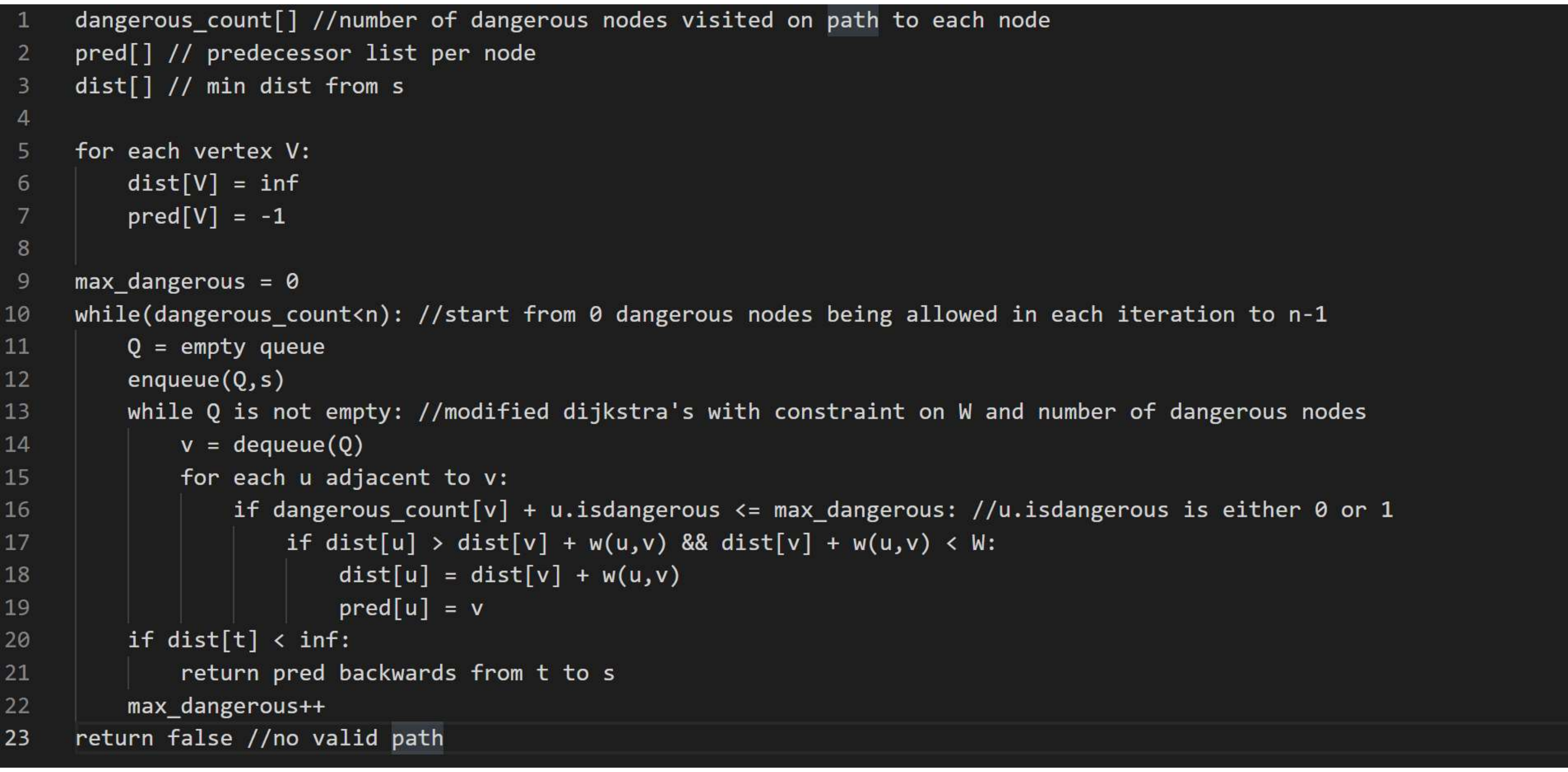

```
dangerous_count[] //number of dangerous nodes visited on path to each node
pred[] // predecessor list per node
dist[] // min dist from s

for each vertex V:
    dist[V] = inf
    pred[V] = -1

max_dangerous = 0
while(dangerous_count<n): //start from 0 dangerous nodes being allowed in each iteration to n-1
    Q = empty queue
    enqueue(Q,s)
    while Q is not empty: //modified dijkstra's with constraint on W and number of dangerous nodes
        v = dequeue(Q)
        for each u adjacent to v:
            if dangerous_count[v] + u.isdangerous <= max_dangerous: //u.isdangerous is either 0 or 1
                if dist[u] > dist[v] + w(u,v) && dist[v] + w(u,v) < W:
                    dist[u] = dist[v] + w(u,v)
                    pred[u] = v
    if dist[t] < inf:
        return pred backwards from t to s
    max_dangerous++
return false //no valid path
```