Practices of Discrete Mathematics

Session 6 (Searching algorithms)

1. Searching strategies
2. BFS algorithm
3. DFS algorithm

Searching strategy

A searching strategyin a connected graph is a systematic procedure for visiting all the vertices of G “travelling” along its edges.

That is, a searching strategy is a systematic procedure to construct a **generating subgraph** of a connected graph G. Dicho de otro modo, un algoritmo de búsqueda es un proceso sistemático para construir un subgrafo de G que contiene a todos sus vértices, es decir, un subgrafo generador de G.

We are going to describe two algorithms to obtain a **generating tree** of G:

- Breadth-first search (or BFS).
- Depth-first search (or DFS).

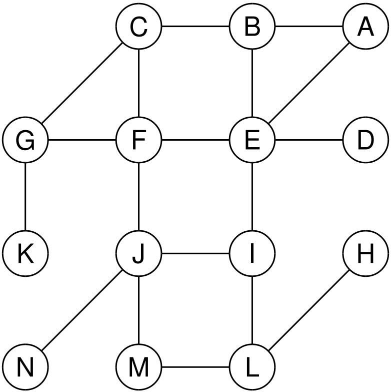
BFS algorithm

Let G be a connected graph

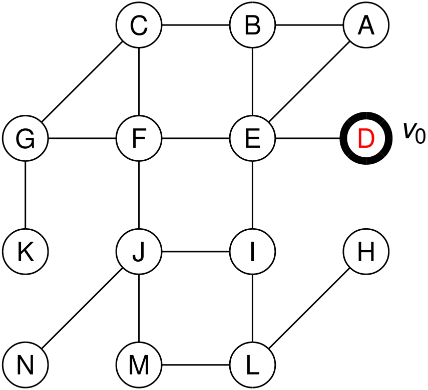
1. Choose a vertex v_0.
2. Let $m = 0$, $w := v_m = v_0$ and $n = 0$. (The vertex $w = v_m$ is called the current "center"). Let T_0 the tree without edges whose unique vertex is v_0.
3. If there exists some new vertex (new means "non-visited") that is adjacent to w then
   - choose one of them, v_{n+1};
   - Add it to T_n and add also an edge e_n that joins w and v_{n+1}. Doing this we will obtain the next tree T_{n+1};
   - replace n by $n + 1$;
   - repeat Step 3 until there are no further new vertices adjacent to the current center w.
4. If all vertices have now been visited then T_n is a generating tree and we stop. Otherwise replace m by $m + 1$, take $w = v_m$ and go to Step 3.

The obtained tree T_n is a generating tree of G.

Example

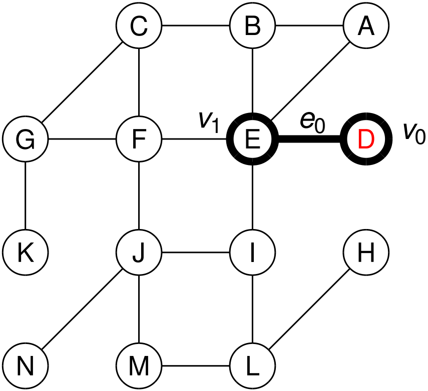


Example (Steps 1 and 2)



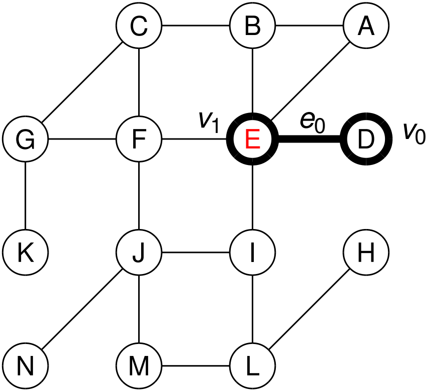
$m = 0$ Center: $w = v_0 = D$ $n = 0$ T_0 : (tree)

Example (Step 3)



