॥ सा विद्या या विमुक्तये ॥

भारतीय प्रौद्योगिकी संस्थान धारवाड़

Indian Institute of Technology Dharwad

Maximal Neighborhood Search
v/s
Maximum Cardinality Search

Generic Search

- It is one of the most fundamental and general search method.
- In its most general form, a generic search is a method of traversing vertices of a given graph such that every prefix of the obtained vertex ordering induces a connected graph.
- This general definition leaves much freedom for a selection rule determining which node is to be chosen next for exploration.
- By defining some specific rule that restricts this choice, various different graph search methods are defined.

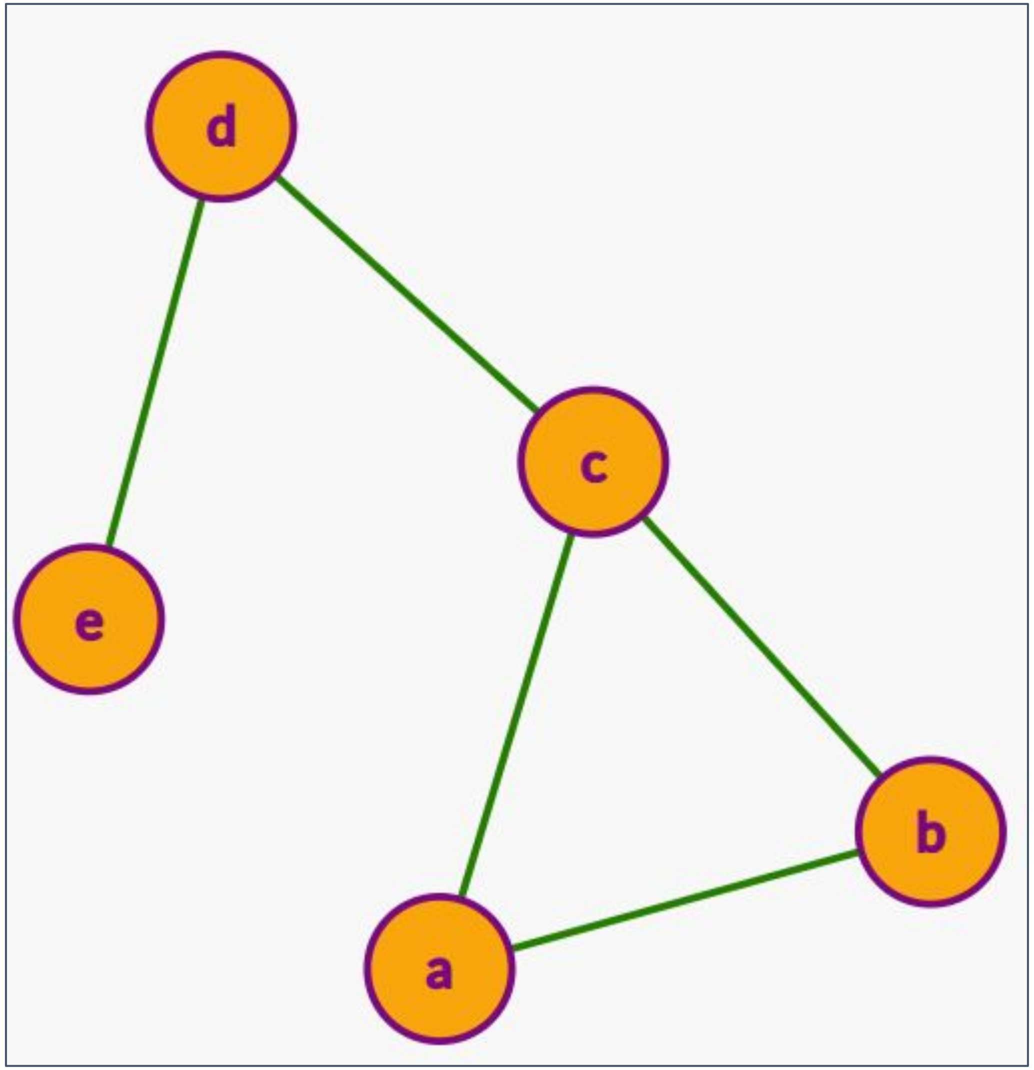
Algorithm : Generic Search

input : a graph $G = (V, E)$ and start vertex $s \in V$
output: an ordering σ of V

$S \leftarrow \{s\}$
for $i \leftarrow 1$ *to* n **do**
    pick and remove an unnumbered vertex v from S
    $\sigma(i) \leftarrow v$ // This assigns to v the number i
    foreach *unnumbered vertex* w *adjacent to* v **do**
        add w to S

Example for illustrating Generic Search, BFS, DFS

- ➔ Vertex ordering for Generic Search : {a, c, d, b, e}.
- ➔ Vertex ordering for BFS : {a, c, b, d, e}
- ➔ Vertex ordering for DFS : {a, c, d, e, b}



