CS473-Algorithms I

Lecture 15

Graph Searching:

Depth-First Search and Topological Sort

DFS: Parenthesis Theorem

Thm: In any DFS of G=(V,E), let int[v] = [d[v], f[v]] then exactly one of the following holds for any u and $v \in V$

- int[u] and int[v] are entirely disjoint
- int[v] is entirely contained in int[u] and v is a descendant of u in a DFT
- int[u] is entirely contained in int[v] and u is a descendant of v in a DFT

Parenthesis Thm
(proof for the case d[u] < d[v])

<u>Subcase</u> d[v] < f[u] (int[u] and int[v] are overlapping)

- v was discovered while u was still GRAY
- This implies that v is a descendant of u
- So search returns back to u and finishes u after finishing v
- i.e., d[v]< f[u] $\Rightarrow$int[v] is entirely contained in int[u]

<u>Subcase</u> d[v] >f[u] $\Rightarrow$int[v] and int[u] are entirely disjoint

Proof for the case d[v] < d[u] is similar (dual)

QED

Nesting of Descendents' Intervals

Corollary 1 (Nesting of Descendents' Intervals):

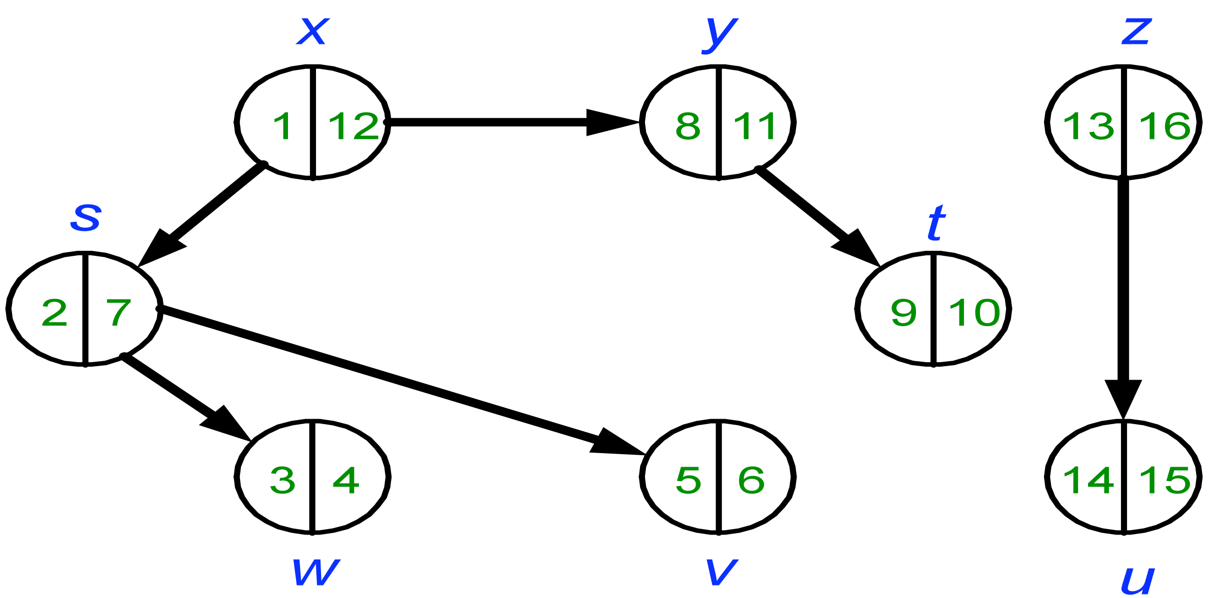
v is a descendant of u if and only if

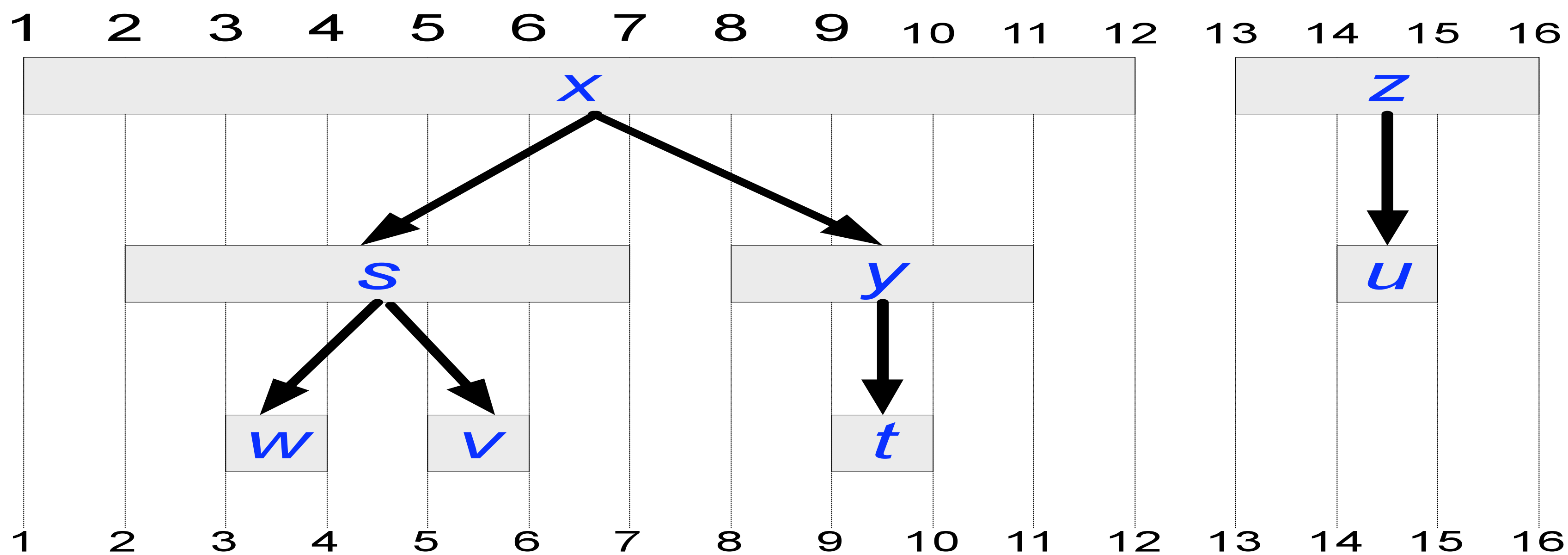
$$d[u] < d[v] < f[v] < f[u]$$

Proof: immediate from the Parenthesis Thrm

QED

Parenthesis Theorem





(x (s (w w) (v v) s) (y (t t) y) x) (z (u u) z)

Edge Classification in a DFF

Tree Edge: discover a new (WHITE) vertex

▷GRAY to WHITE◁

Back Edge: from a descendent to an ancestor in DFT

▷GRAY to GRAY◁

Forward Edge: from ancestor to descendent in DFT

▷GRAY to BLACK◁

Cross Edge: remaining edges (btwn trees and subtrees)

▷GRAY to BLACK◁

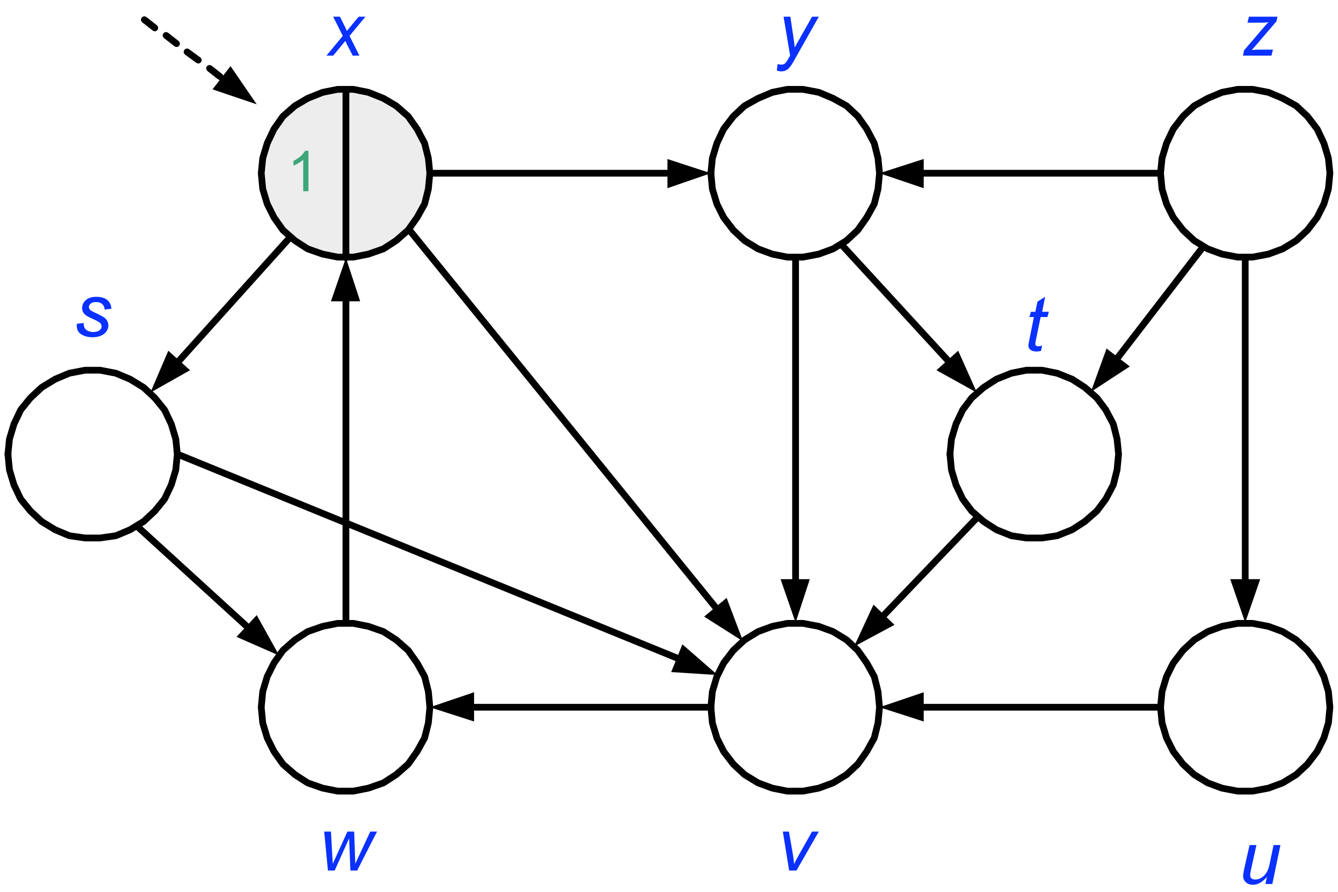
Note: ancestor/descendent is wrt Tree Edges

Edge Classification in a DFF

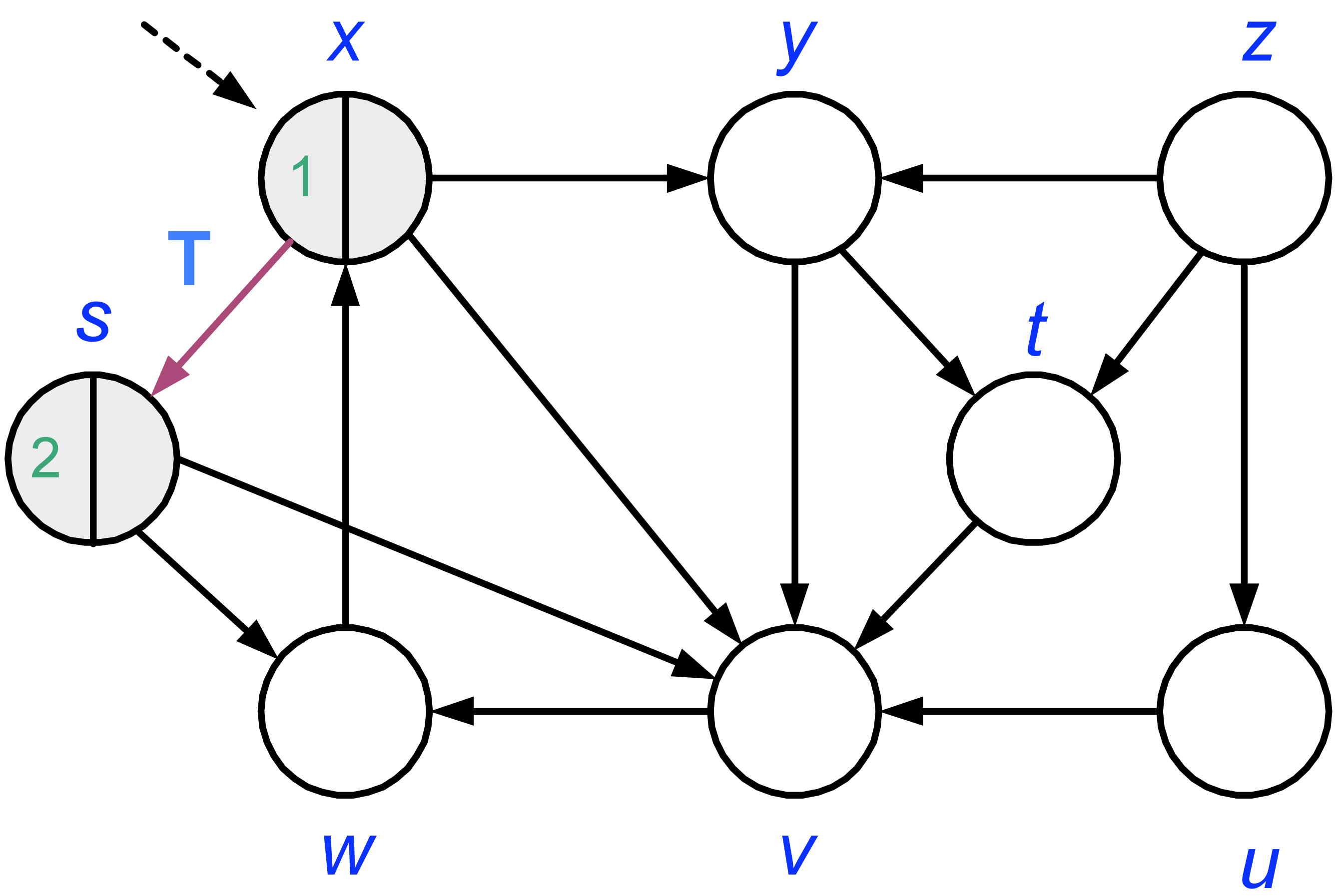
- How to decide which GRAY to BLACK edges are forward, which are cross