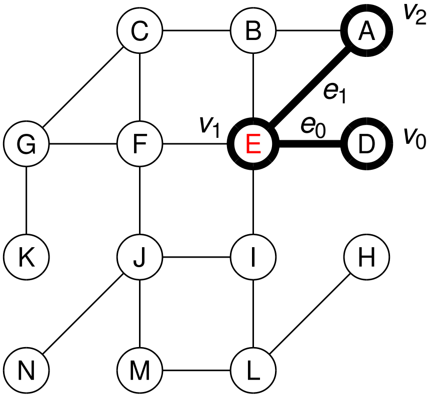
$m = 0$ Center: $w = v_0 = D$ $n = 1$ T_1 : (tree)

Example (Step 4)



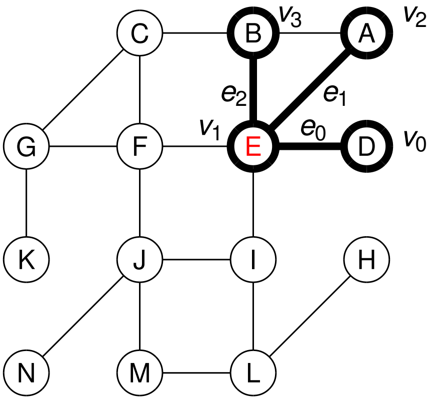
$m = 1$ Center: $w = v_1 = E$ $n = 1$ T_1 : (tree)

Example (Step 3)



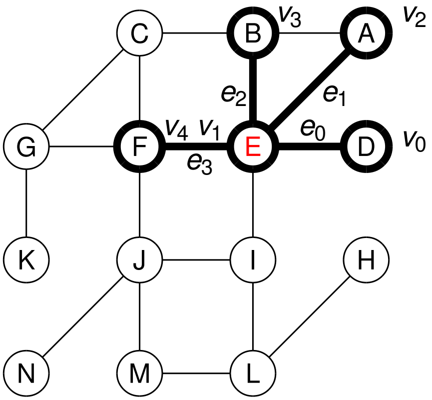
$m = 1$ Center: $w = v_1 = E$ $n = 2$ T_2 : (tree)

Example (Step 3)



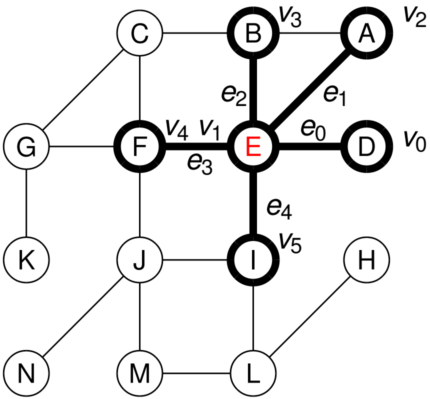
$m = 1$ Center: $w = v_1 = E$ $n = 3$ T_3 : (tree)

Example (Step 3)



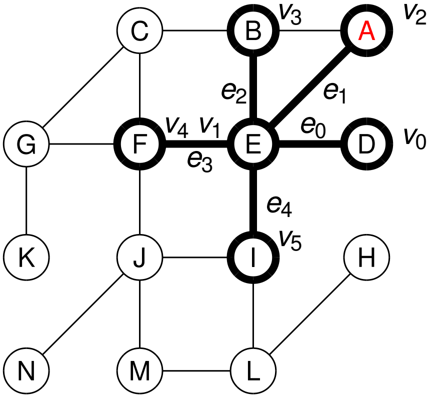
$m = 1$ Center: $w = v_1 = E$ $n = 4$ T_4 : (tree)

Example (Step 3)



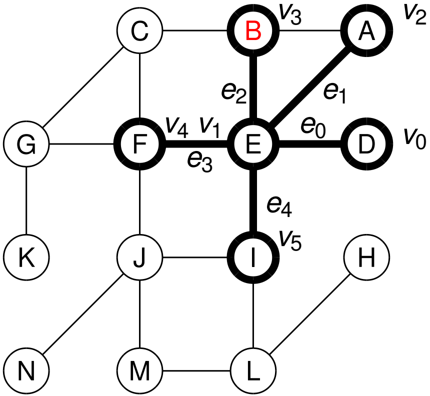
$m = 1$ Center: $w = v_1 = E$ $n = 5$ T_5 : (tree)

