Wednesday, 20 October 2021 3:10 PM

Shortest path algo

→ Dijkstra
→ Bellman Ford
→ Floyd Warshall (All pairs shortest path)

Floyd Warshall

diff. long → 0
no edge → ∞

$$A^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & \infty \\ 3 & 5 & \infty & 0 & 1 \\ 4 & 2 & \infty & \infty & 0 \end{array}$$

$$A_2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & \infty & 0 & 1 \\ 4 & 2 & \infty & \infty & 0 \end{array}$$

$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 7 \\ 2 & 6 & 0 & 2 & 5 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$A^2[1,2] = \min(\infty, 3+2) = 5$

$A^2[2,3] = \min(\infty, 8+2) = 10$

$A^2[1,3] = \min(\infty, 3+\infty) = \infty$

$A^2[1,3] = A^1[1,3]$
$+ A^1[2,3]$

$$A^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 7 & 0 & 2 & 3 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$A^3[1,4] = \min(\infty, 5+1) = 6$

$A^3[2,4] = \min(15, 2+1) = 3$

for (k=1; k≤n; k++) {
for (i=1; i≤n; i++) {
for (j=1; j≤n; j++)
$A[i,j] = \min(A[i,j], A[i,k] + A[k,j])$;
}
}
}

A → path with max length

O(n³)

Cycle detection

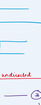
→ Detect cycle in directed graph

There is a cycle in a graph iff there is a back edge.

A back edge is an edge that is from a node to itself or a node to one of its ancestors.

→ DFS to detect a cycle in a graph problem

stack



Detect cycle in undirected graph

→ Keep track of parent node as well

visited
parent

stack

→ visited

① v[x] → true ✓
② o is not a parent ✗

① v[x] → true ✓
② o is not a parent of x ✓

g = (V,E)
main () {
for (v, → v,v) {
if (!visited[v])
if (isCyclic(v, visited, -1);
return true;
}
}

boolean isCyclic (v, visited[], parent)
visited[v] = true;
for all u in adj[v] {
if [!visited[u]) {
if (isCyclic(u, visited, v))
return true;
else if (u != parent)
return true;
}
return false
}

Topological Sorting

→ Linear ordering of vertices such that for every directed edge u,v, vertex u will come before vertex v in the ordering

→ TS is only valid for directed acyclic graph (DAG)

→ There is no unique TS ordering for a graph.

Order 1 → 1 2 4 3 5

order 2 → 1 2 4 3 5

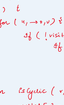
Finishing

Topological Sorting

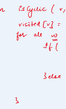
→ Kahn's Algorithm

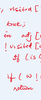
→ Finishing time

1. Perform DFS based on finishing time
2. Arrange the nodes based on decreasing order of finishing time

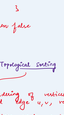


deg(v) is ended, all vertices that are reachable from v either directly or indirectly would already be visited





Kahn's Algorithm



Set out with the nodes having indegree 0.

topological sort (N, adj[N][N])
Q
T= []
visited = []
indegree = []
for (i=0; i<n; i++)
in_degree[i] = visited[i] = 0
for i=0 to N
for j=0 to N
if adj[i][j] → 1
indegree[j] += 1;
for i=0 to N
if in_degree[i] → 0
Q.enqueue(i);
visited[i] = 1;

while Q is not empty
vertex = q.front();
q.dequeue();
T.append(vertex);
for j=0 to N
if (adj[vertex][j] → 1 & visited[j] → 0)
in_degree[j] -= 1;
if (in_degree[j] → 0)
q.enqueue(j);
visited[j] → 1;

Course Schedule