  Let BLACK vertex $v \in \text{Adj}[u]$ is encountered while processing GRAY vertex u
  - (u,v) is a forward edge if $d[u] < d[v]$
  - (u,v) is a cross edge if $d[u] > d[v]$

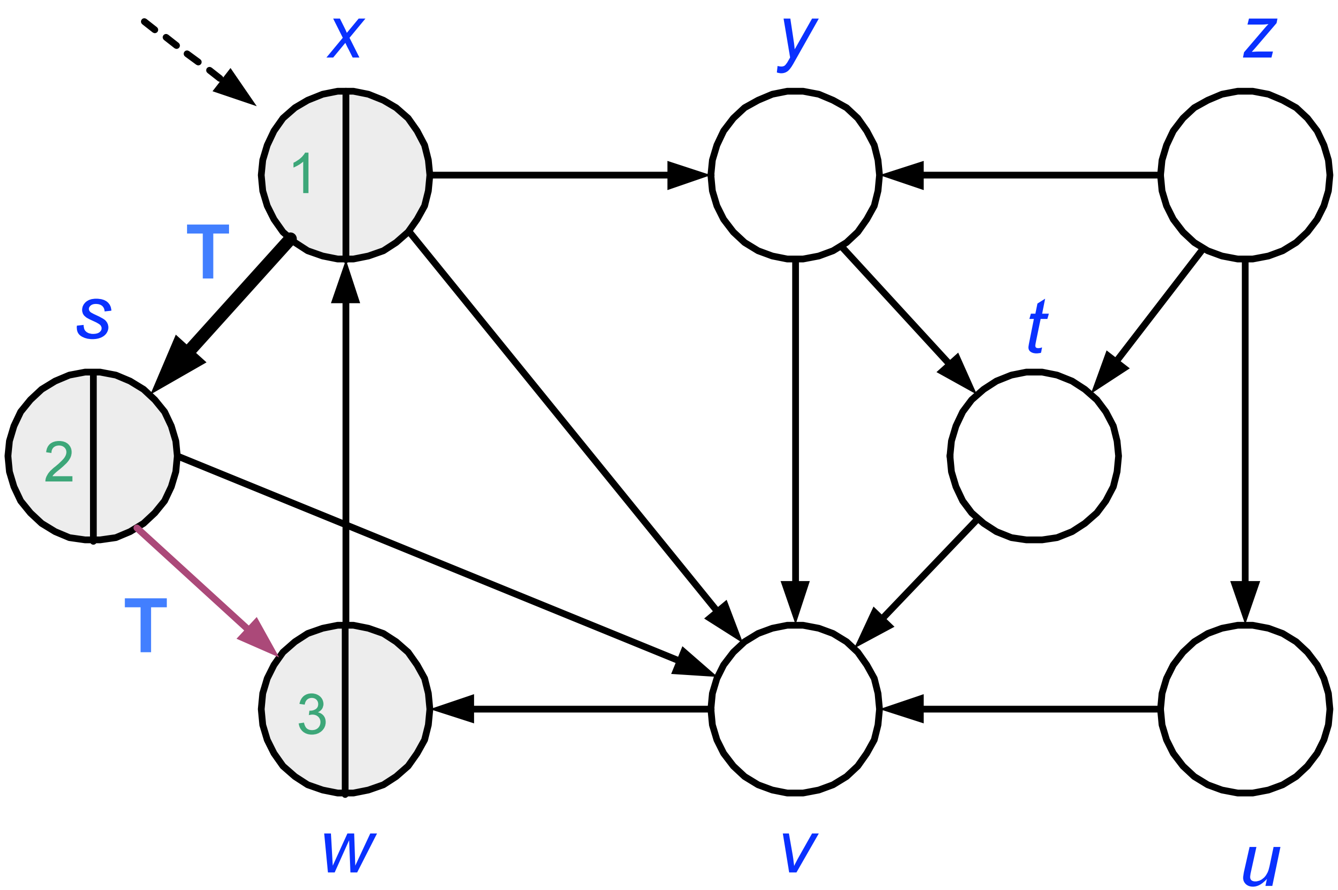
Depth-First Search: Example



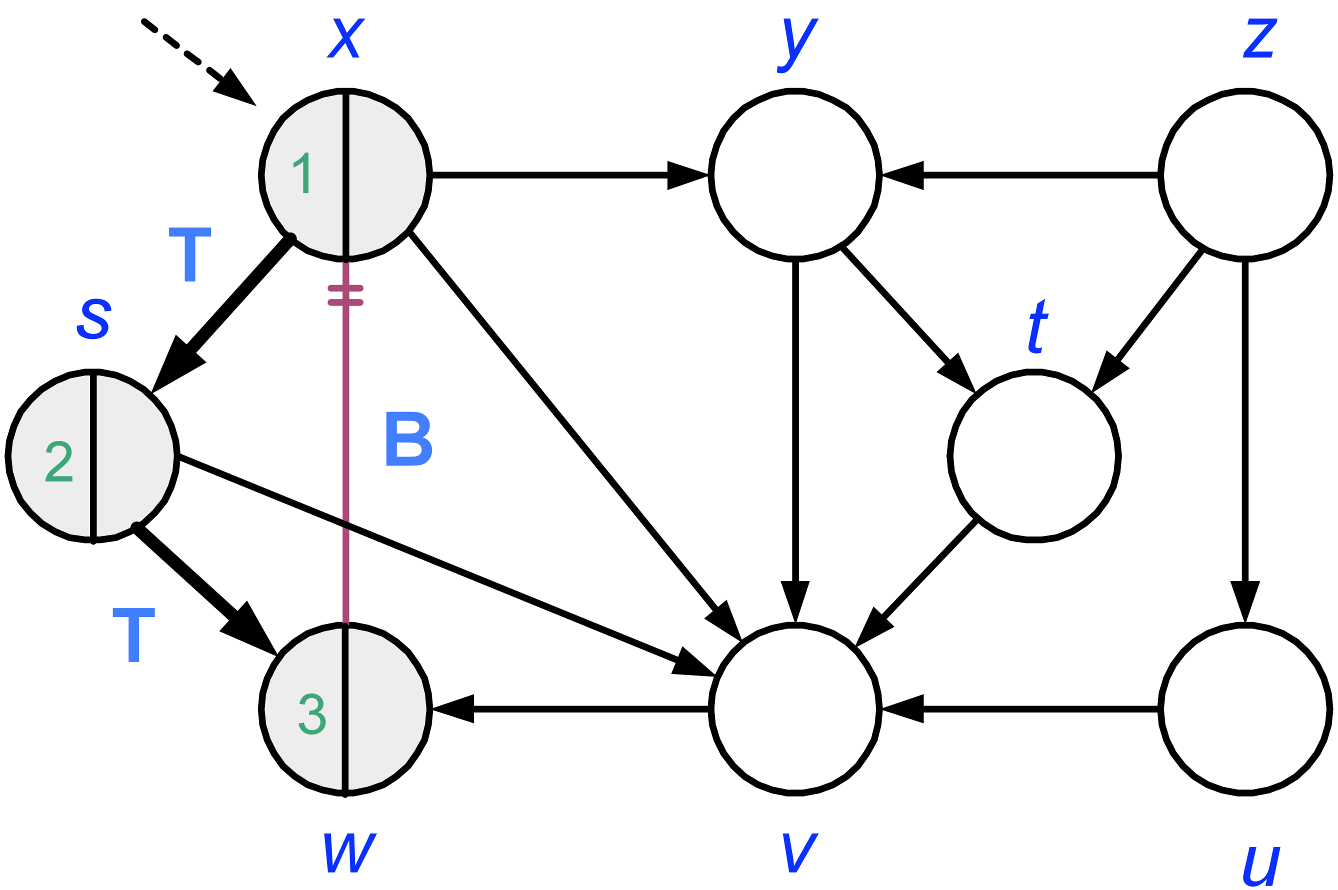
Depth-First Search: Example



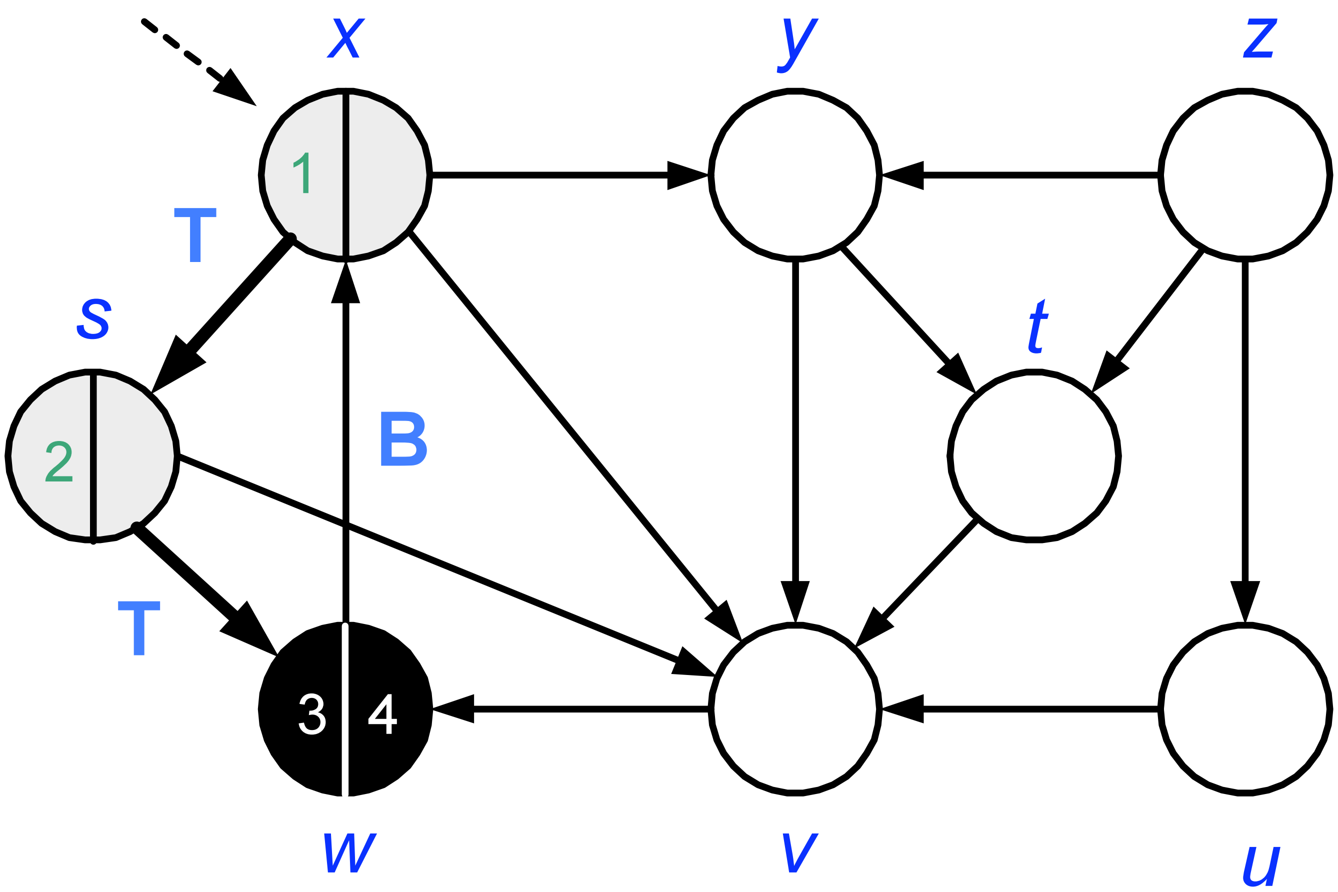
