Prim’s Algorithm

procedure prim(G,W,s)
for each vertex v ∈ V[G] do
    d[v] ← ∞
    π[v] ← NIL
end for
outside ← V[G]
d[s] ← 0
while outside ≠ ∅ do
    u ← Extract_Min(outside with respect to distance d)
    for each v adjacent to u do
        if v ∈ outside and W[u,v] < d[v] then
            d[v] ← W[u,v]
            π[v] ← u
        end if
    end for
end while
end procedure
Prim's Algorithm, Dense Graphs

procedure prim(G,W)

for i = 1 to n do
    d[i] ← ∞
    outside[i] ← true
    π[i] ← NIL
end for

d[0] ← ∞

d[1] ← 0
for i = 1 to n do
    k ← 0
    for j = 1 to n do
        if outside[j] and d[j] ≤ d[k] then k ← j
        outside[k] := false
    end for

    for j = 1 to n do
        if outside[j] and W[j,k] < d[j] then
            d[j] ← W[j,k]
            π[j] ← k
        end if
    end for

end for

end procedure
Prim’s Algorithm, Sparse Graphs

{The priority queue for the distances of each vertex from the tree is stored as a min heap. The actual item in the heap is the name of the vertex. Its value (for heap operations) is in the array d[1,...,n]}

procedure prim(G,W)

for i = 1 to n do
    MinHeap[i] ← i
    WhereInHeap[i] ← i
    d[i] ← ∞
    outside[i] ← true
    π[i] ← NIL
end for

d[1] ← 0
for i = n downto 1 do
    u ← MinHeap[1]
    MinHeap[1] ← MinHeap[i]
    WhereInHeap[MinHeap[1]] ← 1
    SiftDown(1,i-1)  {Keeping track of WhereInHeap}
    for each v ∈ adj[u] do
        if v ∈ outside and W[u,v] < d[v] then
            d[v] ← W[u,v]
            π[v] ← u
            SiftUp(WhereInHeap[v])  {Keeping track of WhereInHeap}
        end if
    end for
end for
end procedure