Kosen Club CP 勉強会

GRAPH-DFS, CC, TS, SCC

1.Graph Terms & Expression

1.Adjacency Table

int Graph[n][n];

G[i][j] = w; // Edge from i to j with weight w

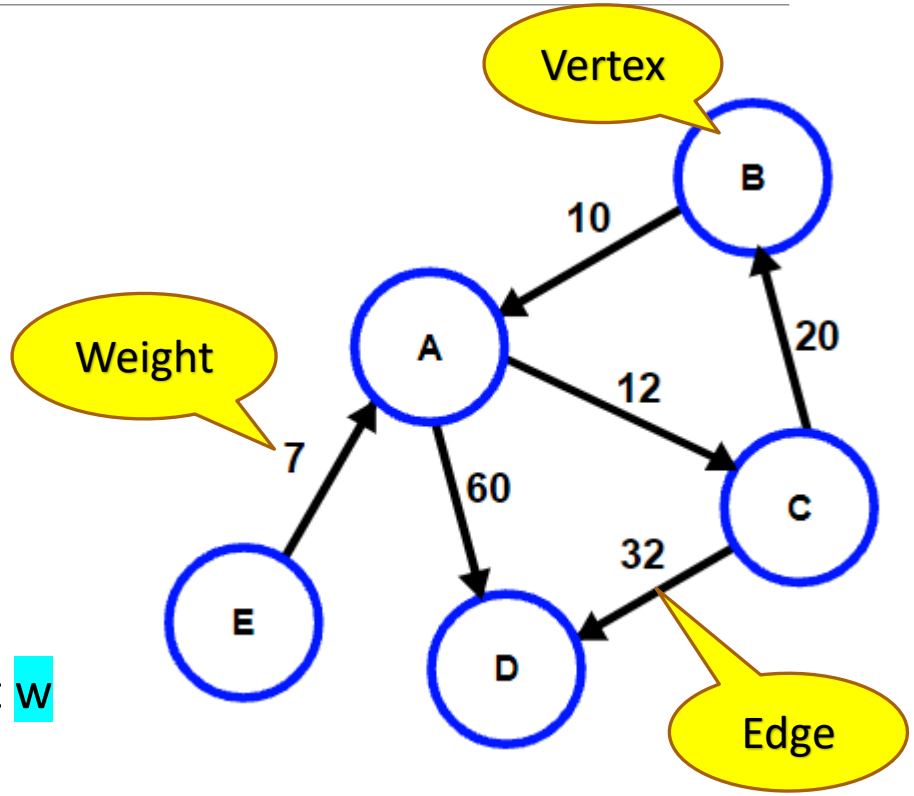
Memory: O(n*n)

2.Adjacency List

vector<int> Graph[n];

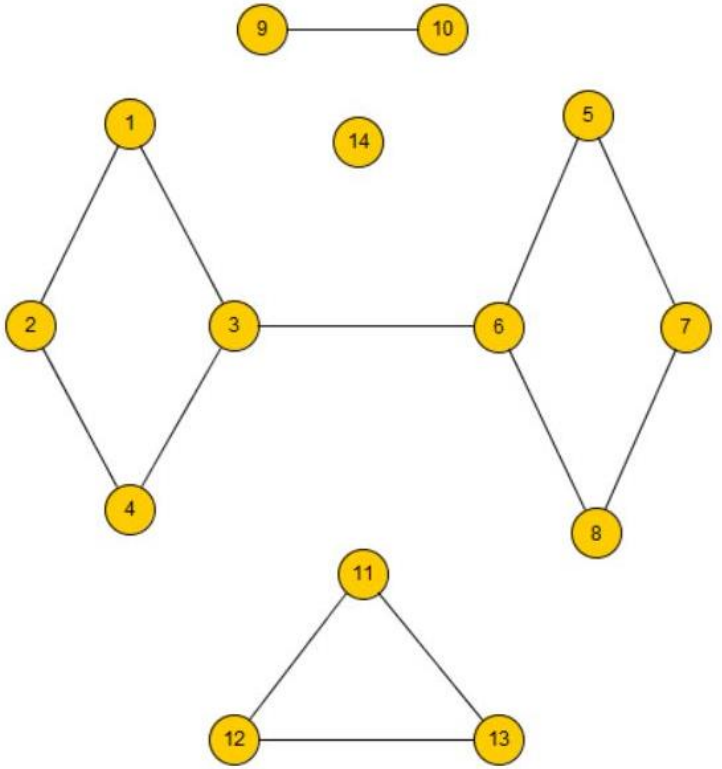
Graph[i].push_back(pair(j, w)); // Edge from i to j with weight w