Graph G with 5 vertices

Algorithm : MNS

input : a graph $G = (V, E)$ and start vertex $s \in V$
output: an ordering σ of V

assign the label $\emptyset$ to all vertices
$label(s) \leftarrow \{n + 1\}$
for $i \leftarrow 1$ *to* n **do**
    pick an unnumbered vertex v with maximal label (under set inclusion)
    $\sigma(i) \leftarrow v$ // This assigns to v the number i
    foreach *unnumbered vertex* w *adjacent to* v **do**
        add i to $label(w)$

Maximal Neighbourhood Search (MNS)

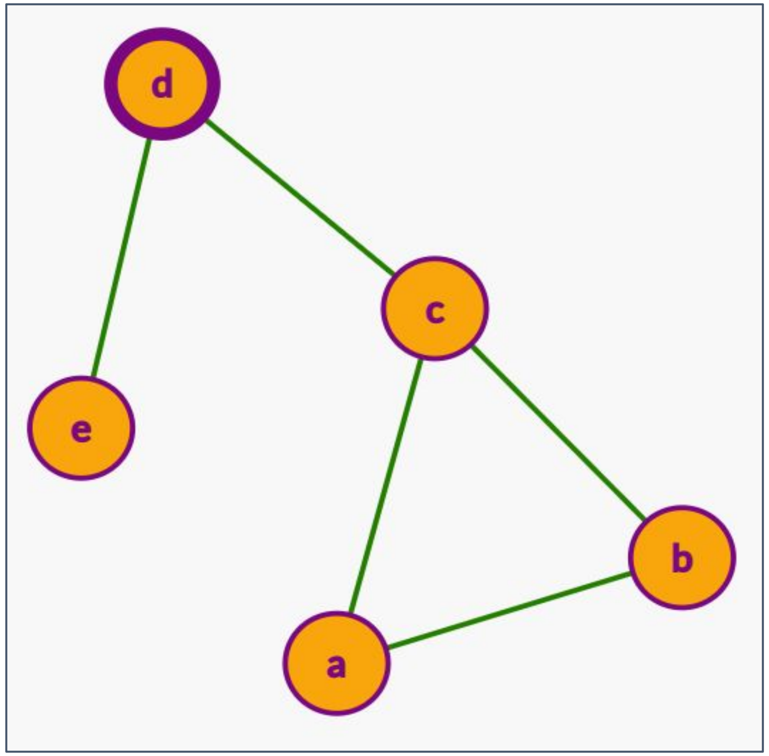
input : a graph $G = (V, E)$ and start vertex $s \in V$
output: an ordering σ of V

1. Each vertex in the graph has a label associated with it which can be treated as a set of integers.
2. Initially, set the label of starting vertex **s** as {n + 1} where n is the number of vertices in the graph. Set the label for other vertices in graph to an empty set {.}.
3. During each iteration, choose a vertex u such that it not yet selected (numbered) in the vertex ordering σ such that label corresponding to u is maximal (under the set inclusion).
4. Then add u to the vertex ordering σ and insert integer k (iteration number) into the label of all its non selected (numbered) neighbouring vertices.
5. Repeat the same process until all the vertices of the graph G are added to the vertex ordering σ.

MNS Example :

For a vertex $u \in V(G)$, $label(u)$ is maximal iff for all vertices $v \in V(G)$ which are not yet explored, $\exists\ e \in label(u)$ such that $e \notin label(v)$.

- ★ Consider, the following 5 vertex graph G with starting vertex **d**.
- ★ One of the valid MNS ordering for this graph is {d, c, a, e, b}.
- ★ The following vertex ordering is a convincing example for illustrating MNS rule.
- ★ e is explore before b because e has maximal label (and not maximum).



