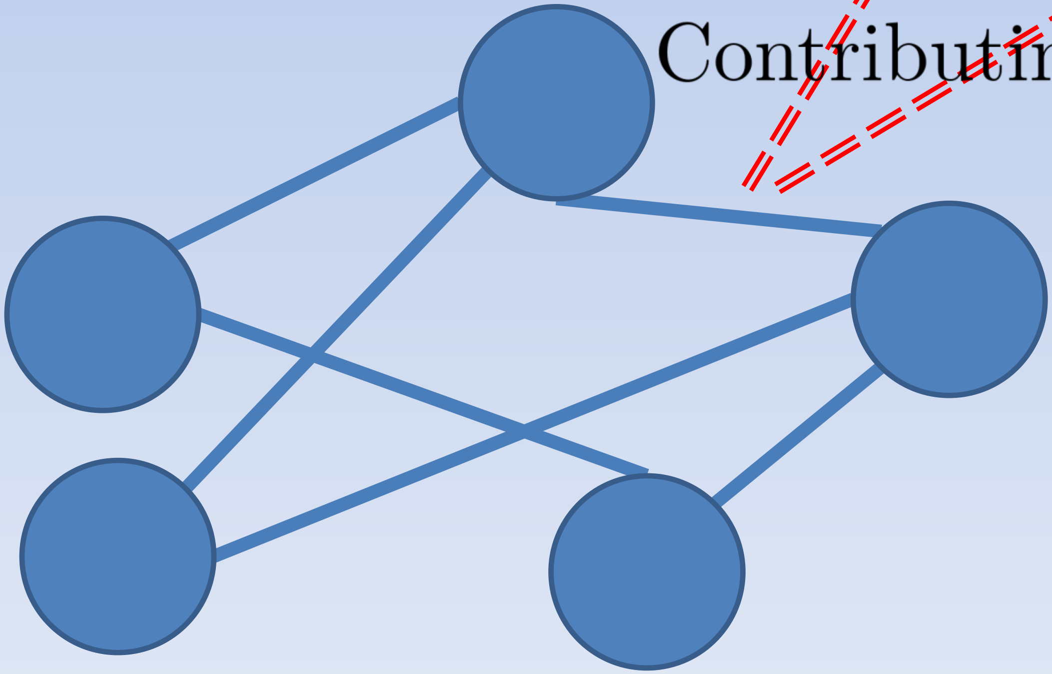
- Calling $\mathrm{DFS}(G, v)$ labels v as visited, forbidding subsequent executions of lines 3–4 from calling $\mathrm{DFS}(G, v)$.

The handshaking lemma

Theorem. *For each undirected graph* $G = (V, E)$,

$$\sum_{v \in V} \deg(v) = 2 \cdot |E|.$$

Contributing 2 to both sides...



So…

The depth-first search runs in $O(|V| + |E|)$ time.

Which vertices are visited?

Algorithm DFS

Input: Undirected graph $G = (V, E)$ and $v \in V$

```
Label v as visited;
for all u ∈ N(v) do
    if u is not labeled as visited then
        DFS(G, u);
    end if
end for
```

Lemma. *Suppose that a call to* DFS *calls* DFS(G, v) *(possibly recursively), where* $v \in V$. *Then when* DFS(G, v) *returns, all neighbors of* v *are labeled as visited.*

Which vertices are visited?

Theorem. *Given an undirected graph* $G = (V, E)$ *and* $v \in V$, $\mathrm{DFS}(G, v)$ *visits precisely the vertices in* v*'s connected component.*

Theorem. *Given an undirected graph* $G = (V, E)$ *and* $u, v \in V$ *such that* v *is reachable from* u, $\mathrm{DFS}(G, u)$ *visits* v *(sooner or later).*