Depth-First Search: Example



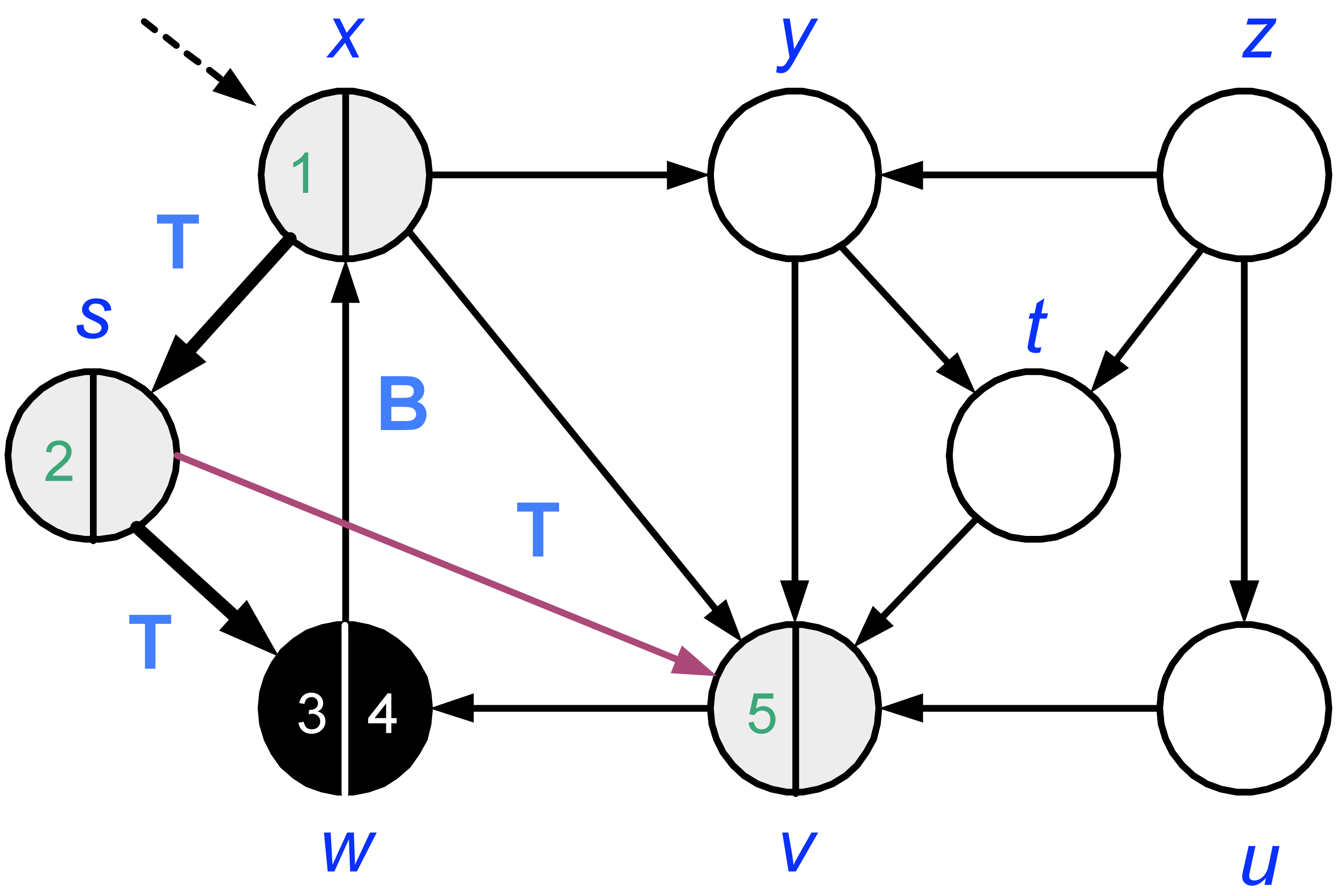
Depth-First Search: Example



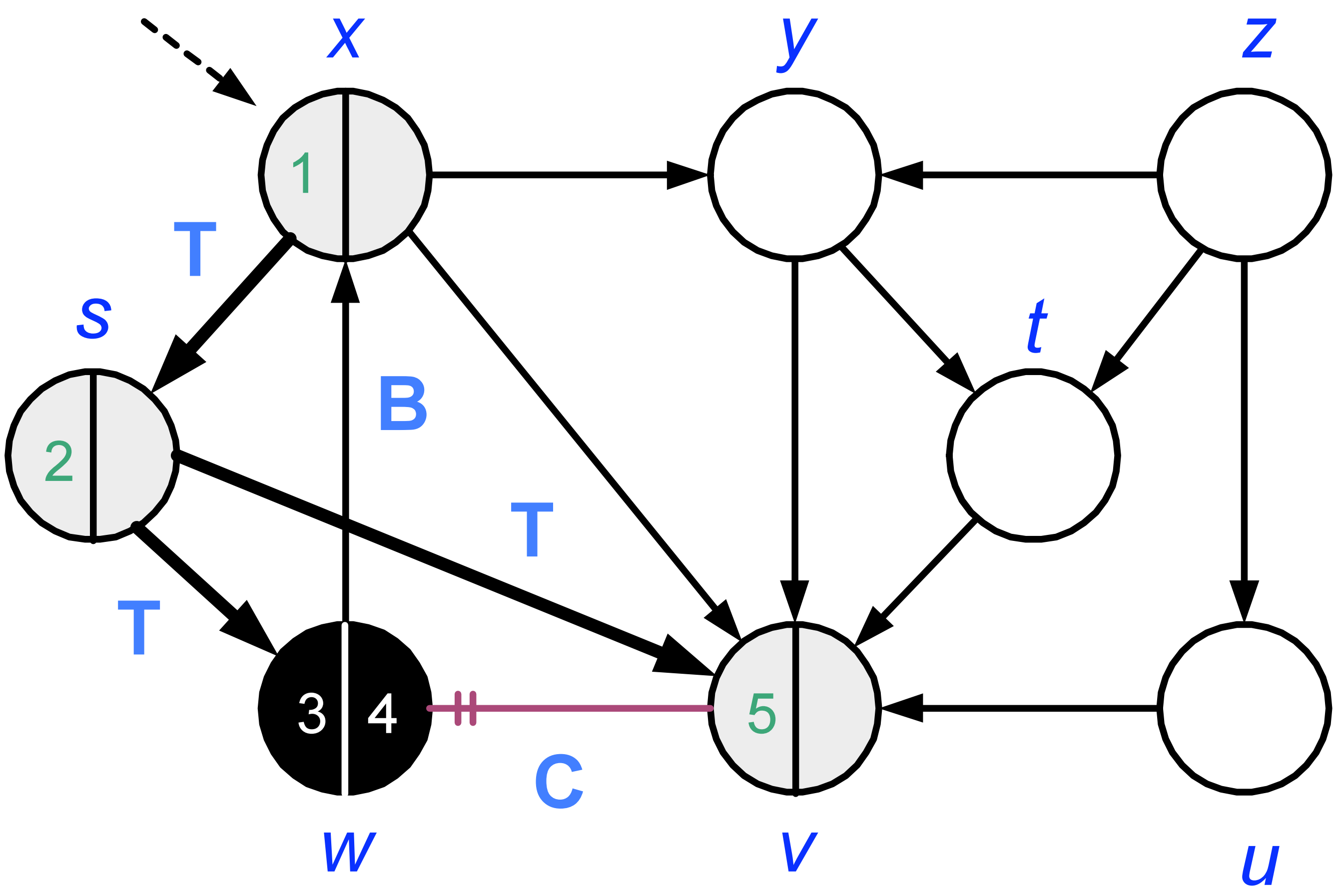
Depth-First Search: Example



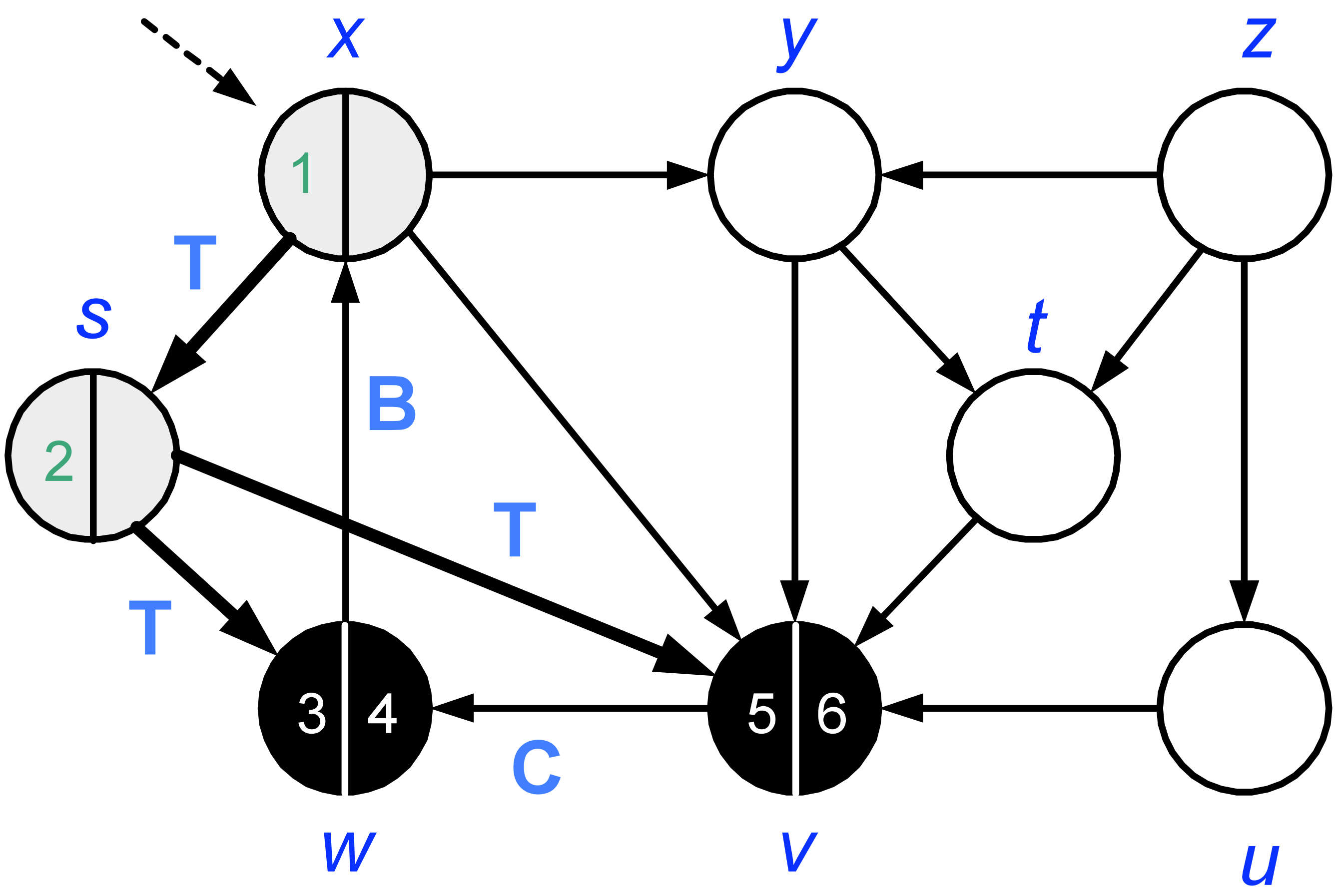
Depth-First Search: Example



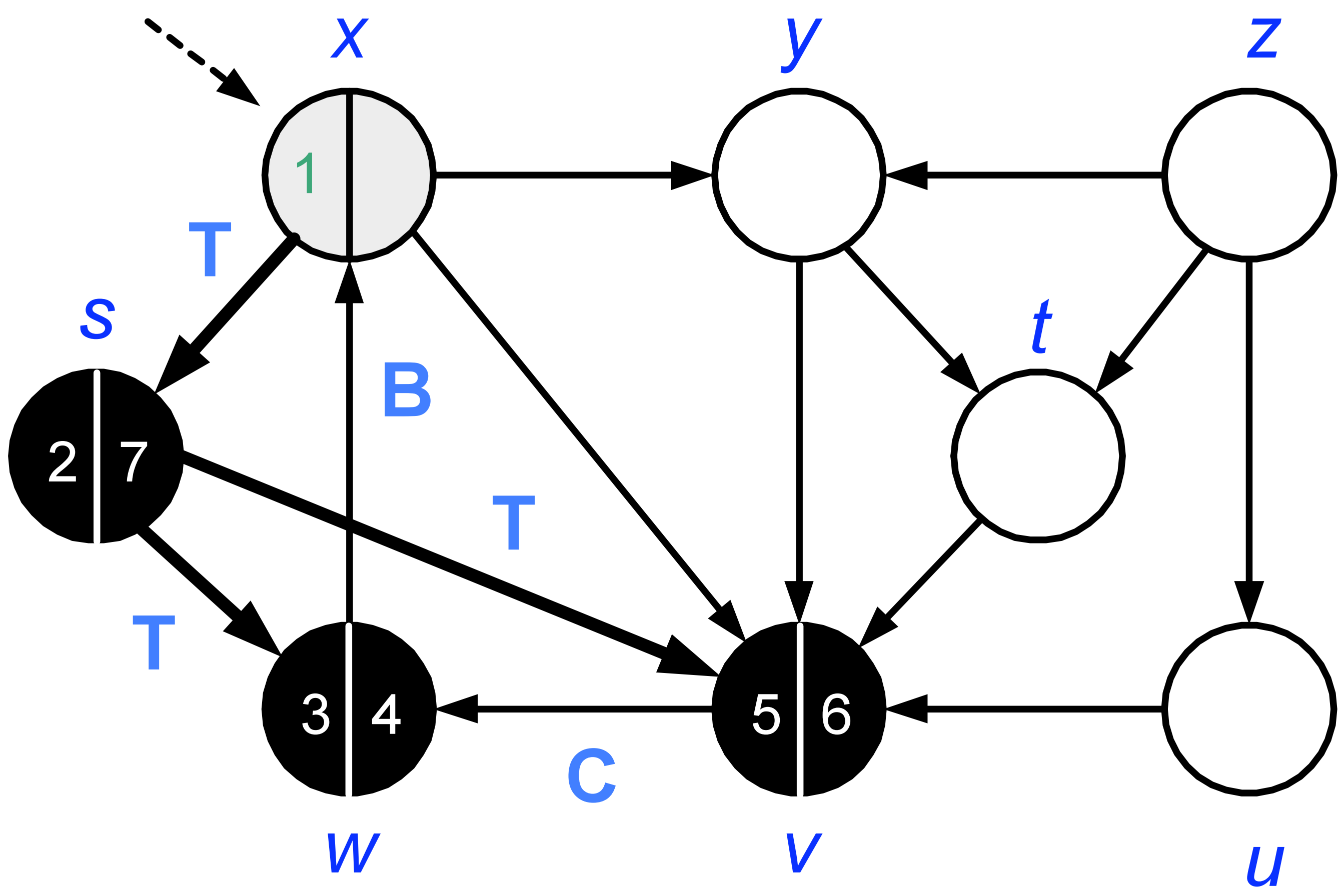
Depth-First Search: Example