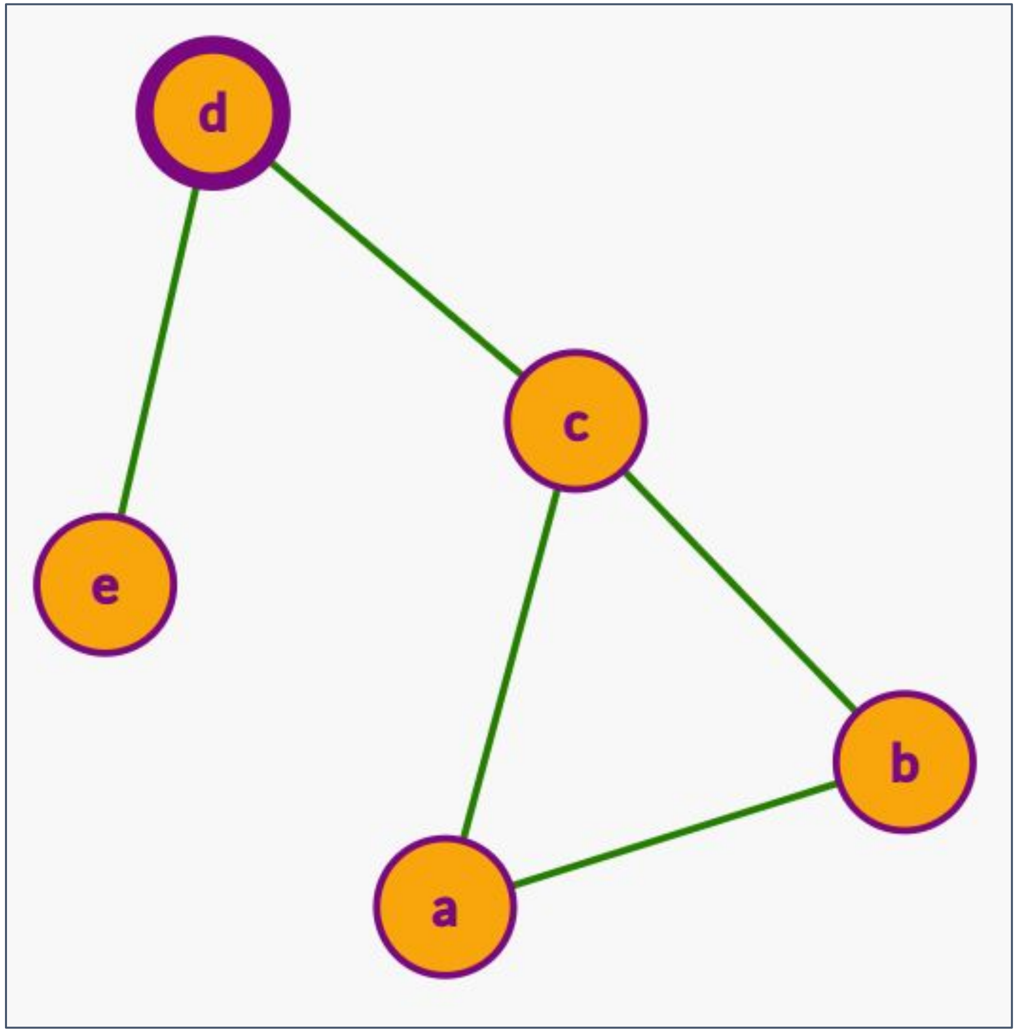
Maximum Cardinality Search (MCS)

input : a graph $G = (V, E)$ and start vertex $s \in V$
output: an ordering σ of V

1. Each vertex in the graph has a label associated with it which can be treated as a set of integers.
2. Initially, set the label of starting vertex **s** as {n + 1} where n is the number of vertices in the graph. Set the label for other vertices in graph to an empty set {.}.
3. During each iteration, choose a vertex u such that it not yet selected (numbered) in the vertex ordering σ such that label corresponding to u is has maximum number of elements (cardinality of label of u maximum).
4. Then add u to the vertex ordering σ and insert integer k (iteration number) into the label of all its non selected (numbered) neighbouring vertices.
5. Repeat the same process until all the vertices of the graph G are added to the vertex ordering σ.

MCS Example :