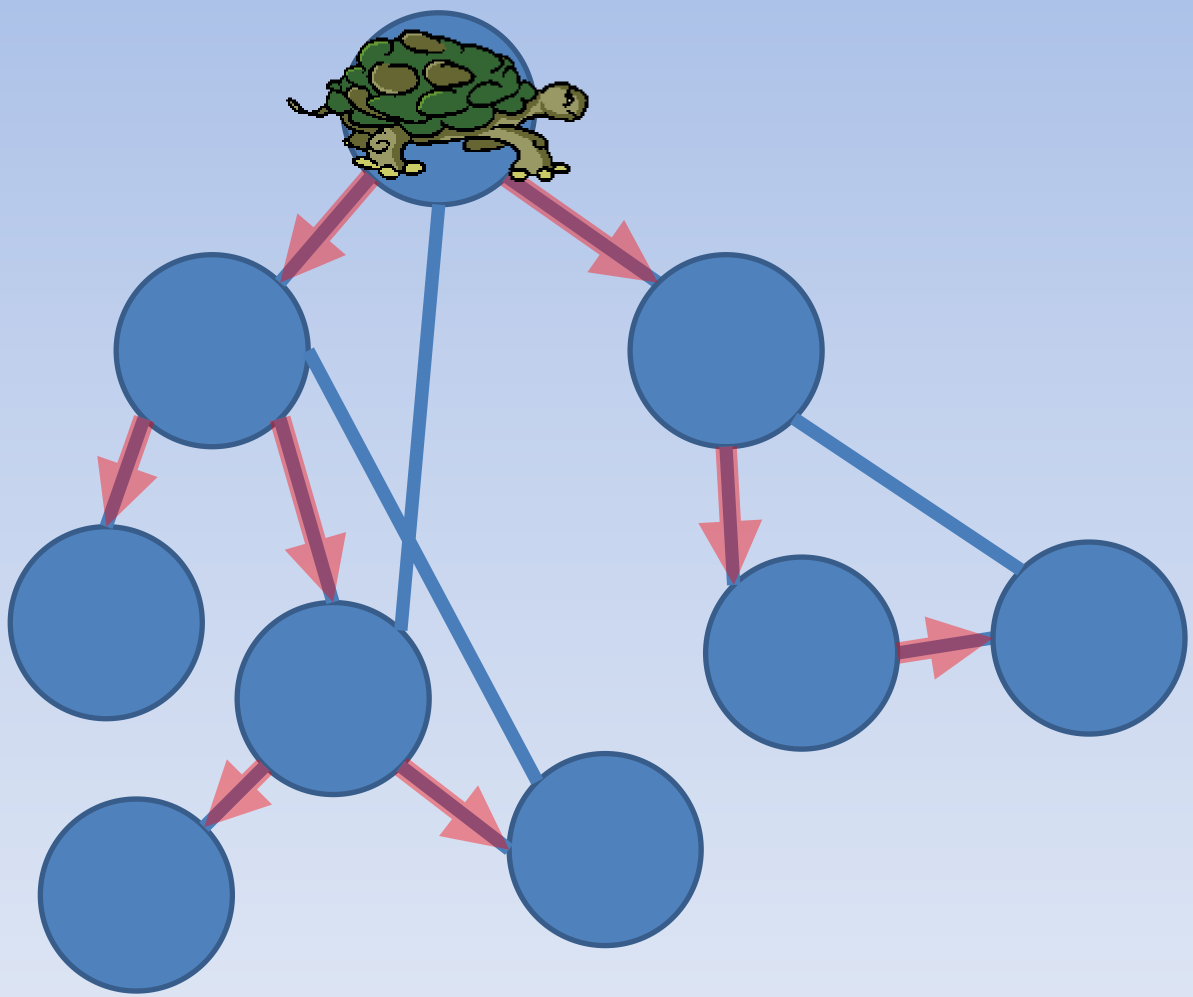
All solved!

- Given an undirected graph $G = (V, E)$, determine whether G is connected.
- Given an undirected graph $G = (V, E)$ and s, $t \in V$, determine whether there exists an s-t path.

