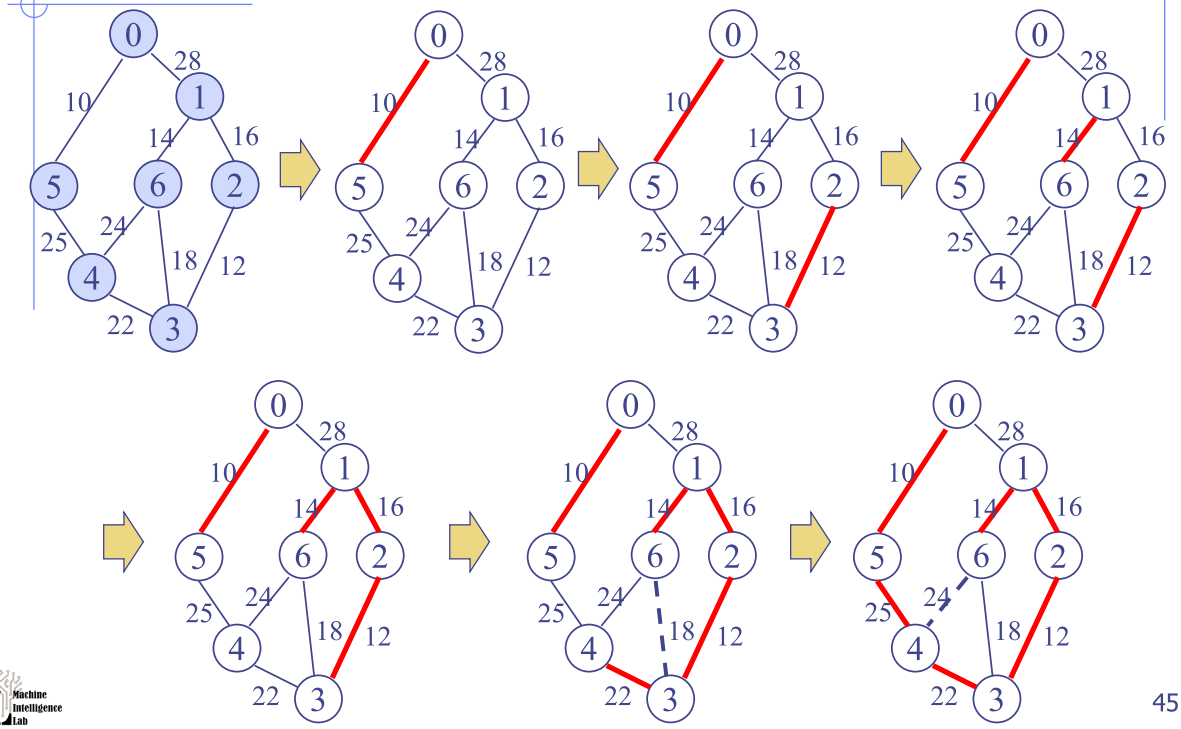
Minimum Cost Spanning Tree

- The cost of a spanning tree of a weighted undirected graph is the sum of the costs (weights) of the edges in the spanning tree.
- A minimum cost spanning tree is a spanning tree of least cost.
- Three different algorithms can be used
  - Kruskal's algorithm
  - Prim's algorithm
  - Sollin's algorithm

Kruskal's Algorithm

- Build a minimum cost spanning tree T by adding edges to T one at a time
- Select the edges for inclusion in T in nondecreasing order of their cost
- An edge is added to T if it does not form a cycle with the edges that are already in T
- Since G is connected and has n>0 vertices, exactly n-1 edges will be selected for inclusion in T

Kruskal's Algorithm



Kruskal's Algorithm

- Assumption
  - E: initially the set of all edges in G
  - maintaining edges in E as a sorted sequential list in $O(e \log e)$ time (Chapter 7) or selection sort (Chapter 1)
    - Obviously a *min heap* is ideally suited for this task since we can determine and delete the next least cost edge in $O(\log e)$ time.
    - Construction of the heap itself requires $O(e)$ time.

```
T= {};
while ( T contains less than n-1 edges && E is not empty )
{
        choose a least cost edge (v, w) from E;
        delete (v, w) from E;
        if ( (v,w) does not create a cycle in T )
                add (v, w) to T
        else
                discard (v, w);
}

if ( T contains fewer than n-1 edges )
        printf( "No spanning tree...\n" );
```

Use of the union-find operations in forests

- Initially T is empty and each vertex of G is in a different set.
- Before adding an edge (v, w), use the find operation to determine if v and w are in the same set (or tree)
- If not in the same set, use the union operation to add into T, resulting an extended set.