Depth-First Search

Depth-first search: Keep following edges from current vertex
Backtrack when no edges to unvisited vertices

Path: 0 → 4 → 6 → 5 → 3 → 7

Subpath is path 0→3, next-to-last on that subpath
pred(3) = 2
DFS-VISIT(G, u)  
for each v adjacent to u  
    if (color[v] = WHITE)  
        pred[v] = u  
        dist[v] = dist[u] + 1  
        color[v] <- GRAY  
        DFS-VISIT(G, v)  

color[u] <- BLACK

so just like for each vertex u  
for each edge (u, v)  
do something

O(n + m)

DFS(G)  
for each u in G.V  
    color[u] <- WHITE

time <- 0

for each u in G.V  
    if color[u] = WHITE  
        DFS-VISIT(G, u)
Breadth-First Search

starting from 5

FIFO queue

list 0 list 1 list 2 list 3 list 4

0 1 2 3 4 5
BFS(V,E,s)

for each vertex u in V
    color[u] <- WHITE
    d[u] <- infinity
    pred[u] <- NULL

F <- []

color[s] <- GRAY
d[s] <- 0
pred[s] <- NULL
Q <- [s]

while not Q.isEmpty() <<< iterates at most once for each vertex
    u <- Q.dequeue() Θ(1)
    for each v adjacent to u
        pred[v] <- u
        d[v] <- d[u] + 1
        color[v] <- GRAY
        Q.enqueue(v) Θ(1)
    color[u] = BLACK
    F = F + u

so just like for each vertex u for each edge (u,v) do something

O(n+m)
Shortest Paths in Weighted Graphs

Shortest Path \(0 \rightarrow 5\)

Times go with cities

For each city, remember times last message arrives