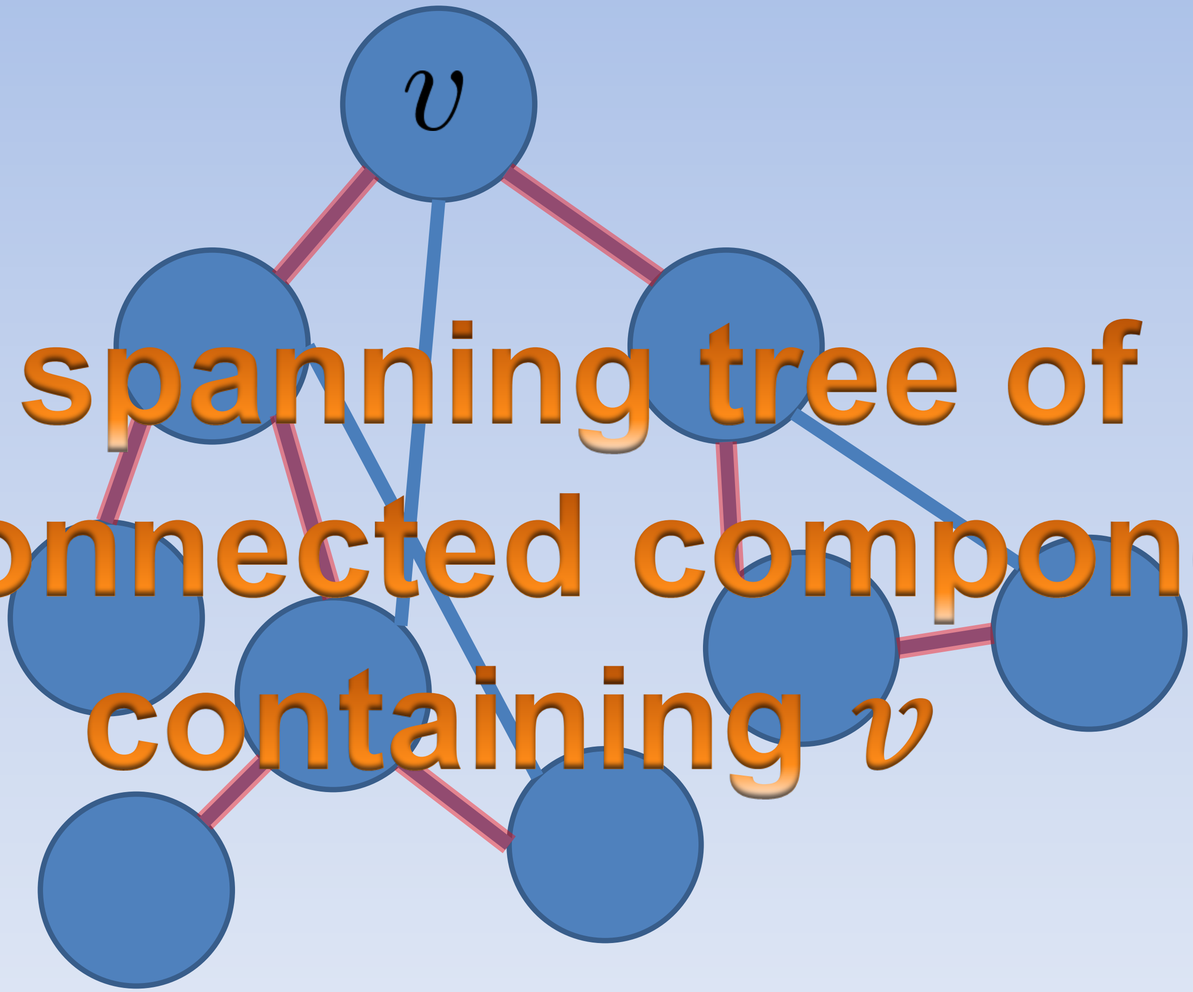
Both can be done in $O(|V| + |E|)$ time!

