Elementary Graph Algorithms (2)

Breadth-First Search

```
BFS(G,s)
for each u ∈ V[G]-{s}
    do color[u] ← White
       d[u] ← ∞
       π[u] ← NIL
color[s] ← Gray
d[s] ← 0
π[s] ← NIL
Q ← 0
Enqueue(Q,s)
while Q ≠ 0
    do u ← Dequeue(Q)
       for each v ∈ Adj[u]
           do if color[v] = White
                 then color[v] = Gray
                      d[v] ← d[u] + 1
                      π[v] ← u
                      Enqueue(Q,v)
       color[u] ← Black
```

Depth-First Search

```
DFS(G,s)
time ← 1
    Push(S,s)
    D[s] ← time
    while S ≠ 0
        do u ← top(S)    // u still on the top of S //
            if  there is undiscovered neighbors  v of u
                then
                    time ← time+1
                    Push(S,v)
                    d[v] ← time
                else
                    time ← time+1
                    Pop(S,u)
                    f[u] ← time
```

d[*i*] is the discovery time if node *i*

f[*i*] is the finishing time of node *I*

S is a stack

Kruskal & Prim Algorithms for finding Minimum Spanning Tree

```
MST-Kruskal(G,s)
sort the edges in increasing order of its weight
for each v
    head[v] ← v
    for each edge (u,v) ∈ sorted E
        do if head[u] ≠ head[v]
            if head[v] has lower index than head[u]
                head[u] ← head[v]
                for each node x with head u
                    change head[x] to head[v]
            else
                head[v] ← head[u]
                for each node x with head v
                    change head[x] to head[u]
            flag the edge (u,v) as MST edge
```

Prim's Algorithms for finding Minimum Spanning Tree

```
MST-Prim(G,s)

for each u ∈ V[G]-{s}
      do  d[u]← ∞
            π[u] ← NIL
            Enqueue (Q,u)
d[s] ← 0
while Q ≠ 0
      do u ← top(Q)
         for each v ∈ Adj[u]
             do if v is in Q
                   and ( w(u,v) < d[v] )
                   then  d[v] ← w(u,v)
                      π[v] ← u
            Dequeue(Q,v)
```

Q: is a Minimum Priority Queue

Finding the Shortest Way

```
FSW(G,s)

for each u ∈ V[G]
      do  d[u] ← ∞
            π[u] ← NIL
            Enqueue (Q,u)
d[s] ← 0

while Q ≠ 0
      do u ← top(Q)
            for each v ∈ Adj[u] still in Q
                  if  w(u,v)+d[u] < d[v]
                        then  d[v] ← w(u,v)+d[u]
                               π[v] ← u
                  Dequeue(Q,u)
```

Q: is a Minimum Priority Queue