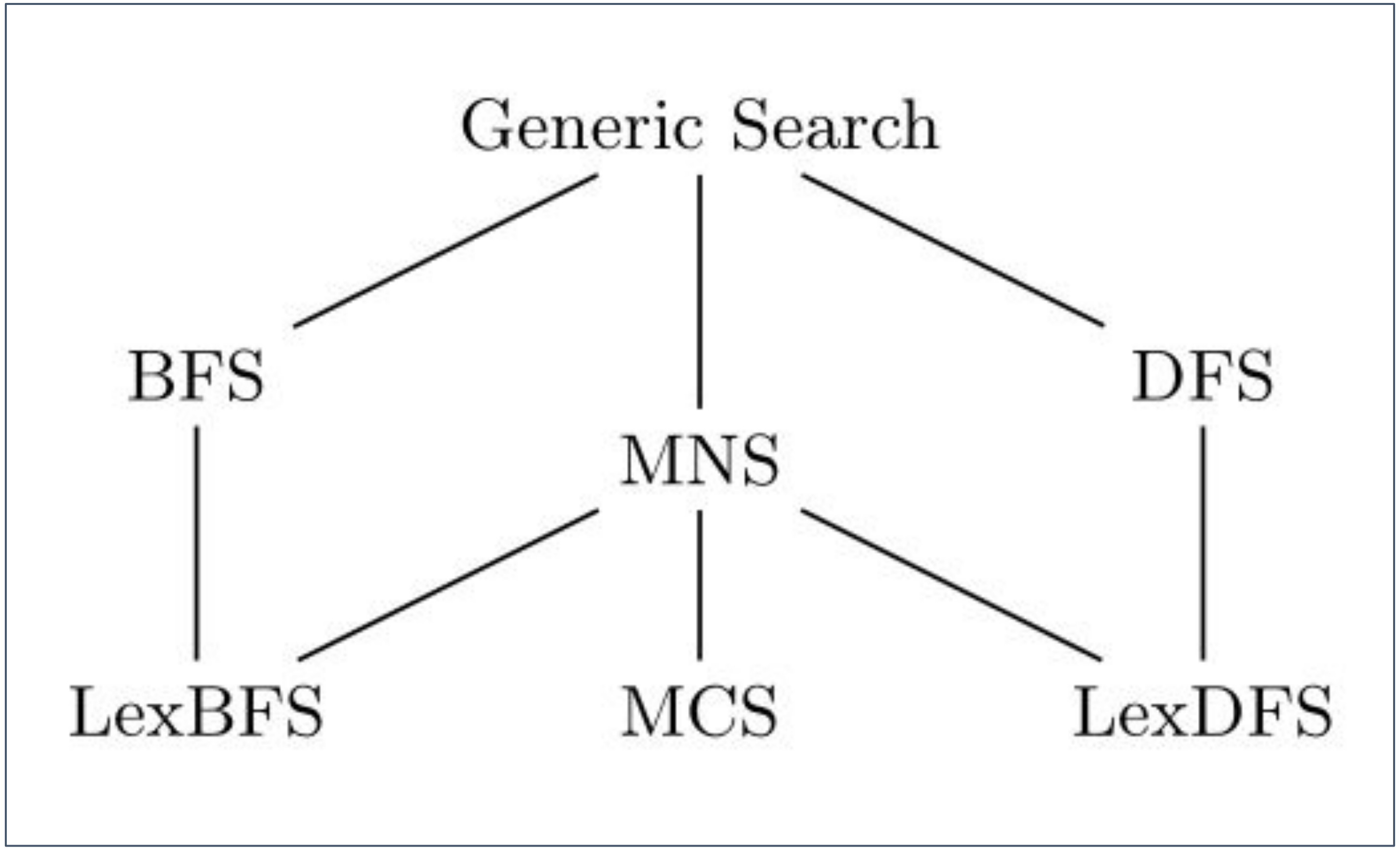
- ★ Consider, the following 5 vertex graph G with starting vertex **d**.
- ★ One of the valid MCS ordering for this graph is {d, c, a, b, e}.
- ★ The following vertex ordering is a convincing example for illusting MCS rule.
- ★ e is explored after b because when we visit {d, c, a}, label corresponding to b is maximum.