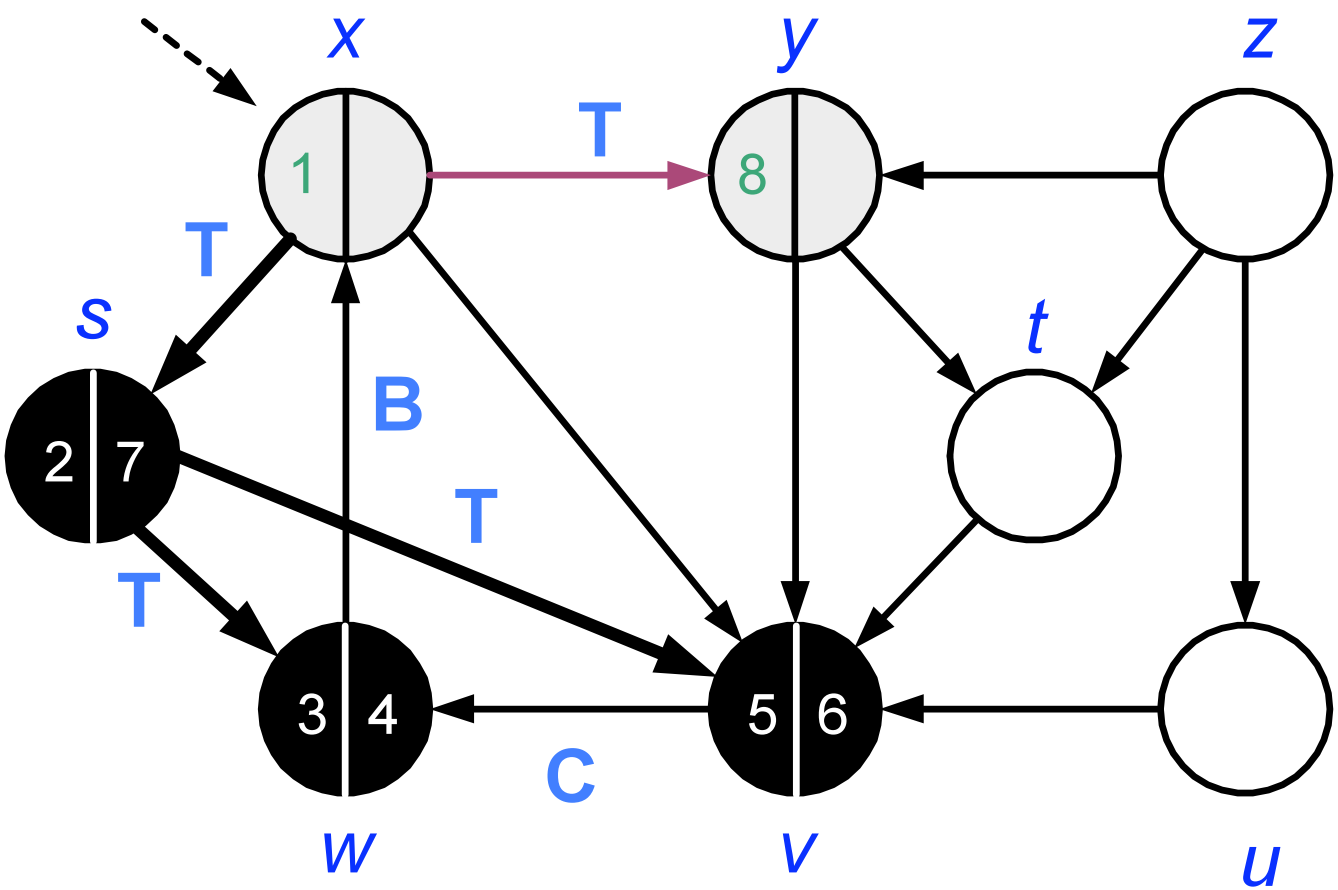
Depth-First Search: Example



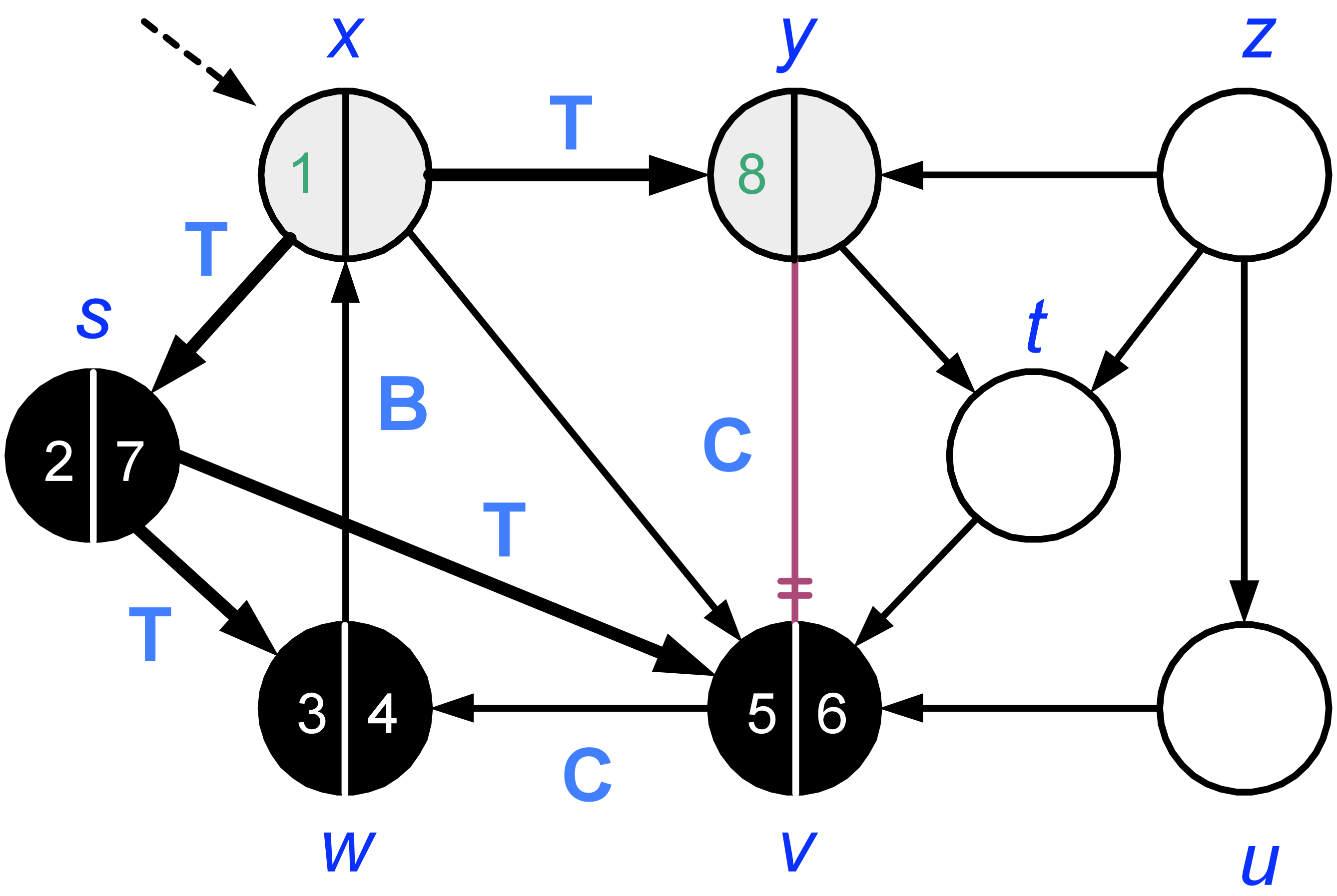
Depth-First Search: Example



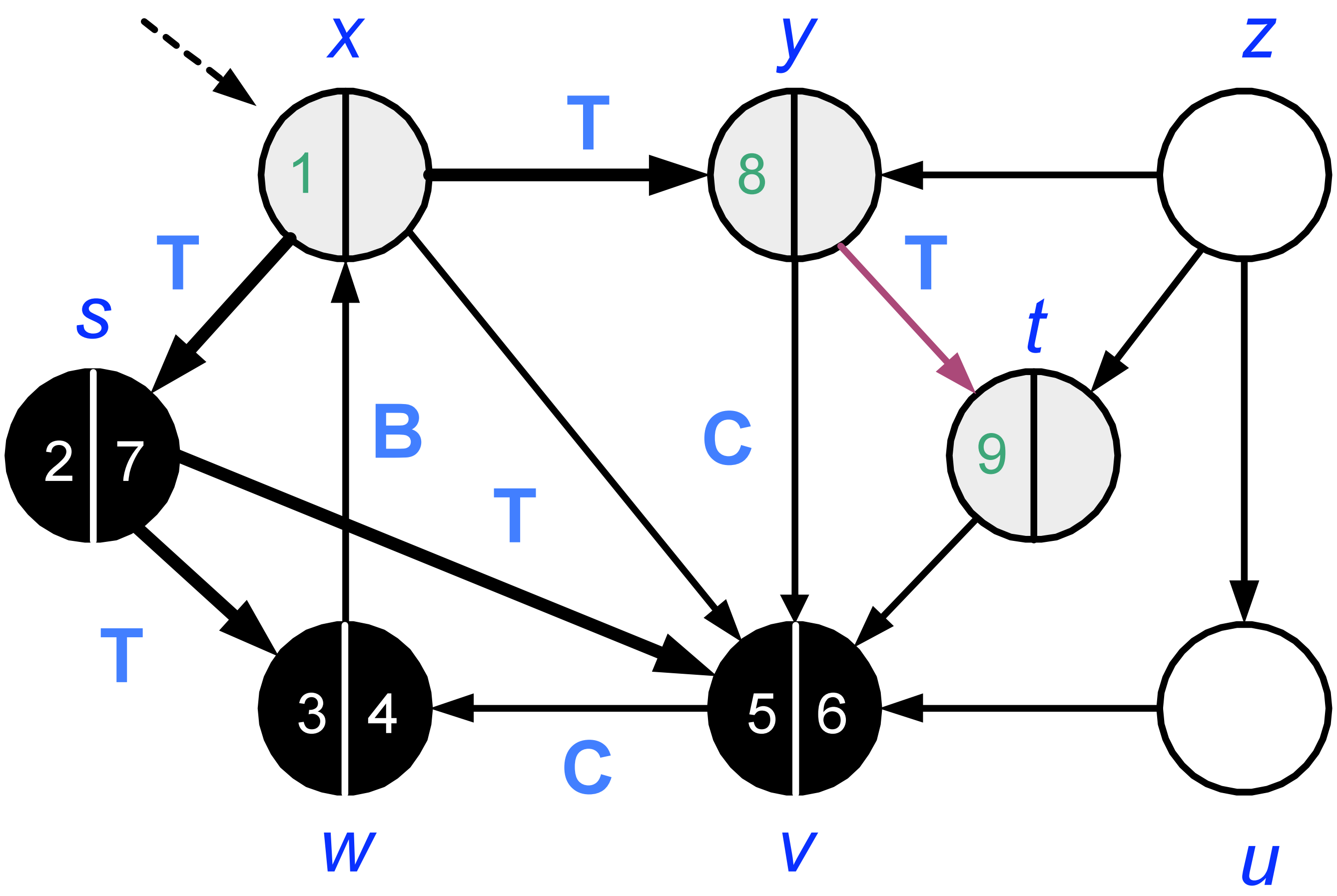
Depth-First Search: Example



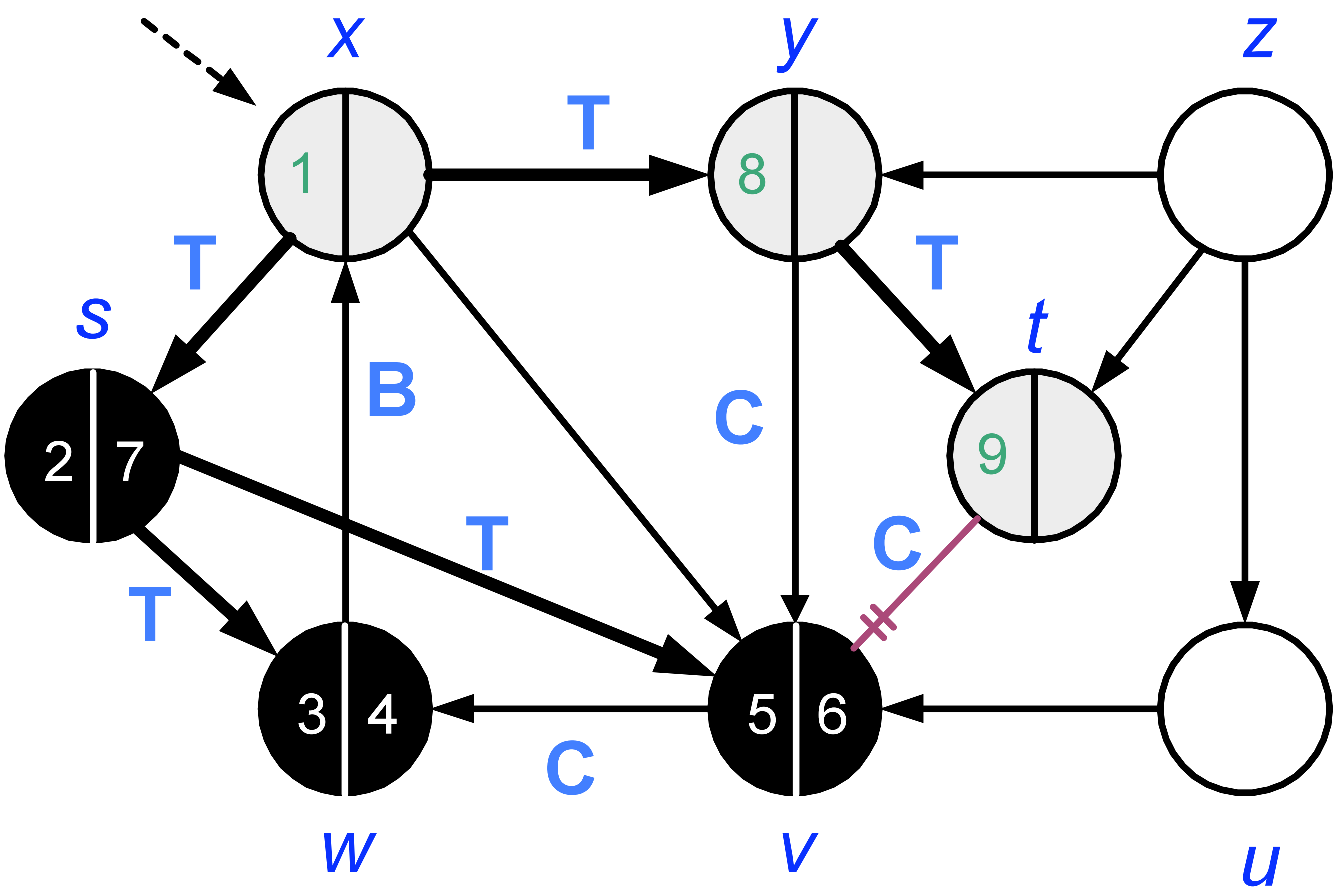
Depth-First Search: Example



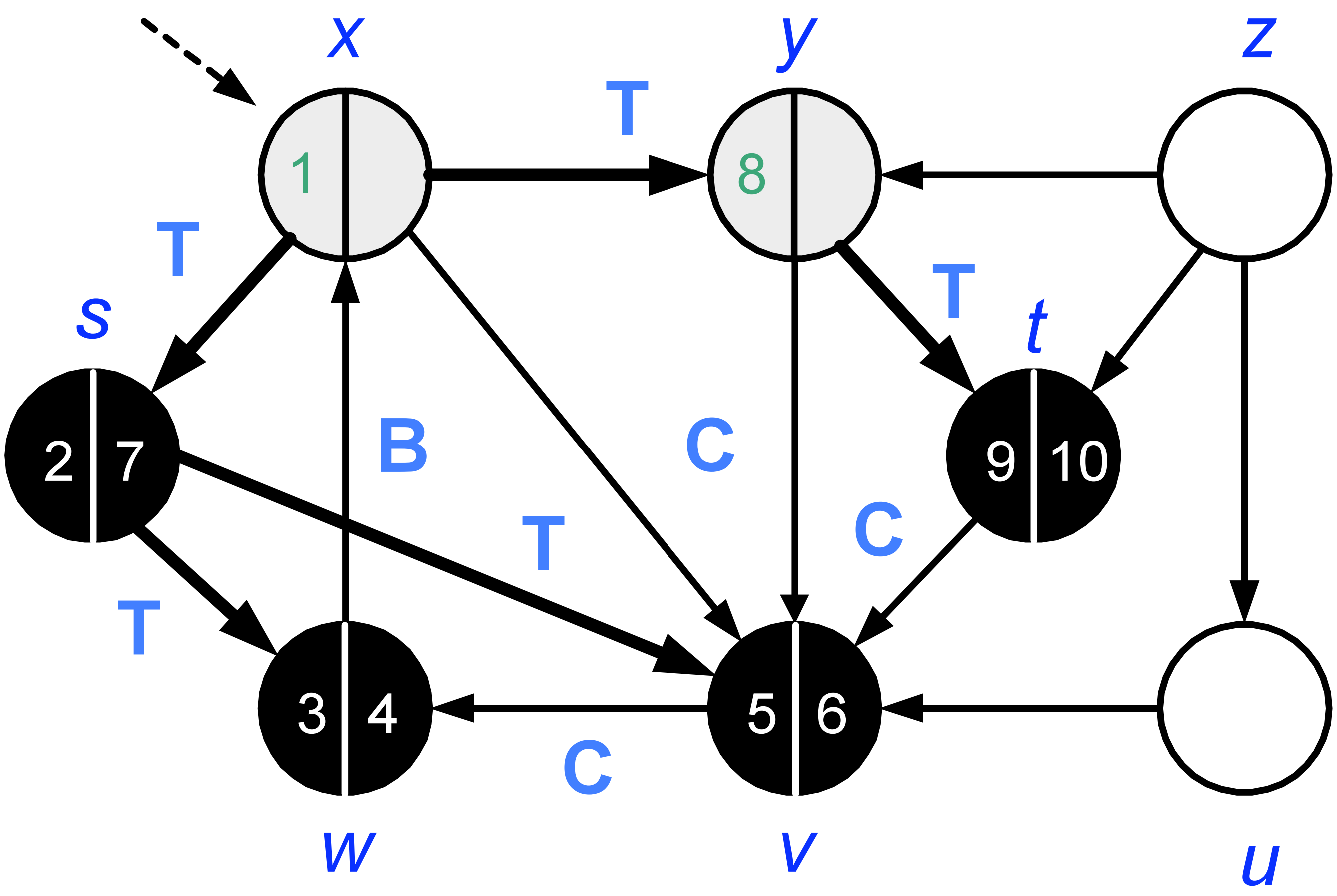
Depth-First Search: Example