DFS tree

The visited forms a tree



By-product: A spanning tree



A spanning tree of the connected component containing v

Marking the visited edges

Algorithm DFS

Input: Undirected graph $G = (V, E)$ and $v \in V$

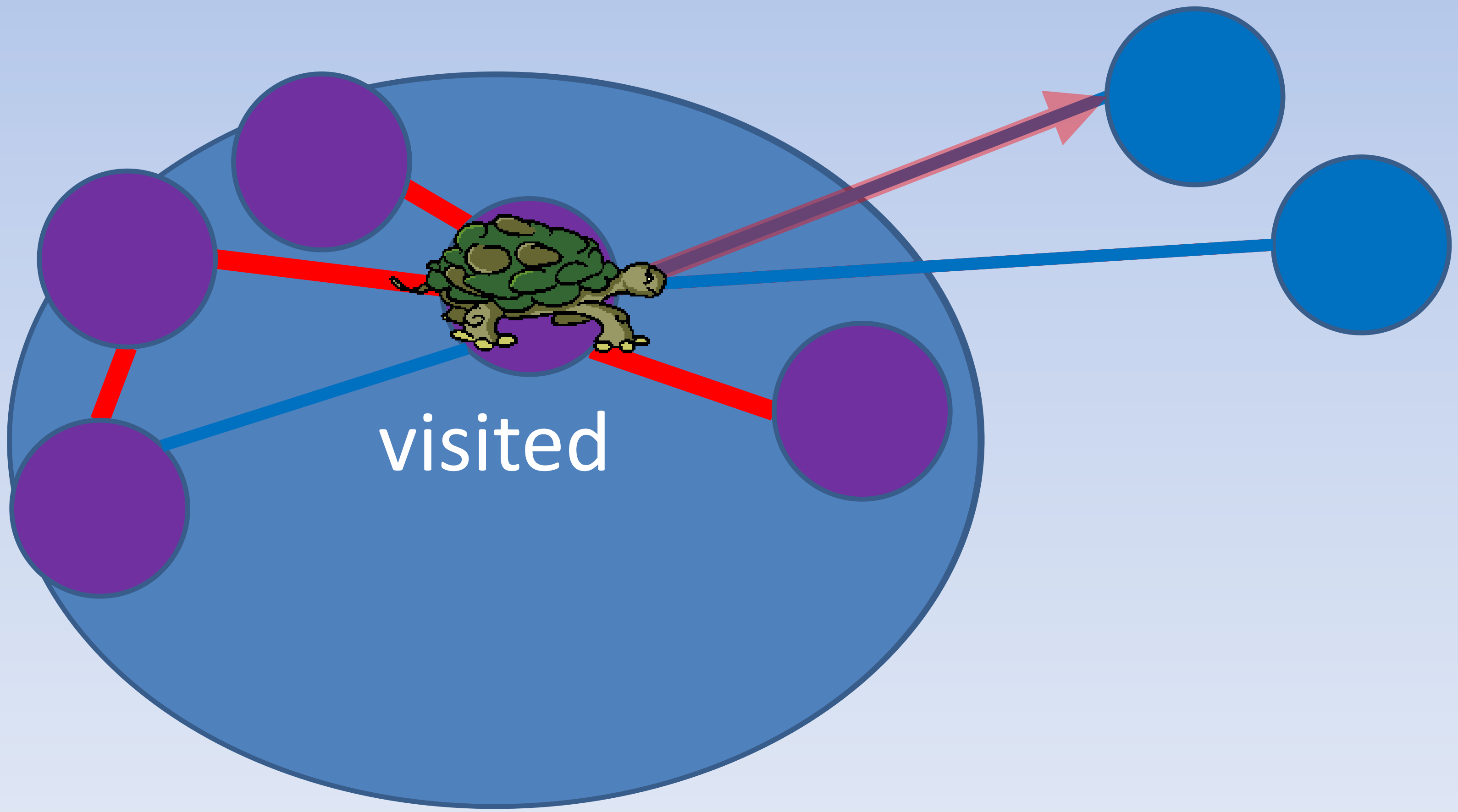
```
Label v as visited;
for all u ∈ N(v) do
    if u is not labeled as visited then
        Mark the edge (v, u);
        DFS(G, u);
    end if
end for
```

Theorem

- At the end of the depth-first search, the visited vertices together with the marked edges form a tree (i.e., a connected acyclic graph).
- The next slide illustrates this.

Growing a tree

Fact. *Adding to a tree a new vertex v and an edge with exactly one endpoint being v yields a tree.*



No cross edges!