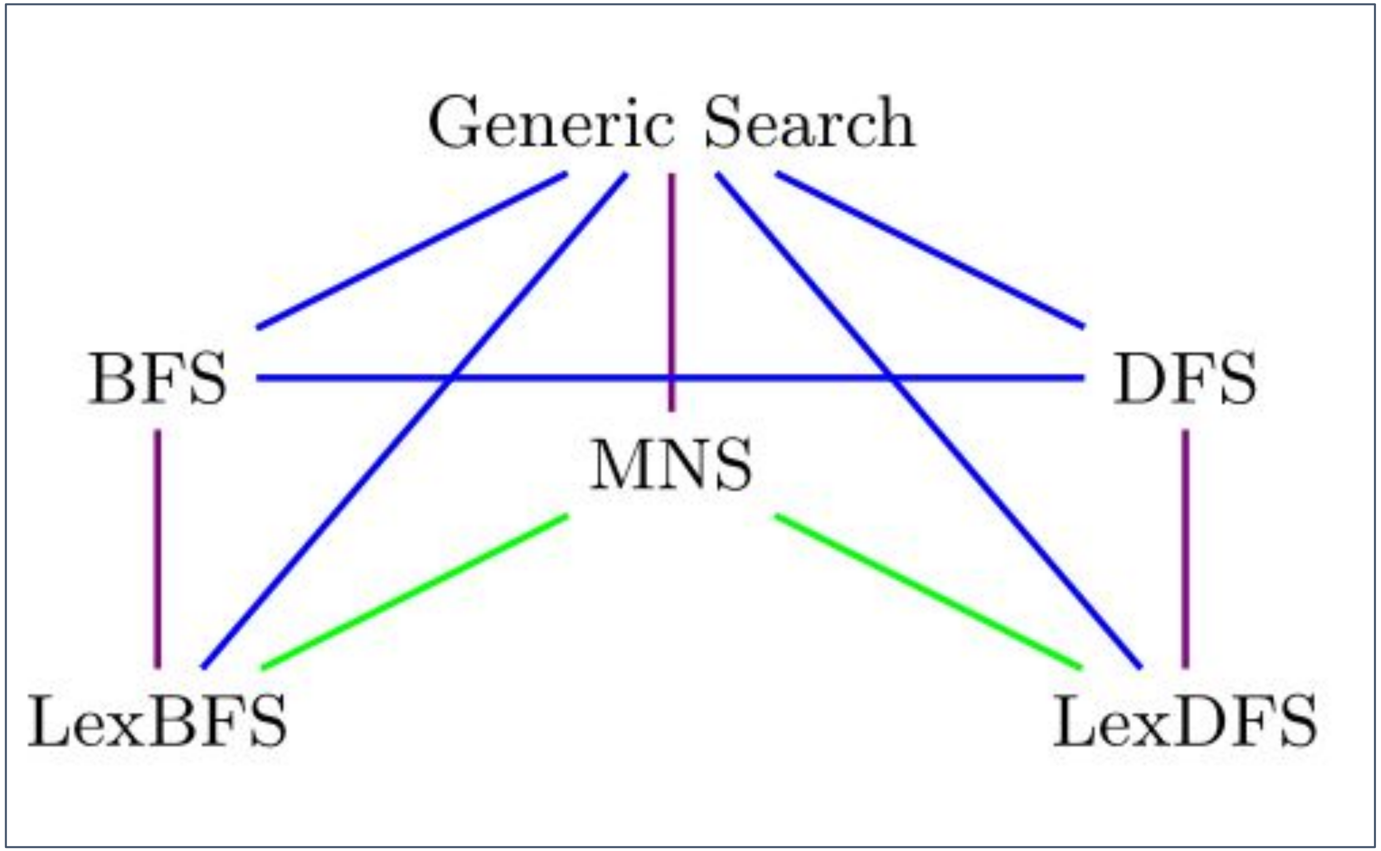
Graph Search Equivalency and our motive

- ★ Two graph searches are equivalent on a graph G if the lists of vertex ordering sequences produced by both are the same.
- ★ Lists of vertex ordering sequences is the set of all possible vertex ordering possible in a graph G using a search method.
- ★ It is evident that every valid MCS vertex ordering gives a valid MNS vertex ordering for any connected graph because MNS is a more generalized version of MCS. But the converse may not be true for some of the connected graphs.
- ★ Our ***AIM*** is to find the class of graphs for which MCS and MNS are equivalent.
- ★ For such graphs, there should not exist any valid MNS ordering σ which is not a valid MCS ordering.
- ★ If this is the case, then we can say that the two search methods are indistinguishable on these class of graphs.