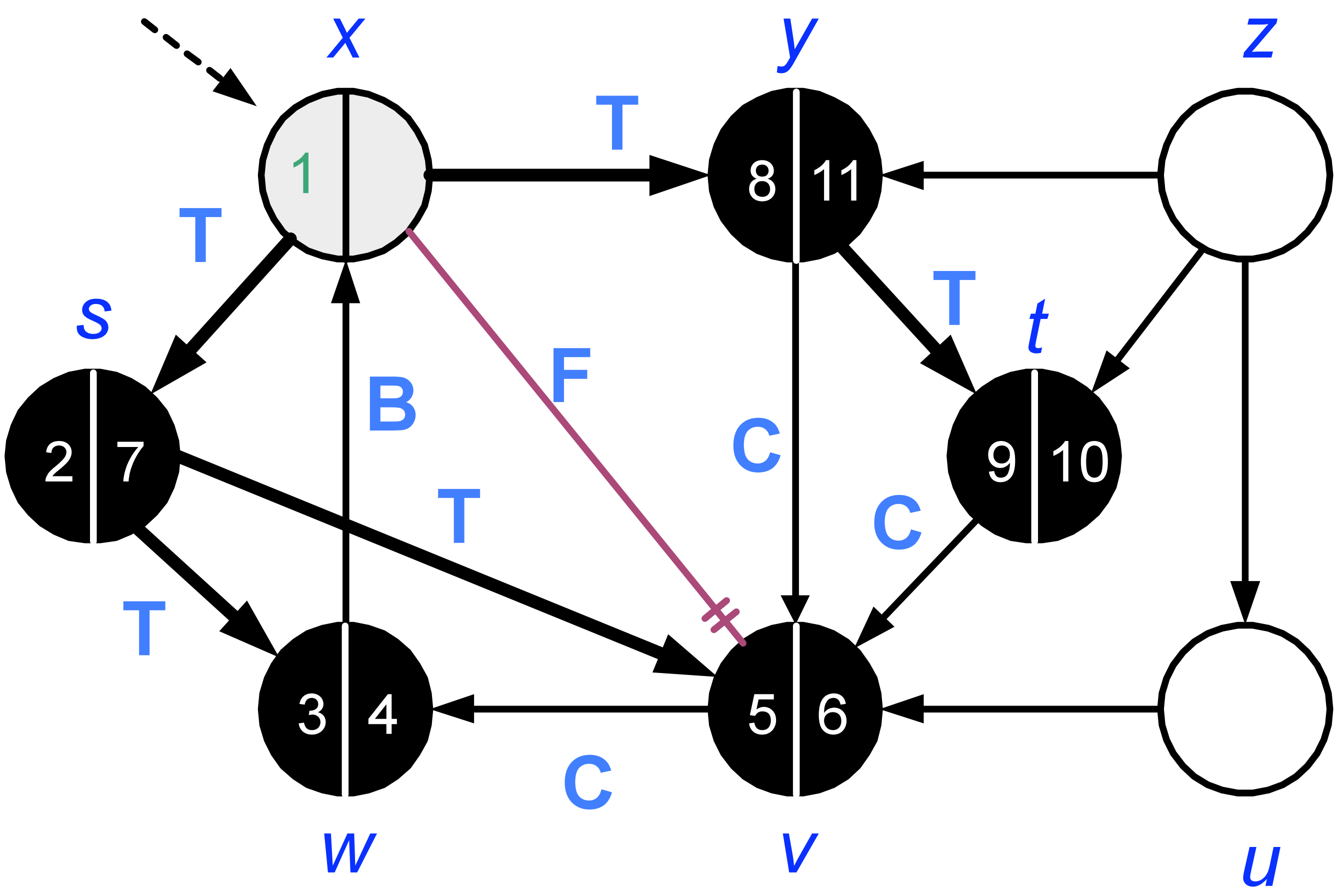
Depth-First Search: Example



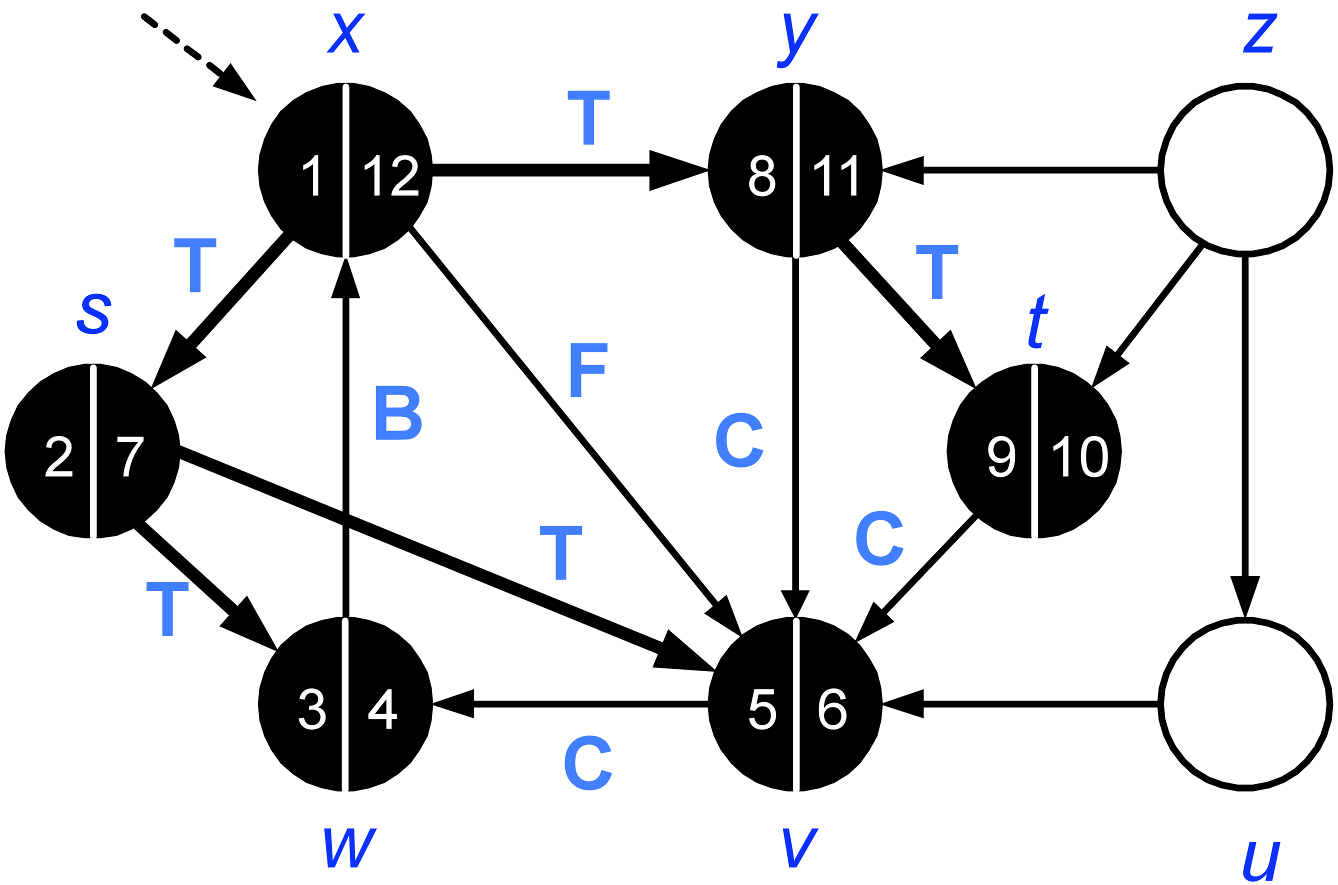
Depth-First Search: Example



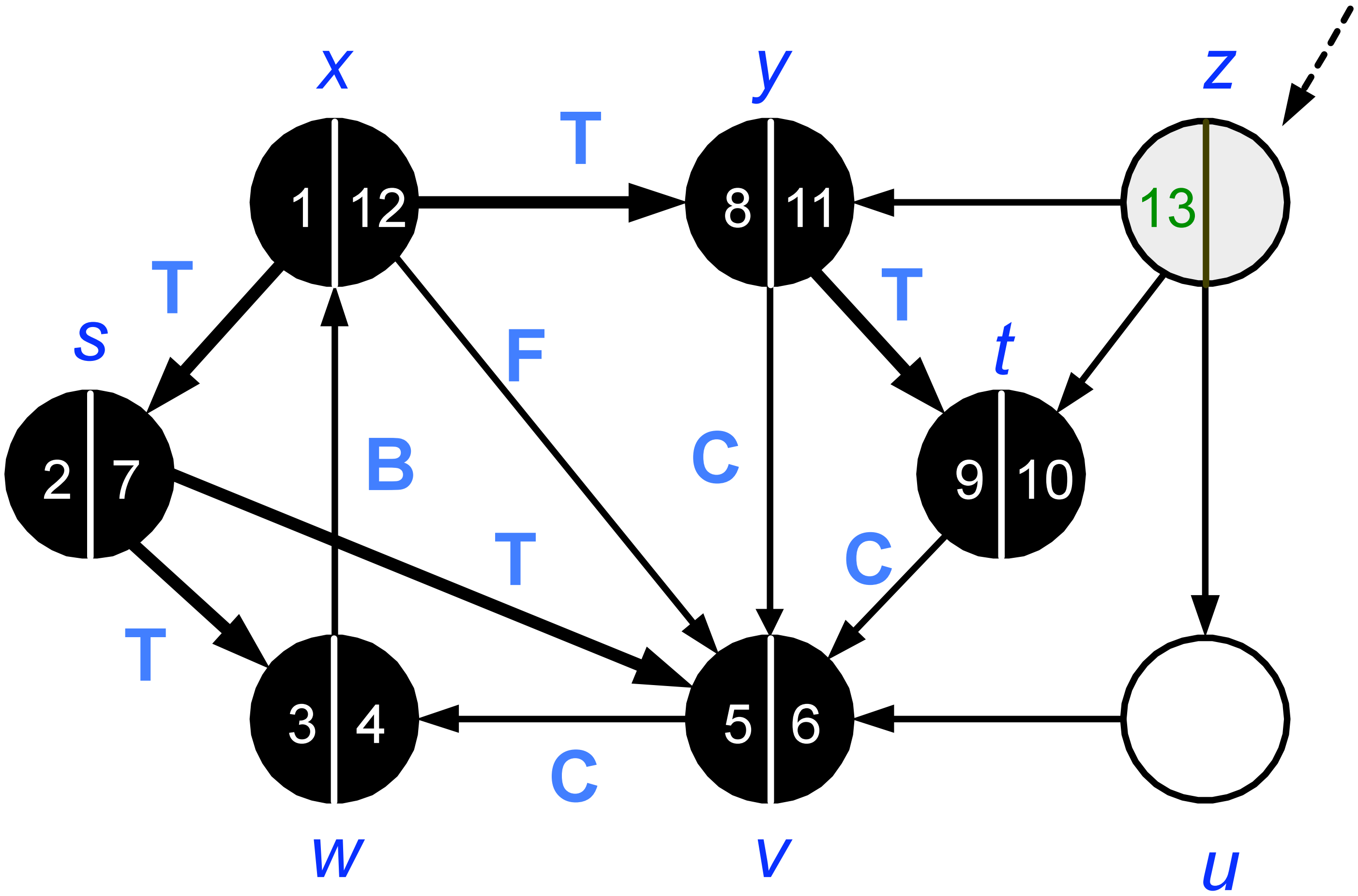
Depth-First Search: Example



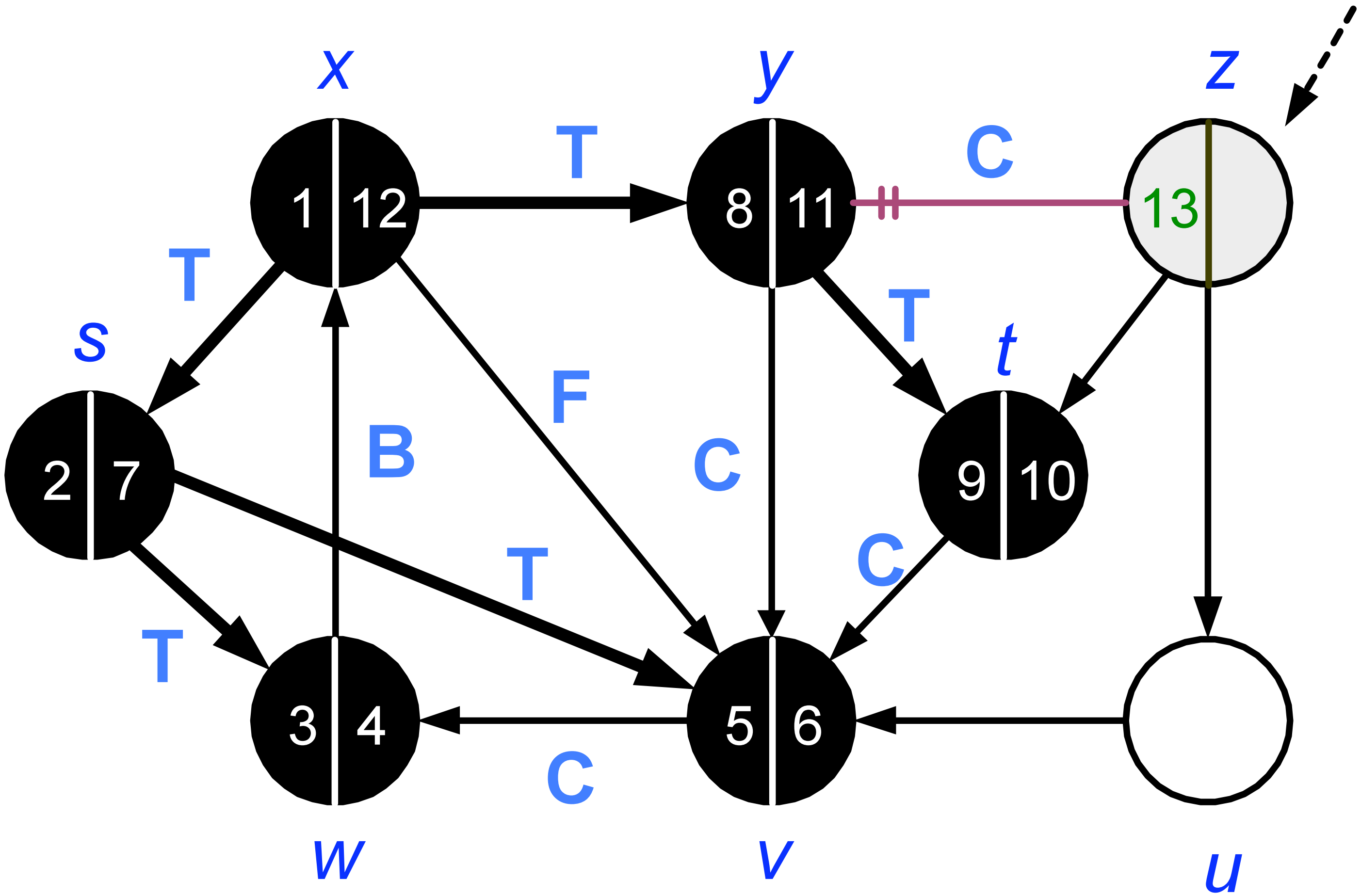
Depth-First Search: Example



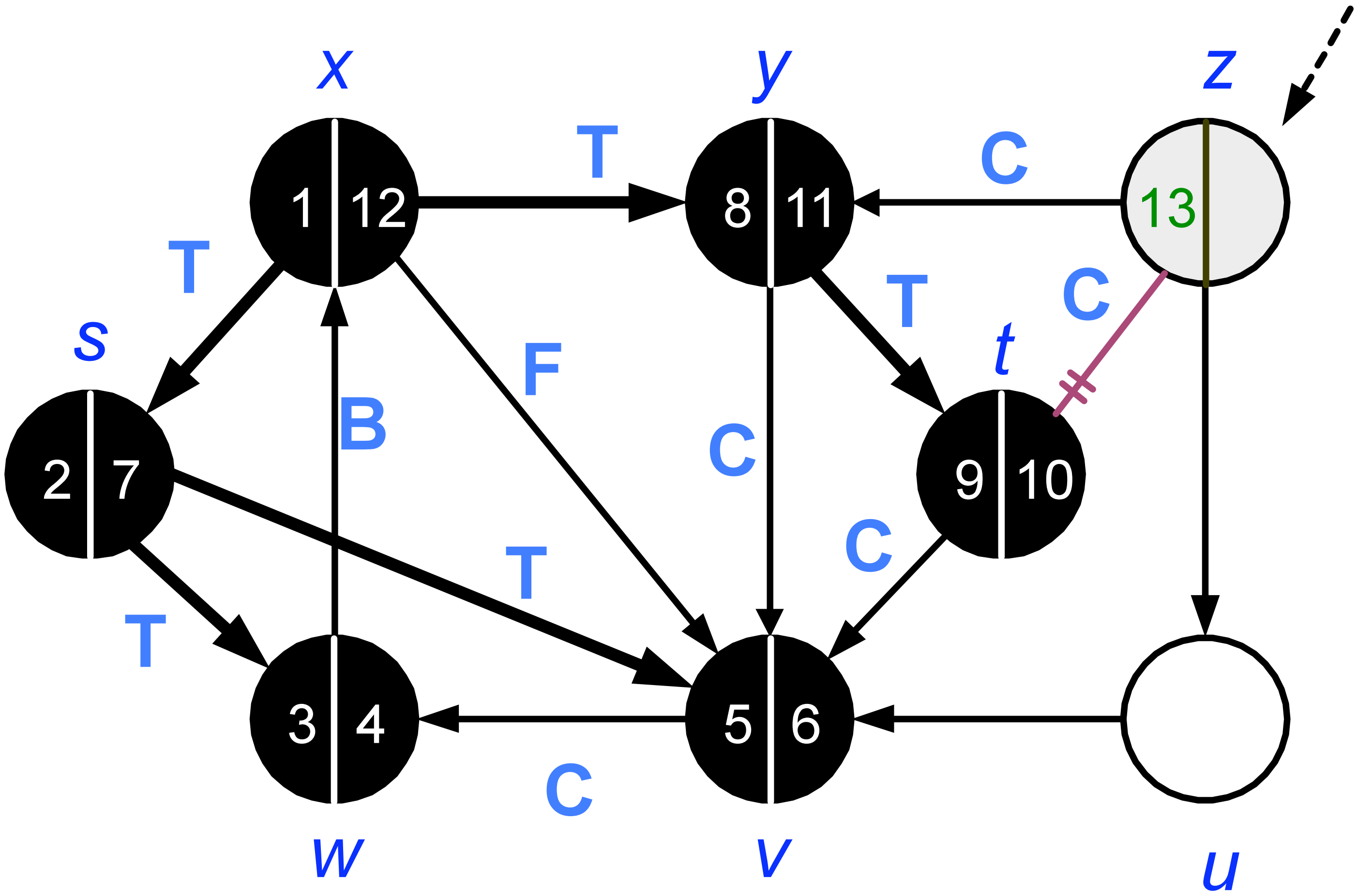