Memory: O(n+m)



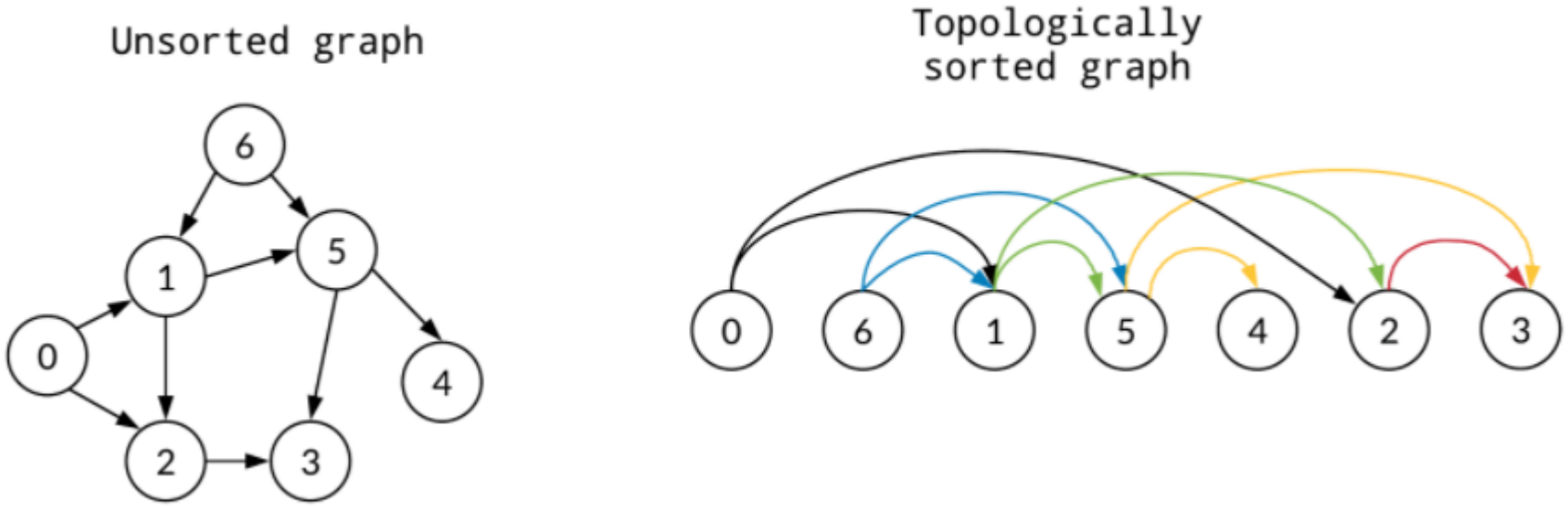
2.Connected Component

```
vector<int> g[N];
int comp[N] = {0};
void dfs(int cur, int color) {
    comp[cur] = color;
    for (auto nxt : g[cur])
        if (!comp[nxt]) dfs(nxt, color);
}
void connectedComponent(int n) {
    int color = 1;
    REP(i, n) {
        if (comp[i]) continue;
        dfs(i, color);
    }
}
```



3.Topological Sort (1/2)

A topological sort is an ordering of nodes for a directed acyclic graph **(DAG)** such that for every directed edge uv from vertex u to vertex v, u comes before v in the ordering.