Hasse Diagram

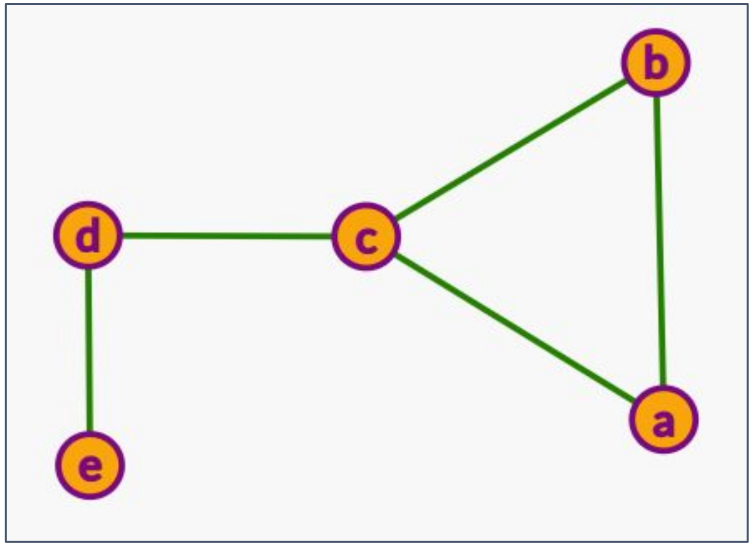
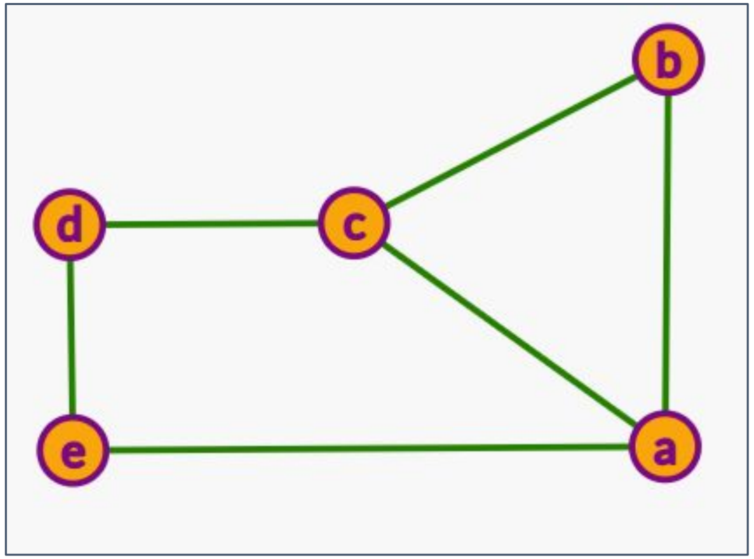
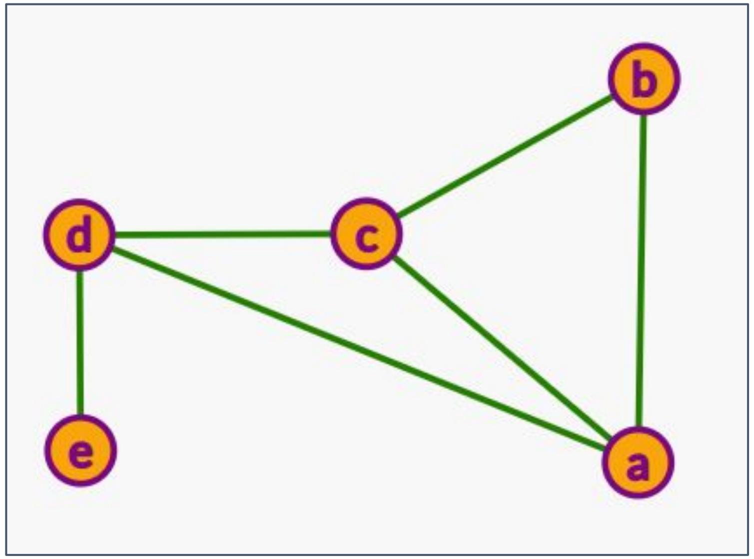
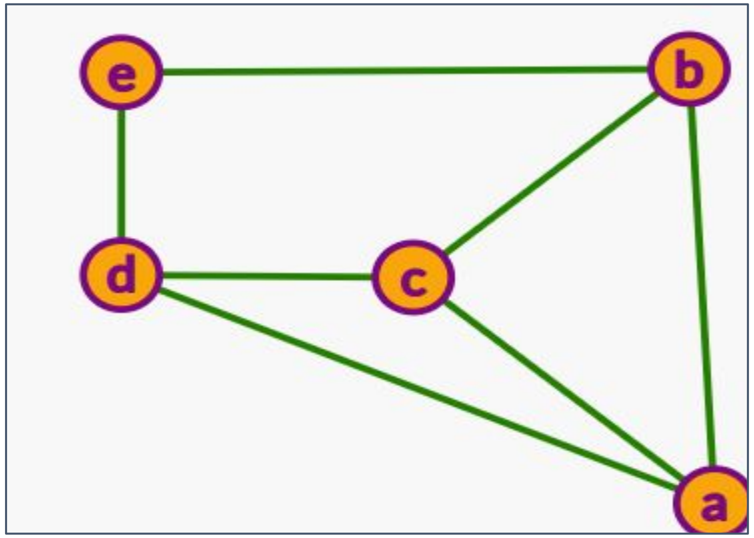
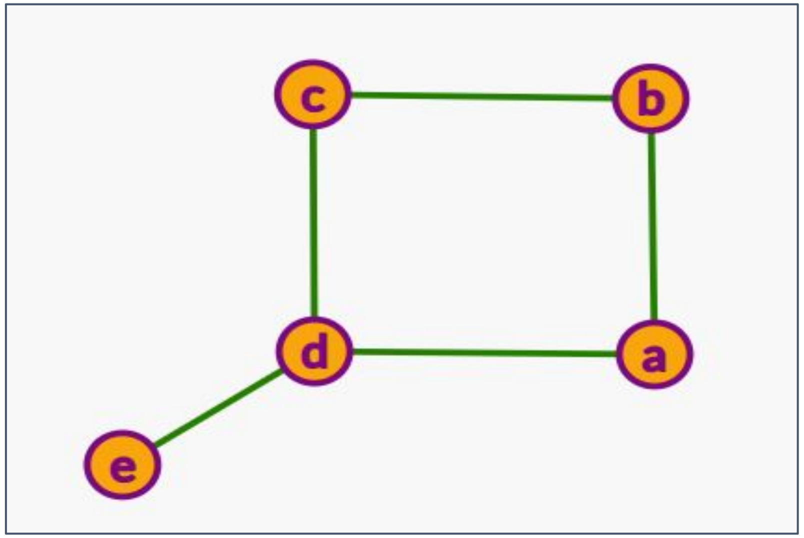


Some of the already known graph search equivalency.