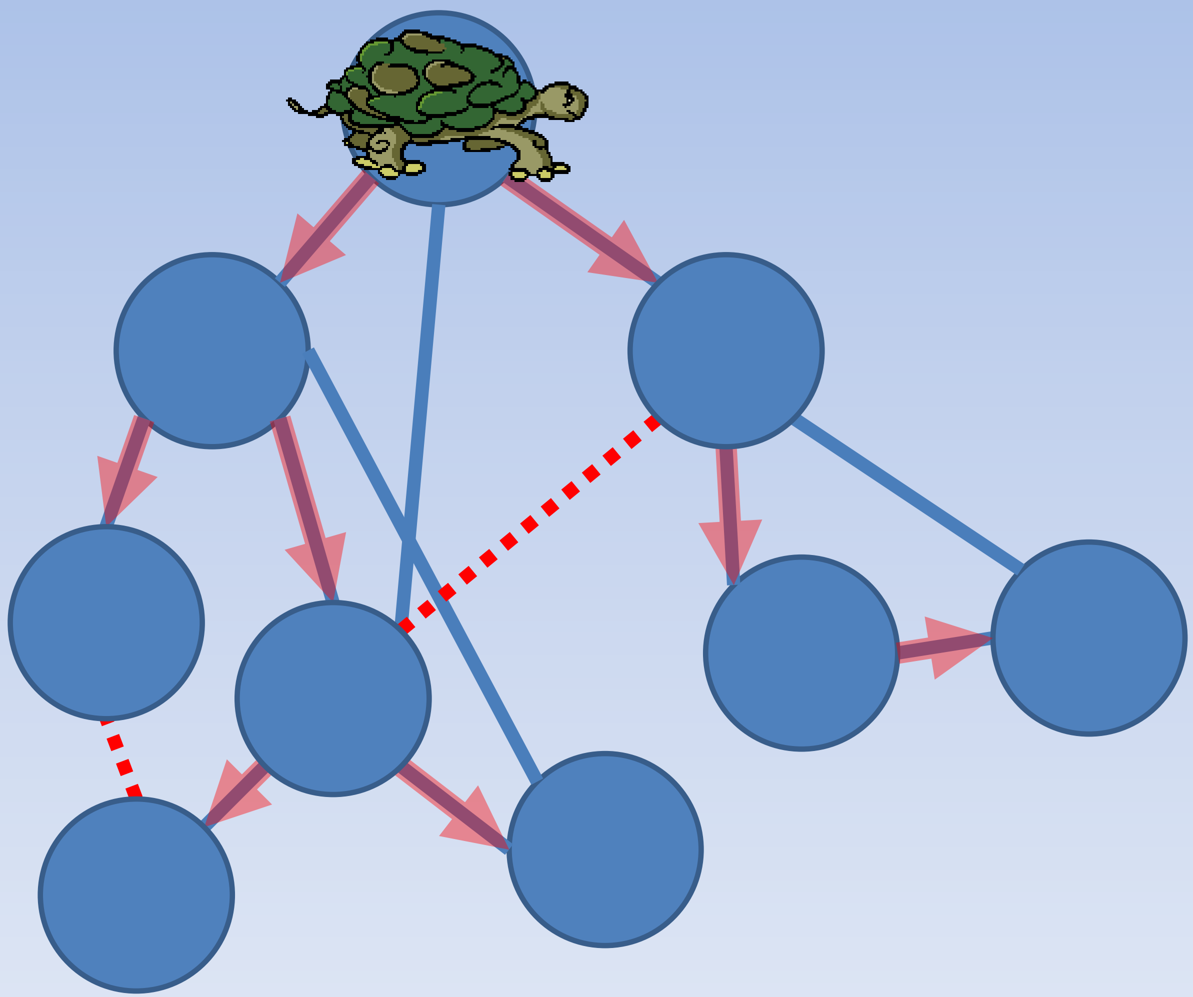