Example (Step 4)



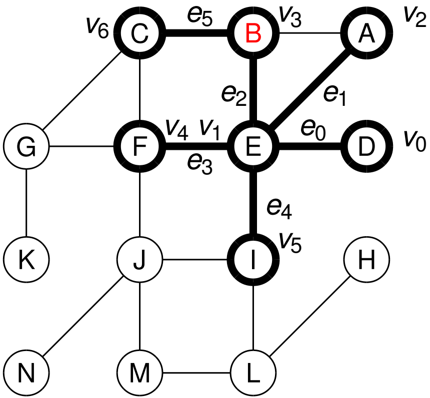
$m = 2$ Center: $w = v_2 = A$ $n = 5$ T_5 : (tree)

Example (Step 4)



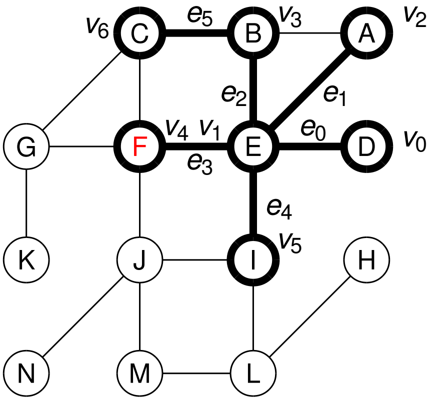
$m = 3$ Center: $w = v_3 = B$ $n = 5$ T_5 : (tree)

Example (Step 3)



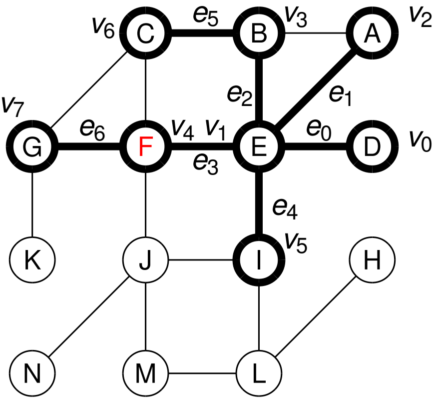
$m = 3$ Center: $w = v_3 = B$ $n = 6$ T_6 : (tree)

Example (Step 4)



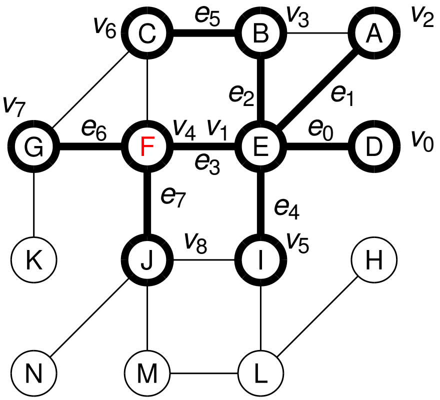
$m = 4$ Center: $w = v_4 = F$ $n = 6$ T_6 : (tree)

Example (Step 3)



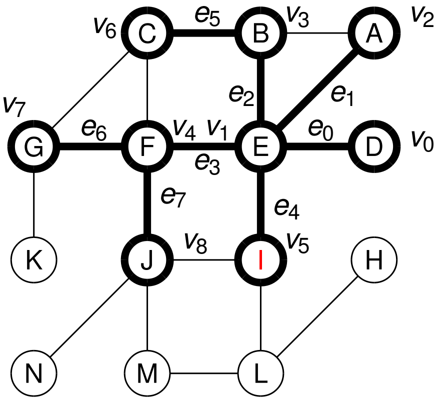
$m = 4$ Center: $w = v_4 = F$ $n = 7$ T_7 : (tree)

Example (Step 3)