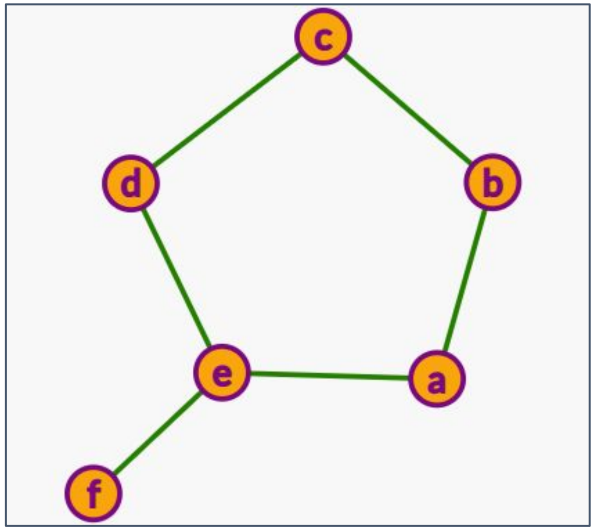
Interesting Graphs (IG)

- ★ IG is the set of all graphs G such that, for each graph G there should exist a valid MNS ordering σ which is not a valid MCS ordering.
- ★ Using a recursive and backtracking notioned program, we can generate the lists of all valid vertex orderings corresponding to a search technique.
- ★ There are around 21 simple connected graphs of 5 vertices. We can find all the interesting graphs of 5 vertices using the following technique.
- ★ Generate the lists of all possible MNS and MCS vertex ordering of that graph. If both the lists are of different size then the graph is interesting.
- ★ There are only 5 graphs which were found to be interesting.


b
d
c
a
e

b
d
c
e
a

b
d
c
e
a

e
b
d
c
a

c
b
d
a
e

Interesting Graphs of 6 vertices

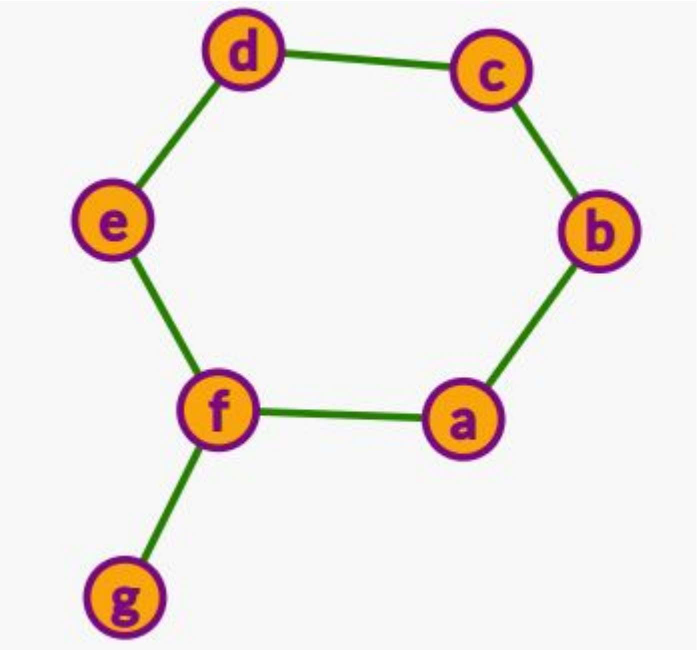
- ★ There a total of 112 structurally distinct 6 vertex connected simple graphs.
- ★ 69 of them are interesting and the remaining 43 are not interesting.
- ★ Out of these 69 interesting graphs, 68 of them have at least one vertex induced subgraph which is interesting.
- ★ There is only 1 graph which is interesting and does not have vertex induced subgraph which a 5-cycle with a pendant vertex.
- ★ The fact that there is only one such graph is very fascinating.



5-cycle with a pendant vertex

Interesting Graphs of 7 vertices

- ★ There a total of 853 structurally distinct 7 vertex connected simple graphs.
- ★ 740 of them are interesting and the remaining 113 are not interesting.
- ★ Again, out of these 740 interesting graphs, 739 of them have at least one vertex induced subgraph which is interesting.
- ★ There is only 1 graph which is interesting and does not have vertex induced subgraph which a 6-cycle with a pendant vertex.



6-cycle with a pendant vertex

Conjecture 1

- ★ Let C be the set of all the k-cycle graphs with pendant vertex such that $k \geq 4$.
- ★ Let B be the set the four interesting graphs excluding the the graph with 4-cycle and a pendant vertex.

Statement: Given an interesting graph G, either it must belong to the set C or B or must have at least one vertex induced subgraph which is interesting.