Depth-First Search: Example



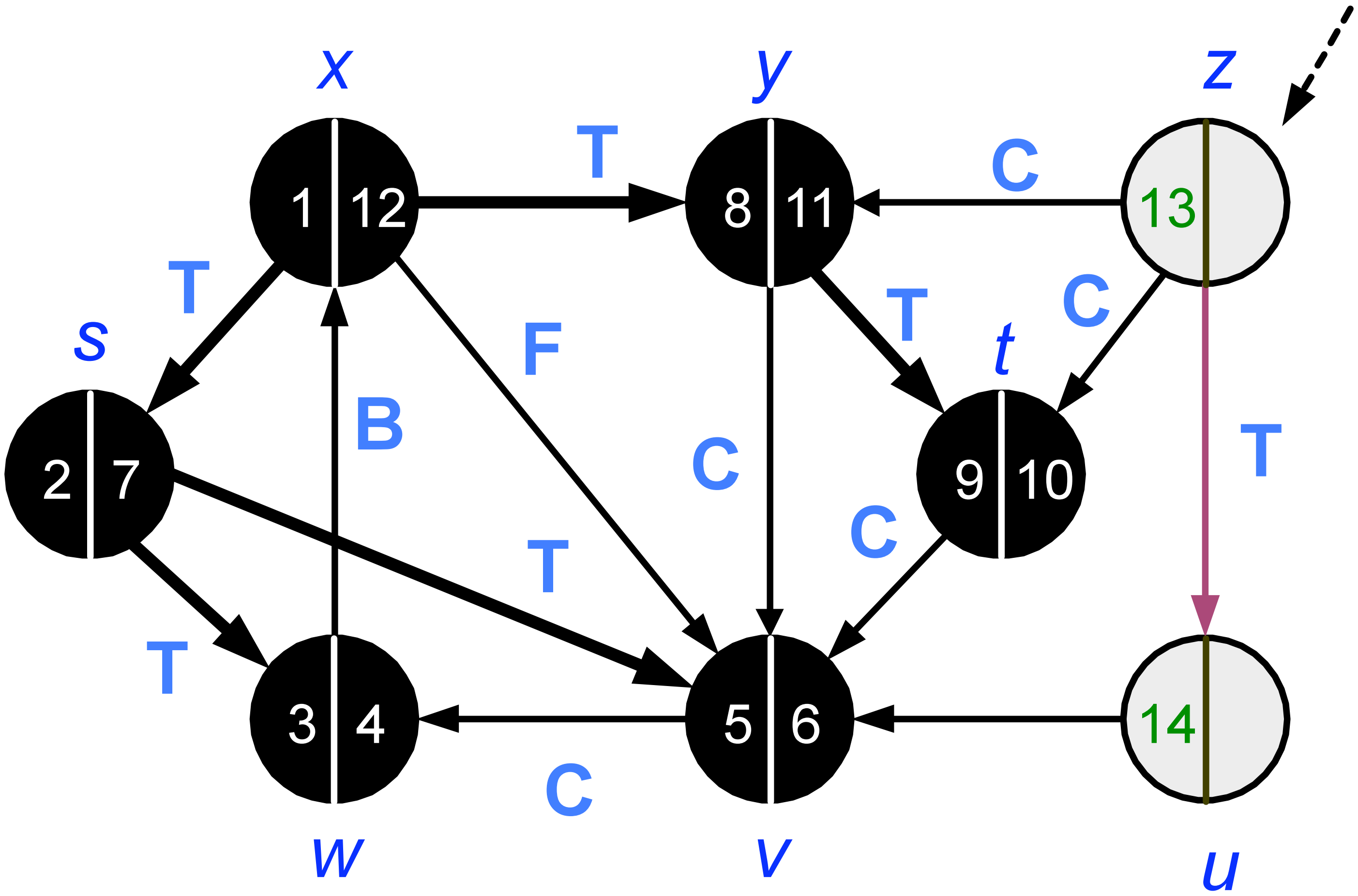
Depth-First Search: Example



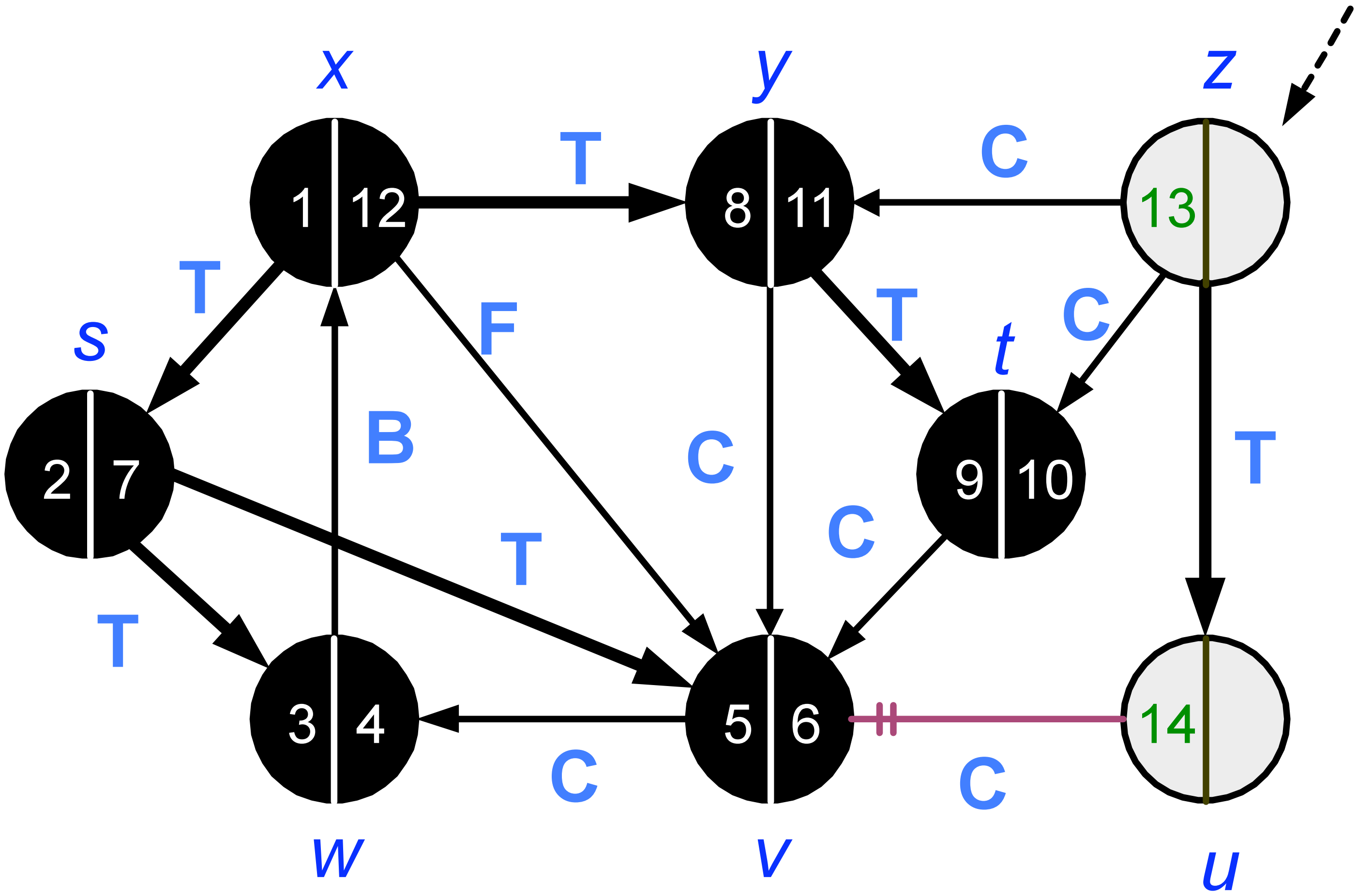
Depth-First Search: Example



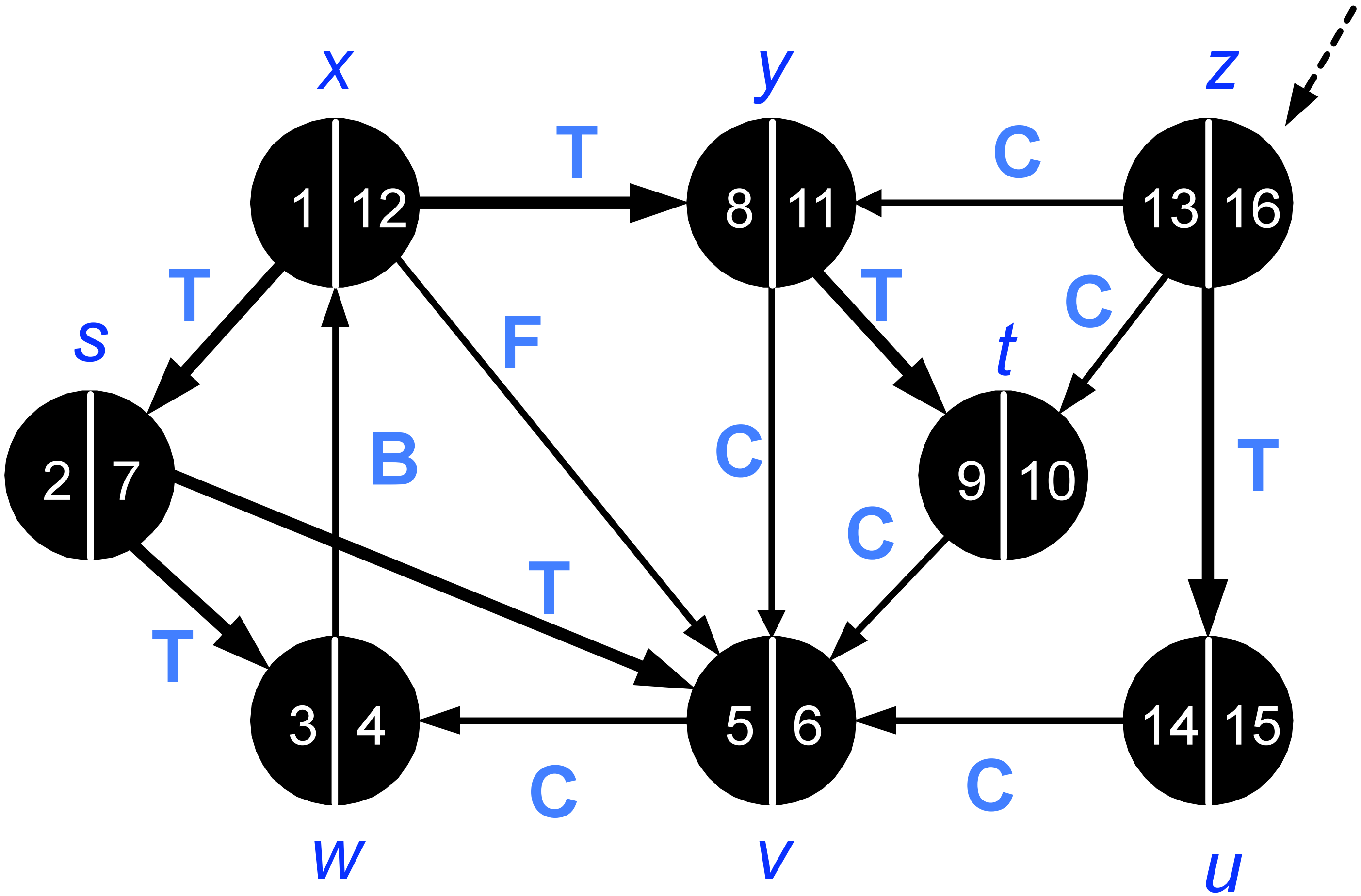
Depth-First Search: Example



Depth-First Search: Example



Depth-First Search: Example

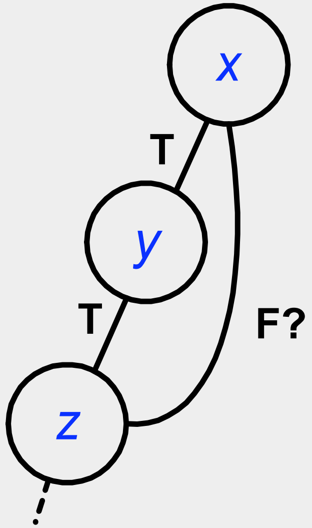


