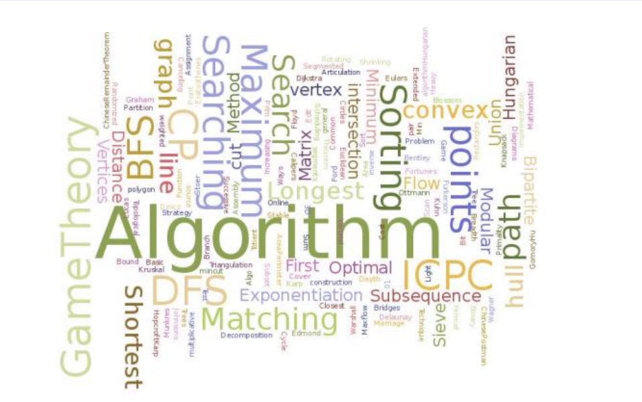
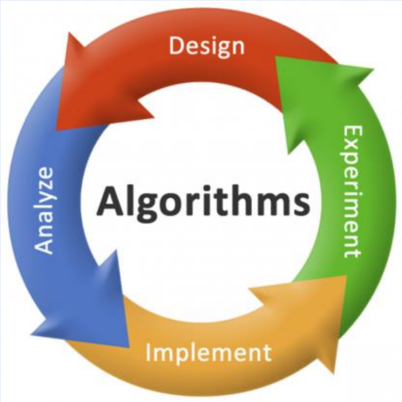
Graph Algorithms
Breadth-First-Search
Depth-First-Search

COP 3503
Fall 2021

Department of Computer Science
University of Central Florida
Dr. Steinberg





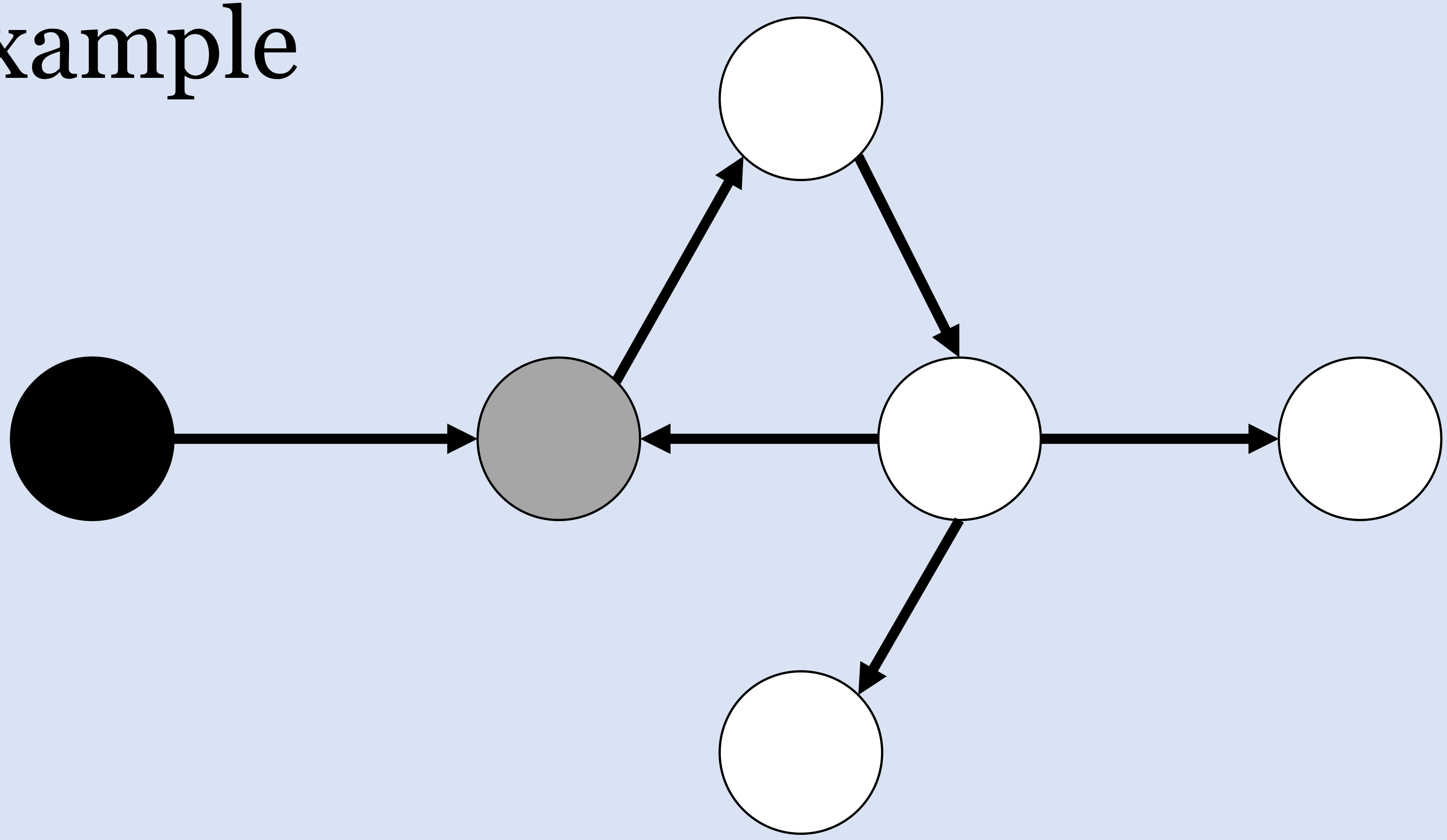
What is Breadth-First-Search (BFS)?

- One of the simplest algorithms for searching a graph?
- The input is a directed or undirected graph G(V,E) and a vertex called source (starting point).
- The overall objective is find all reachable vertices from the source (creating some path from source to another vertex)

Some Terms You should know with BFS

- For each vertex in the graph, the following attributes are maintained.
  - v.d represents the distance (number of edges) from the source vertex
  - v.π represents the predecessor (the previous vertex u to reach vertex v)
  - v.color represent the status
    - White – undiscovered
    - Gray – discovered, but not finished searching the graph
    - Black – finished with searching the graph

Example



```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

$$RT = O(V + E)$$

Example

What is Depth-First-Search (DFS)?

- Similar to BFS, however this time we are given a source to start with!
- Our input is a directed or undirected G(V,E)
- The objective is the same as BFS.

Some Terms You should know with DFS

- For each vertex in the graph, the following attributes are maintained.
  - v.d represents the discovery time
  - v.f represents the finish time
  - v.π represents the predecessor (the previous vertex u to reach vertex v)
  - v.color represent the status
    - White – undiscovered
    - Gray – discovered, but not finished searching the graph
    - Black – finished with searching the graph
  - Time – a global variable to represent the time stamp in the algorithm iteration

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)

DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

$$RT = O(V + E)$$

Parenthesis Theorem

- While running DFS
- Lets say we have vertices u and v belong to V in G(V,E). Then one of the following cases happens.
  1. u.d < v.d < v.f < u.f
     - ([])
  2. v.d < u.d < u.f < v.f
     - [()]
  3. u.d < u.f < v.d < v.f
     - ()[]
  4. v.d < v.f < u.d < u.f
     - []()
- There will never be a combination of ([)]

Edge Classification

- Tree edges (T)
  - All edges in the DF-forest
- Backedges (B)
  - (u,v) is a back edge if u is a descendant of v
- Forward edges (F)
  - (v,u) is a forward edge if u is a descendant of v
- Cross edges
  - All other edges where there is no relation between u and v
- Theorem: Let G(V,E) be an undirected graph, then DFS generates T and B edges.

Example