➢ Using recursive approach, the statement can be modified as :

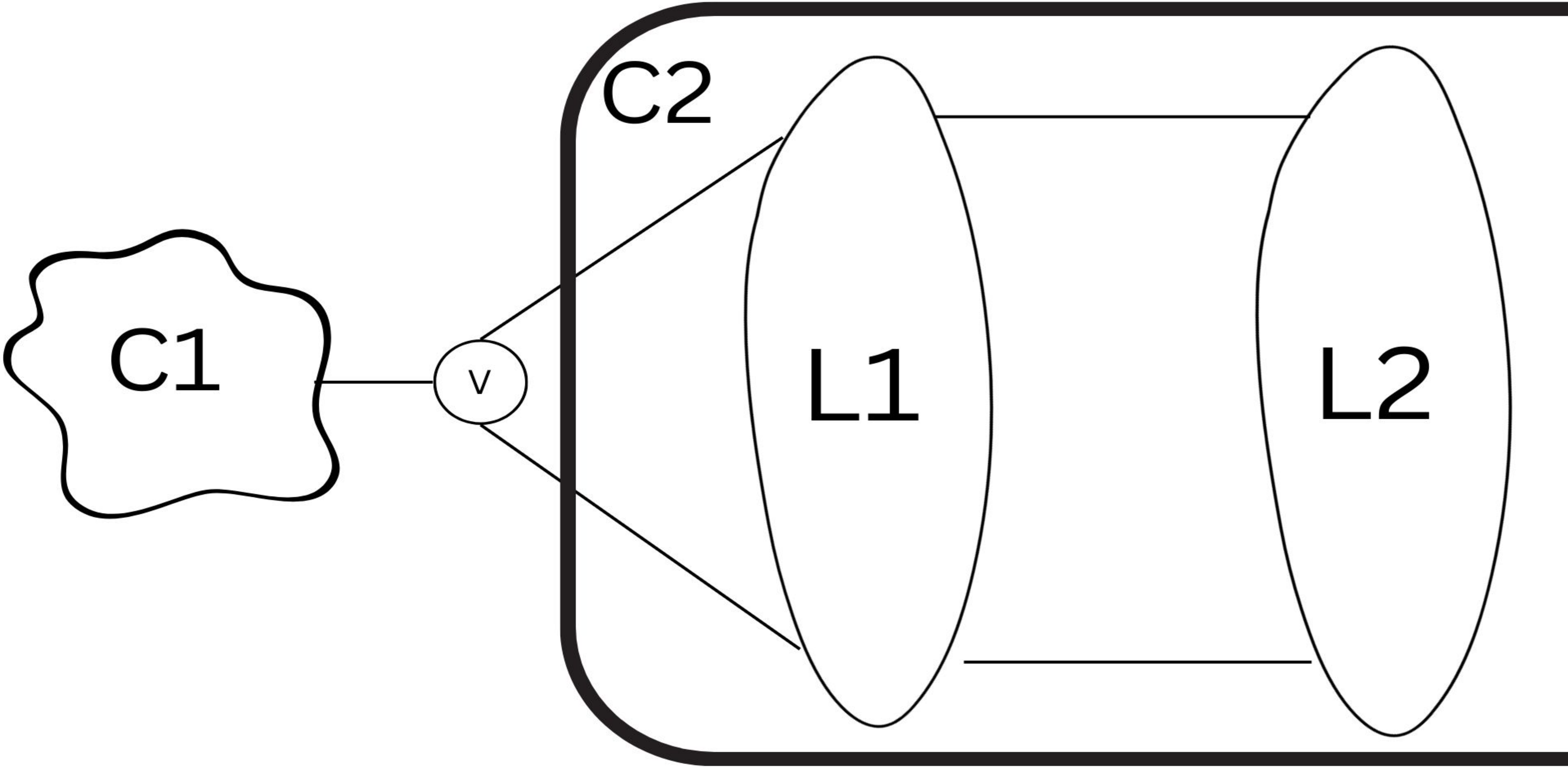
Given an interesting graph, either it must belong to the set C or B or must have at least one vertex induced subgraph which belong to C or B.

__Theorem 1__: Given a graph G, let v be an universal vertex in the graph. G is interesting iff G-v is interesting.

__Theorem 2__: Given a graph G, let v be an isolated vertex in the graph. G is interesting iff G-v is interesting.

Characterizing a graph.

★ Let v be a cut vertex of the graph G. Removing that vertex will create at least two components C1 and C2 and maybe more.
★ Visualize C2 as layers of vertices (L1, L2, ..., Lk).
★ The vertices in C2 that have an edge with v is present in L1 layer.
★ Similarly, the vertices in C2 that have an edge with at least vertex in L1 be present in L2 and so on.
★ Picking vertex v will insert 1 in all the vertices adjacent to it. Let u be a vertex in C1 which is connected to v1.
★ We can explore all the vertices in L1 and add it to vertex ordering, then proceed to layer L2.
★ With a similar logic, all the vertices in the L2 can be explored before proceeding with L3.
★ Whenever there is edge between two vertices in the same layer Li where i > 1, then we can obtain a vertex ordering which is a MNS and not a MCS.
★ Whenever there are at least two edges to a vertex in layer Li from the layer L(i-1), where i > 1, then we can obtain a vertex ordering which is a MNS but not a MCS.


C2
C1
v
L1
L2

THANK YOU!