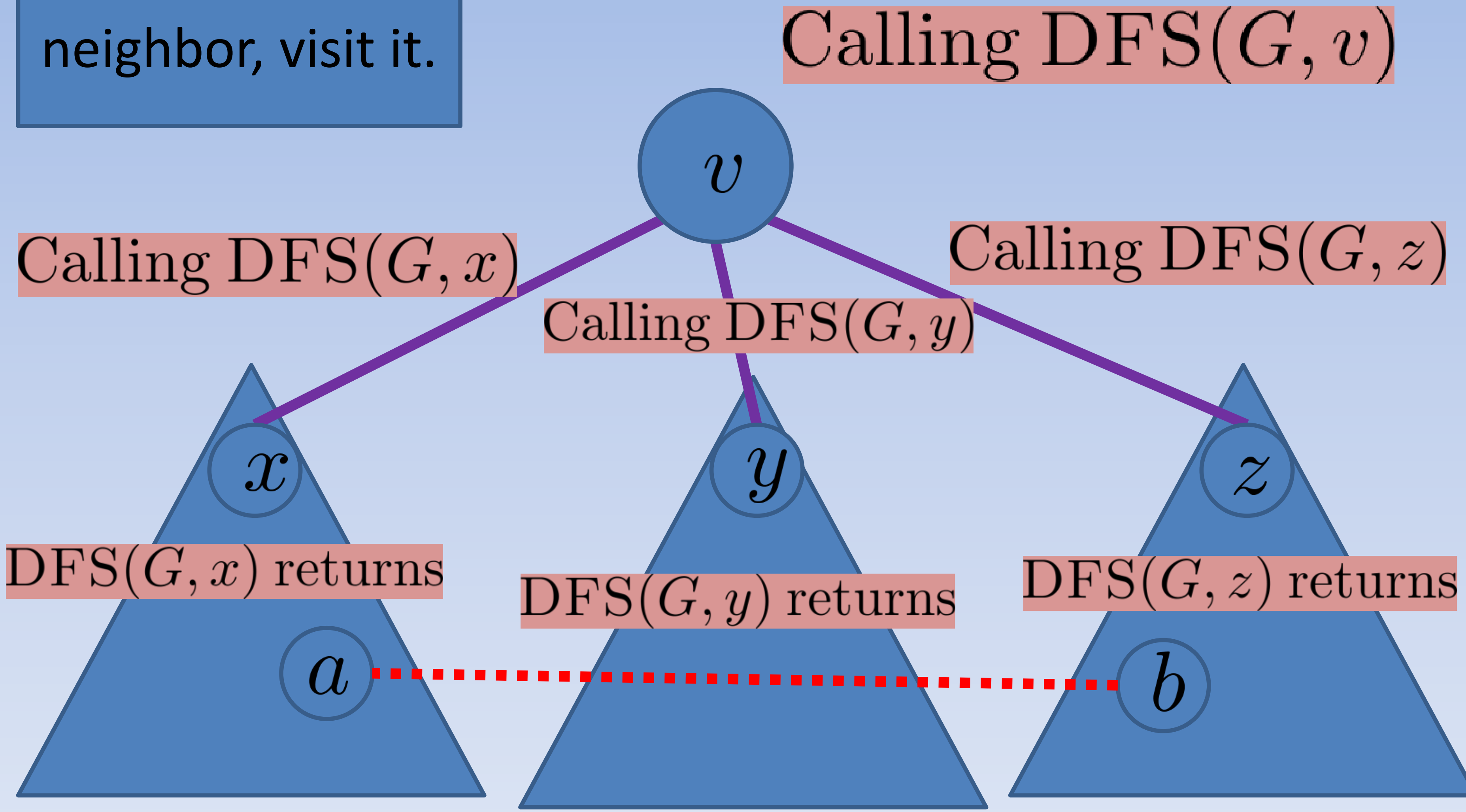