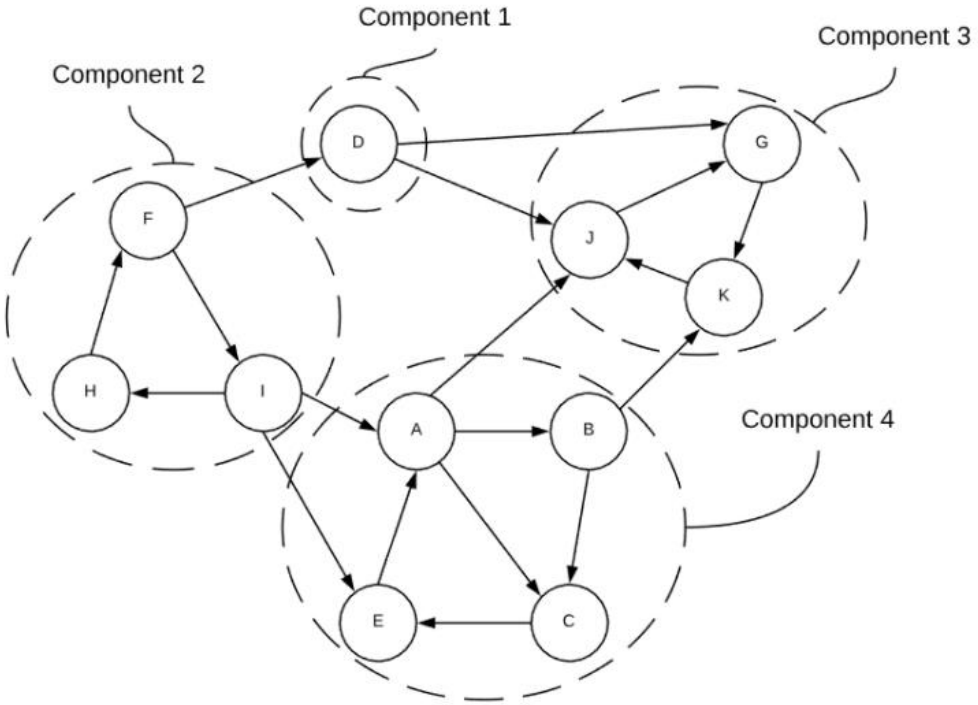
3.Topological Sort (2/2)

```
vector<int> g[N];
int visisted[N];
vector<int> order;
void dfs(int u) {
  visisted[u] = 1;
  for (int x : g[u]) {
    if (visisted[x] == 0)
     dfs(x);
  }
  order.push_back(u);
}
```

```
void topologicalSort(int n) {
  RREP(u, n) {
    if (visisted[u] == 0)
      dfs(u);
  }
  reverse(order.begin(), order.end());
  REP(i, n) rev_order[order[i]] = i;
  REP(i, m) {
    if (rev_order[edges[i][0]] >= rev_order[edges[i][1]])
      cout << “Found cycle” << endl;
  }
}
```

4.Strongly Connected Components (SCC)

A strongly connected component is a subsection of a directed graph in which there is a directed path from every vertex to every other vertex.



[強連結成分（SCC） | technical-note (hkawabata.github.io)]

4.Strongly Connected Components (SCC)

```
void Dfs(const int cur, std::vector<int> &ord) {
  scc[cur] = true;
  for (auto dst : adj[cur])
  if (!scc[dst]) Dfs(dst, ord);
  ord.push_back(cur);
}
void RevDfs(const int id, const int cur) {
  scc[cur] = id;
  for (auto dst : radj[cur])
    if (scc[dst] == -1) RevDfs(id, dst);
}
```

```
int StronglyConnectedComponents() {
  std::vector<int> ord(n);
  for (int v = 0; v < n; ++v)
    if (!scc[v]) Dfs(v, ord);
  std::fill(scc.begin(), scc.end(), -1);
  for (int i = n - 1; 0 <= i; --i)
  if (scc[ord[i]] == -1)
    RevDfs(num_comp++, ord[i]);
  return num_comp;
}
```

algorithm/strongly_connected_component_kosaraju.cc at master · shiitada/algorithm (github.com)