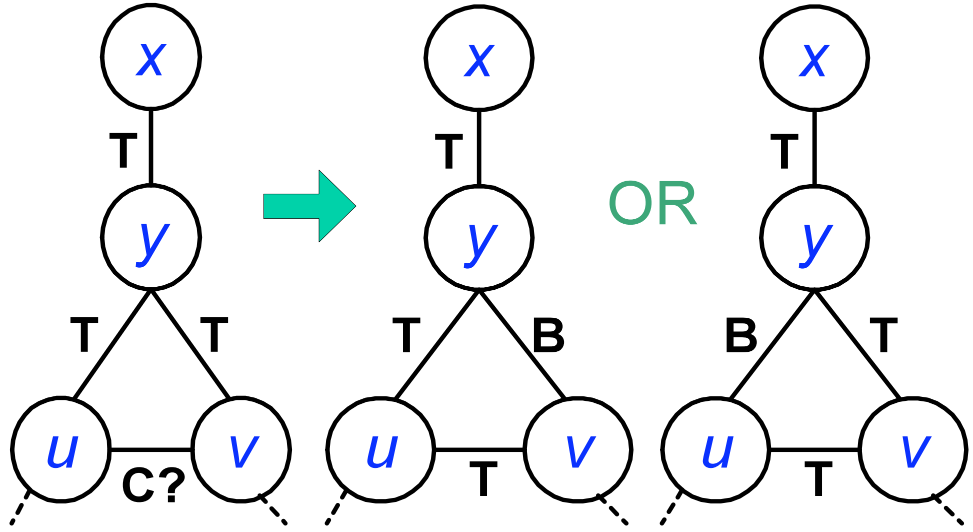
DFS on Undirected Graphs

- Ambiguity in edge classification, since (u,v) and (v,u) are the same edge
  - First classification is valid (whichever of (u,v) or (v,u) is explored first)

Lemma 1: any DFS on an undirected graph produces only Tree and Back edges

Lemma 1: Proof





Assume (x,z) is a **F** (F?)