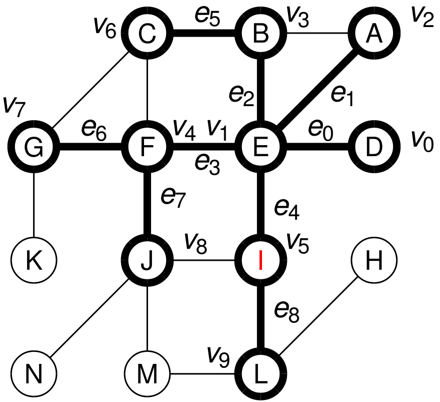
$m = 4$ Center: $w = v_4 = F$ $n = 8$ T_8 : (tree)

Example (Step 4)



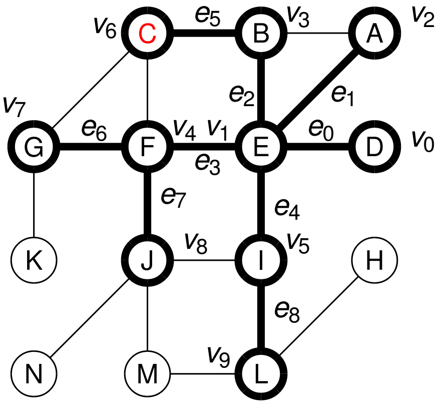
$m = 5$ Center: $w = v_5 = I$ $n = 8$ T_8 : (tree)

Example (Step 3)



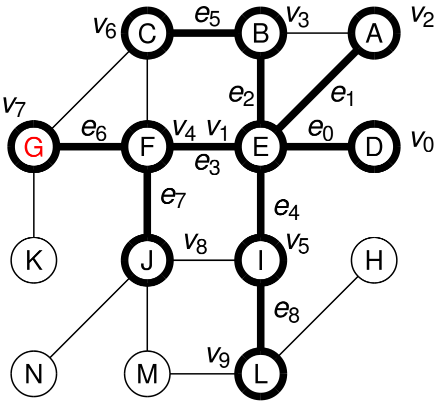
$m = 5$ Center: $w = v_5 = I$ $n = 9$ T_9 : (tree)

Example (Step 4)



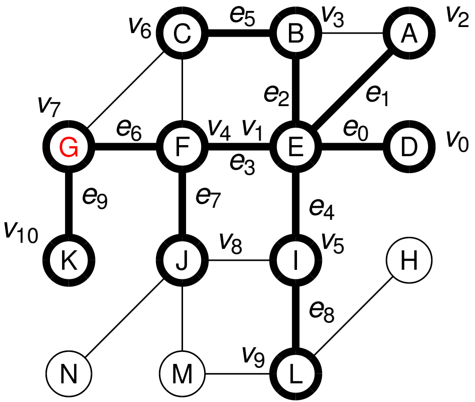
$m = 6$ Center: $w = v_6 = C$ $n = 9$ T_9 : (tree)

Example (Step 4)



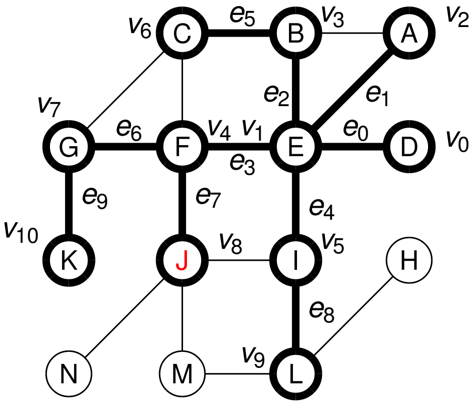
$m = 7$ Center: $w = v_7 = G$ $n = 9$ T_9 : (tree)

Example (Step 3)



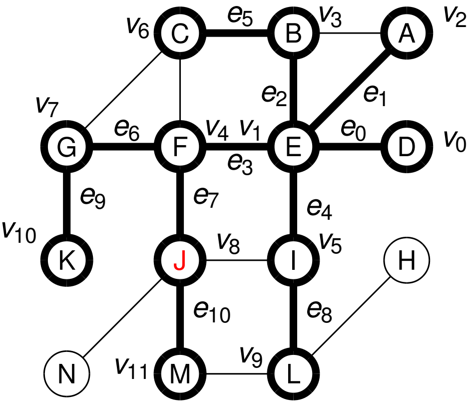
$m = 7$ Center: $w = v_7 = G$ $n = 10$ T_{10} : (tree)