Illustration ☺

While v has an unvisited neighbor, visit it.



The order of function calls ☺

Calling $\mathrm{DFS}(G, x)$

$\ldots$ Calling $\mathrm{DFS}(G, a)$ $\ldots$ $\mathrm{DFS}(G, a)$ returning

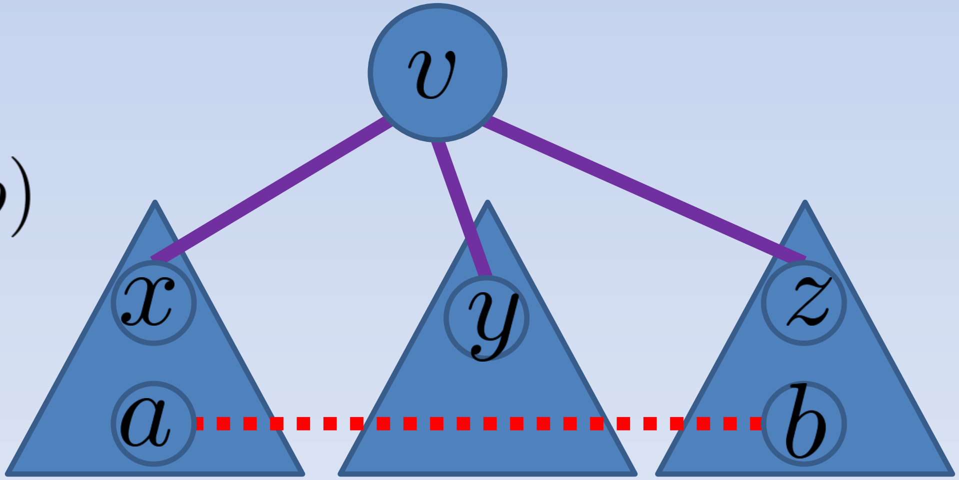
$\mathrm{DFS}(G, x)$ returning

Calling $\mathrm{DFS}(G, y) \ldots \mathrm{DFS}(G, y)$ returning

Calling $\mathrm{DFS}(G, z)$

$\ldots$ Calling $\mathrm{DFS}(G, b)$

$\mathrm{DFS}(G, z)$ returning



But when $\mathrm{DFS}(G, a)$ returns, b must have been traversed!

A recap

Algorithm DFS

Input: Undirected graph $G = (V, E)$ and $v \in V$

Label v as visited;
for all $u \in N(v)$ **do**
    if u is not labeled as visited **then**
        $\mathrm{DFS}(G, u)$;
    end if
end for

Lemma. *Suppose that a call to* DFS *calls* $\mathrm{DFS}(G, v)$ *(possibly recursively), where* $v \in V$. *Then when* $\mathrm{DFS}(G, v)$ *returns, all neighbors of* v *are labeled as visited.*

Quiz

Define a simple undirected graph $G = (V, E)$ by

$$\begin{aligned} V &= \{1, 2, 3, 4, 5, 6\}, \\ E &= \{(1,2), (1,6), (2,3), (2,4), (5,4), (5,6), (6,2)\}. \end{aligned}$$

Running $\mathrm{DFS}(G, 1)$ labels the vertices in V as visited in the order of ____________________. Please fill the blank with an ordered list of the elements in V.

Remark: There may be many correct answers, but you only need to give one of them.

Quiz

Suppose that G has a path visiting a, b, c and d, in that order.

- Is it necessarily true that DFS(a) visits b before c?
- Is it necessarily true that DFS(a) visits c before d?
- Is it necessarily true that DFS(a) visits b at some time?
- Is it necessarily true that DFS(a) visits c at some time?
- Is it necessarily true that DFS(a) visits d at some time?

A question

- Can we save lots of space, while possibly wasting time, in determining whether a graph is connected?

Algorithm PATH

Input: vertices v_1, $v_2 \in V$, $i \geq 1$

```
if i = 1 then
    if (v1, v2) ∈ E or v1 = v2 then
        return true;
    else
        return false;
    end if
else
    for each v ∈ V do
        if PATH(v1, v, ⌈i/2⌉) ∧ PATH(v, v2, ⌊i/2⌋) then
            return true;
        end if
    end for
end if
return false;
```

Quiz

Show that given a simple undirected graph $G = (V, E)$ and v_1, $v_2 \in V$, the existence of a v_1-v_2 path in G can be determined in worst-case $O(\sqrt{|V|})$ space.

- Hint: See the previous page

Comments?