But (x,z) must be a **B**, since DFS must finish z before resuming x

Assume (u,v) is a **C** (C?) btw subtrees

But (y,u) & (y,v) cannot be both **T**; one must be a **B** and (u,v) must be a **T**

If (u,v) is first explored while processing u/v, (y,v) / (y,u) must be a **B**

DFS on Undirected Graphs

Lemma 2: an undirected graph is acyclic (i.e. a forest) iff DFS yields no Back edges

Proof

(acyclic ⇒ no Back edges; by contradiction):

Let (u,v) be a **B** then color[u] = color[v] = GRAY

⇒ there exists a path between u and v

So, (u,v) will complete a cycle (Back edge ⇒ cycle)

(no Back edges ⇒ acyclic):

If there are no Back edges then there are only **T** edges by Lemma 1 ⇒ forest ⇒ acyclic

QED

DFS on Undirected Graphs

How to determine whether an undirected graph G=(V,E) is acyclic

- Run a DFS on G: if a Back edge is found then there is a cycle
- Running time: O(V), not O(V + E)
  - If ever seen |V| distinct edges, must have seen a back edge (|E|≤ |V| -1 in a forest)

DFS: White Path Theorem

WPT: In a DFS of G, v is a descendent of u iff at time d[u], v can be reached from u along a WHITE path

Proof (⇒): assume v is a descendent of u

Let w be any vertex on the path from u to v in the DFT

So, w is a descendent of u ⇒ d[u] < d[w]

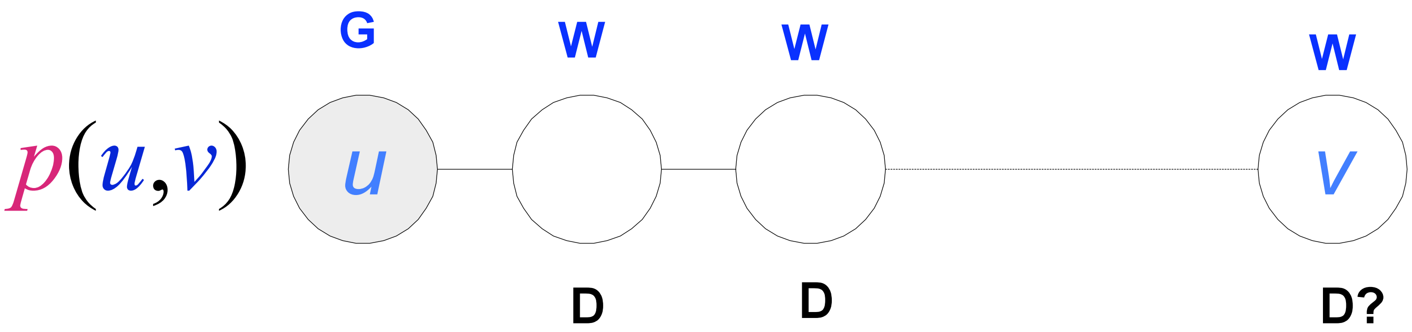
(by Corollary 1 nesting of descendents' intervals)

Hence, w is white at time d[u]

DFS: White Path Theorem

Proof (⇐) assume a white path $p(u,v)$ at time d[u] but v does not become a descendent of u in the DFT (contradiction):

Assume every other vertex along p becomes a descendent of u in the DFT



at time d[u]

DFS: White Path Theorem

otherwise let v be the closest vertex to u along p that does not become a descendent

Let w be predecessor of v along $p(u,v)$:

(1) $d[u] < d[w] < f[w] < f[u]$ by Corollary 1

(2) Since, v was WHITE at time $d[u]$ (u was GRAY) $d[u] < d[v]$

Since, w is a descendent of u but v is not

(3) $d[w] < d[v] \Rightarrow d[v] < f[w]$

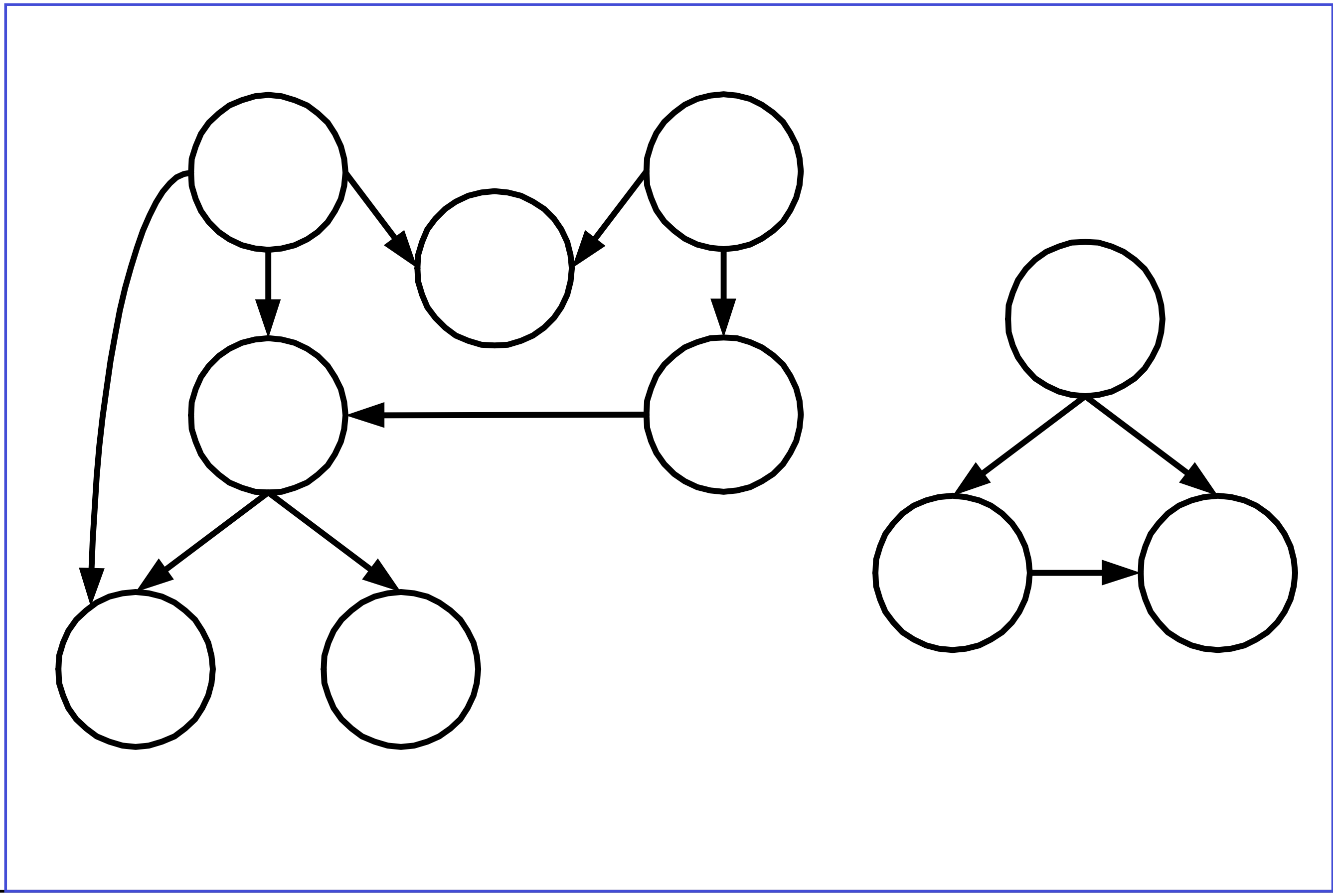
By (1)–(3): $d[u] < d[v] < f[w] < f[u] \Rightarrow d[u] < d[v] < f[w]$

So by Parenthesis Thm int[v] is within int[u], v is descendent of u QED

Directed Acyclic Graphs (DAG)

No directed cycles

Example:



Directed Acyclic Graphs (DAG)

Theorem: a directed graph G is acyclic iff DFS on G yields no Back edges

Proof (acyclic ⇒ no Back edges; by contradiction):

Let (v,u) be a Back edge visited during scanning Adj[v]

⇒ color[v] = color[u] = GRAY and d[u] < d[v]

⇒ int[v] is contained in int[u] ⇒ v is descendent of u

⇒ ∃ a path from u to v in a DFT and hence in G

∴ edge (v,u) will create a cycle (Back edge ⇒ cycle)

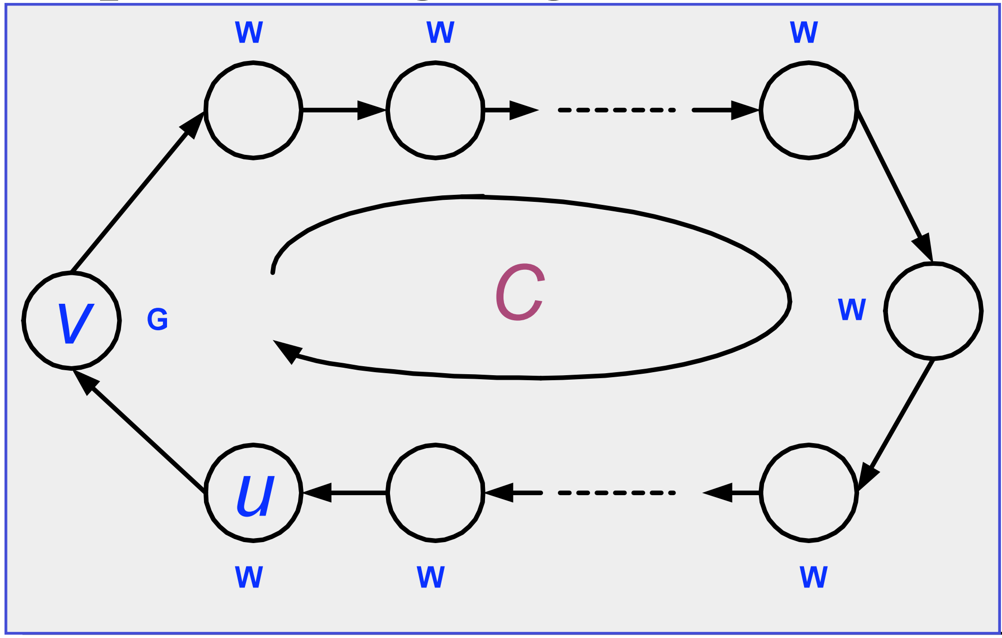
path from u to v in a DFT and hence in G

acyclic iff no Back edges

Proof (no Back edges ⇒ acyclic):

Suppose G contains a cycle *C* (Show that a DFS on G yields a Back edge; proof by contradiction)

Let *v* be the first vertex discovered in *C* and let (*u*,*v*) be proceeding edge in *C*



At time d[*v*]: ∃ a white path from *v* to *u* along *C*

By White Path Thrm *u* becomes a descendent of *v* in a DFT

Therefore (*u*,*v*) is a Back edge (descendent to ancestor)

Topological Sort of a DAG

- Linear ordering '$<$' of V such that

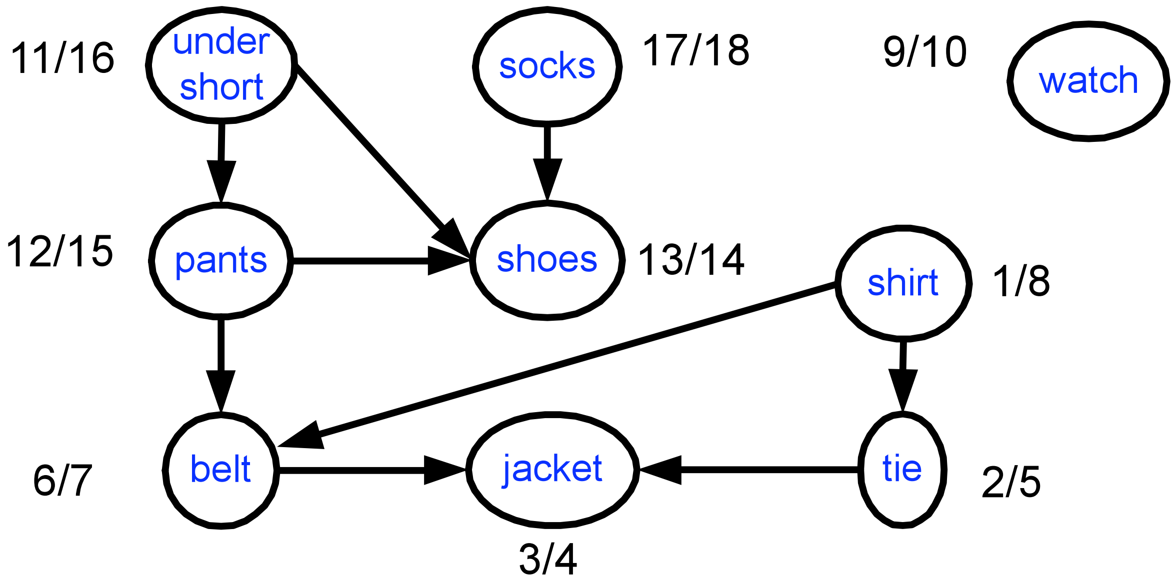
  $(u,v) \in E \Rightarrow u < v$ in ordering

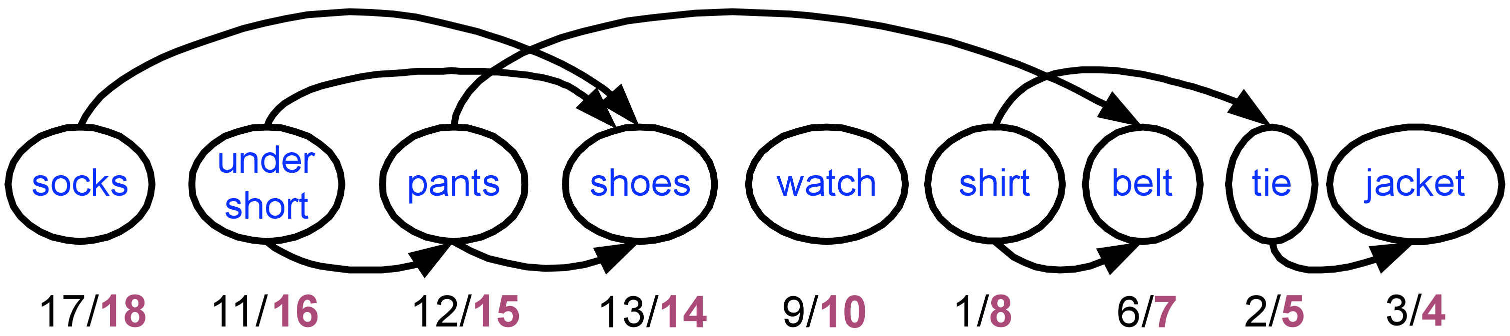
  – Ordering may not be unique

  – i.e., mapping the partial ordering to total ordering may yield more than one orderings

Topological Sort of a DAG

Example: Getting dressed





Topological Sort of a DAG

Algorithm

run DFS(G)

when a vertex finished, output it

vertices output in reverse topologically sorted order

Runs in O(V+E) time

Topological Sort of a DAG

Correctness of the Algorithm

Claim: $(u,v) \in E \Rightarrow f[u] > f[v]$

Proof: consider any edge (u,v) explored by DFS

when (u,v) is explored, u is GRAY

- if v is GRAY, (u,v) is a Back edge (contradicting acyclic theorem)
- if v is WHITE, v becomes a descendent of u (b WPT) $\Rightarrow f[v] < f[u]$
- if v is BLACK, $f[v] < d[u] \Rightarrow f[v] < f[u]$

QED