Example (Step 4)



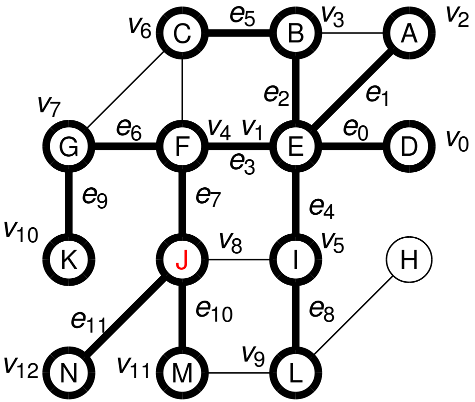
$m = 8$ Center: $w = v_8 = J$ $n = 10$ T_{10} : (tree)

Example (Step 3)



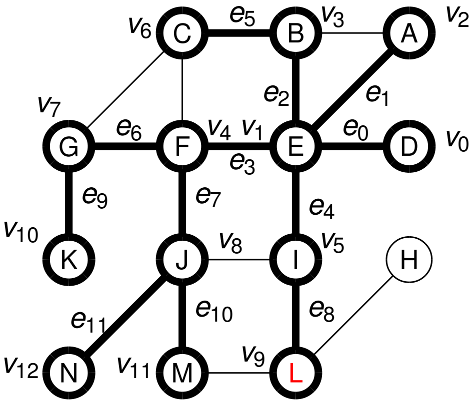
$m = 8$ Center: $w = v_8 = J$ $n = 11$ T_{11} : (tree)

Example (Step 3)



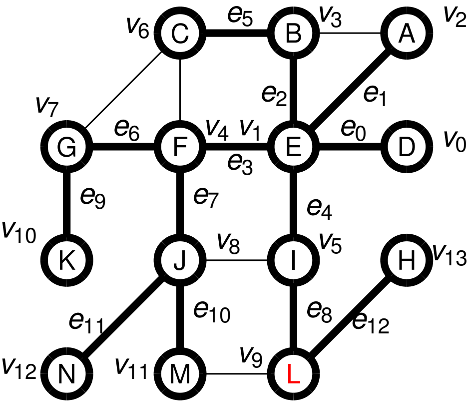
$m = 8$ Center: $w = v_8 = J$ $n = 12$ T_{12} : (tree)

Example (Step 4)



$m = 9$ Center: $w = v_9 = L$ $n = 12$ T_{12} : (tree)

Example (Step 3)



$m = 9$ Center: $w = v_9 = L$ $n = 13$ T_{13} : (tree)

1. Searching strategies
2. BFS algorithm
3. DFS algorithm

DFS algorithm

1. Choose a vertex v_0 and define $n = 0$, $m = 0$, $w_m = v_0$ and $w = w_m$ (w is the current "center" of the search). Let T_0 be the tree without edges whose unique vertex is v_0.
2. Is there exists some new vertex that is adjacent to w then
   - choose one of them, v_{n+1};
   - add it to T_n and an edge joining w and v_{n+1}. Doing this we will get the next tree T_{n+1};
   - take $w_{m+1} = v_{n+1}$, $w = w_{m+1}$ and increase m and n in one unity;
   - repeat Step 2 until the current center w is not adjacent to any new vertex.
3. If all vertices have been visited then we have a generating tree and we stop. Otherwise backtrack along the last edge to the previous center (that is, let $w = w_{m-1}$), replace m by $m - 1$ and go to Step 2.

Observation

The algorithms BFS and DFS can also be applied on non-connected graphs:

If we start the algorithm with an initial vertex v, the result is a tree whose vertices are all the vertices of the graph that are connected with v, that is, they are the vertices of the connected component of v