Graph theory algorithms

Bruce Merry

February 6, 2004

Algorithm 1 Depth first search (recursive)

```
if not seen[x] then
    seen[x] ← true
    process(x)
    for each neighbour y of x do
        dfs(y)
    end for
end if
```

Algorithm 2 Depth first search (stack based)

```
clear stack
push start
while stack not empty do
    pop stack into cur
    if not seen[cur] then
        process(cur)
        seen[cur] ← true
        for each neighbour next of cur do
            push next
        end for
    end if
end while
```

Algorithm 3 Unweighted shortest path (BFS)

clear queue
push $start$
for each node i **do**
  dist[i] $\leftarrow \infty$
  parent[i] $\leftarrow -1$
end for
dist[$start$] $\leftarrow 0$
while queue not empty **do**
  pop queue into cur
  for each neighbour $next$ of cur **do**
    if dist[$next$] $\neq \infty$ **then**
      dist[$next$] $\leftarrow$ dist[cur] $+ 1$
      parent[$next$] $\leftarrow cur$
      push $next$
    end if
  end for
end while

The path is found by starting at the end node, and following the `parent` links backwards.

Algorithm 4 Dijkstra's algorithm (shortest path)

clear priority queue
push $start$ with priority 0
for each node i **do**
  dist[i] $\leftarrow \infty$
  parent[i] $\leftarrow -1$
end for
dist[$start$] $\leftarrow 0$
while priority queue not empty **do**
  pop priority queue into cur with priority $prio$
  if $prio =$ dist[cur] **then**
    for each neighbour $next$ of cur with length len **do**
      if $prio + len <$ dist[$next$] **then**
        dist[$next$] $\leftarrow prio + len$
        parent[$next$] $\leftarrow cur$
        push $next$ with priority dist[$next$]
      end if
    end for
  end if
end while

Algorithm 5 Floyd's algorithm (all shortest paths)

```
for y = 1 to N do
  for x = 1 to N do
    if matrix[x][y] ≠ ∞ then
      for z = 1 to N do
        if matrix[x][y] + matrix[y][z] < matrix[x][z] then
          matrix[x][z] ← matrix[x][y] + matrix[y][z]
        end if
      end for
    end if
  end for
end for
```

The algorithm works in place, converting an adjacency matrix to a minimum distance matrix.