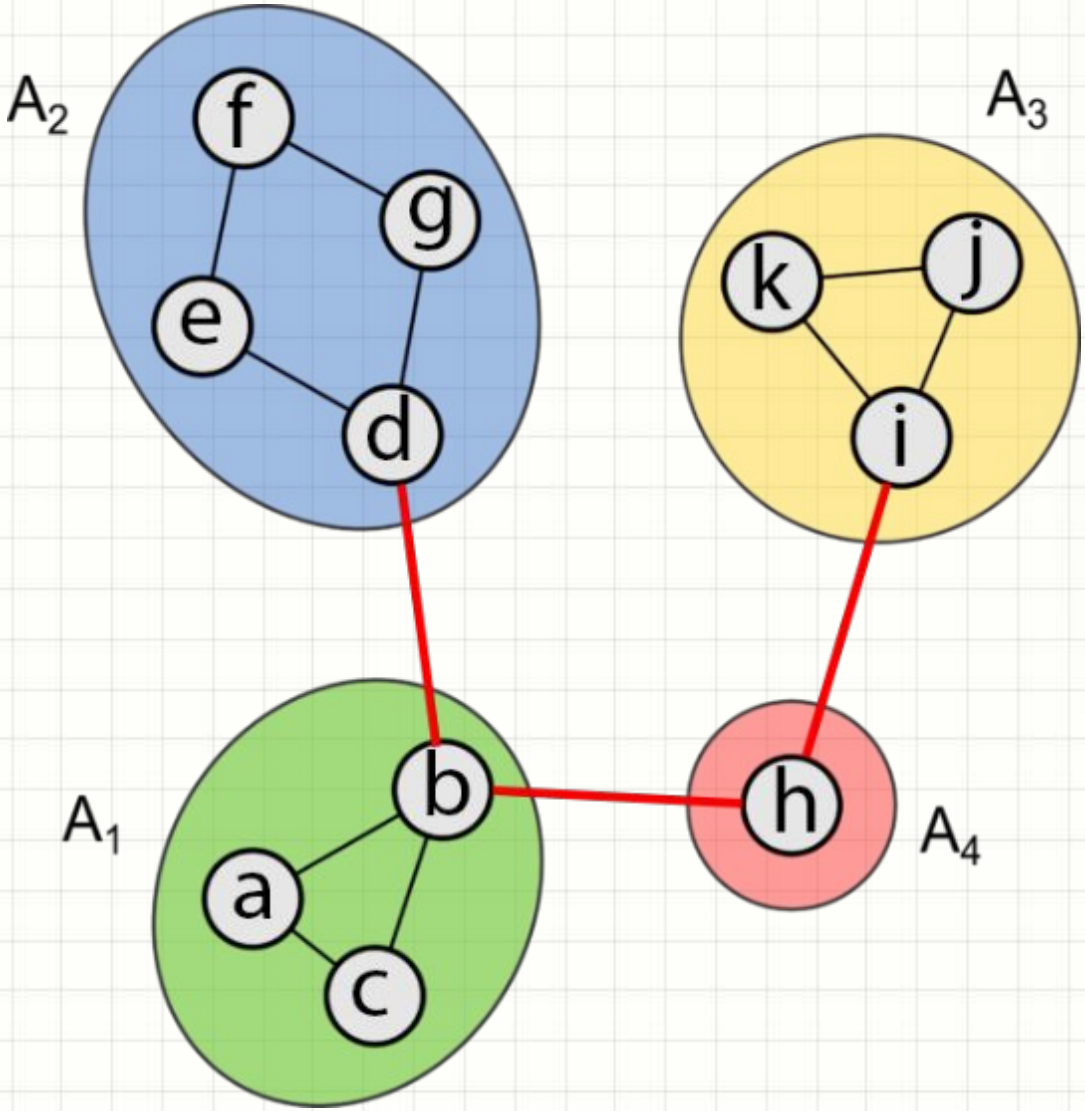
Bridges and Cut Vertices

Lyzhin Ivan, 2016

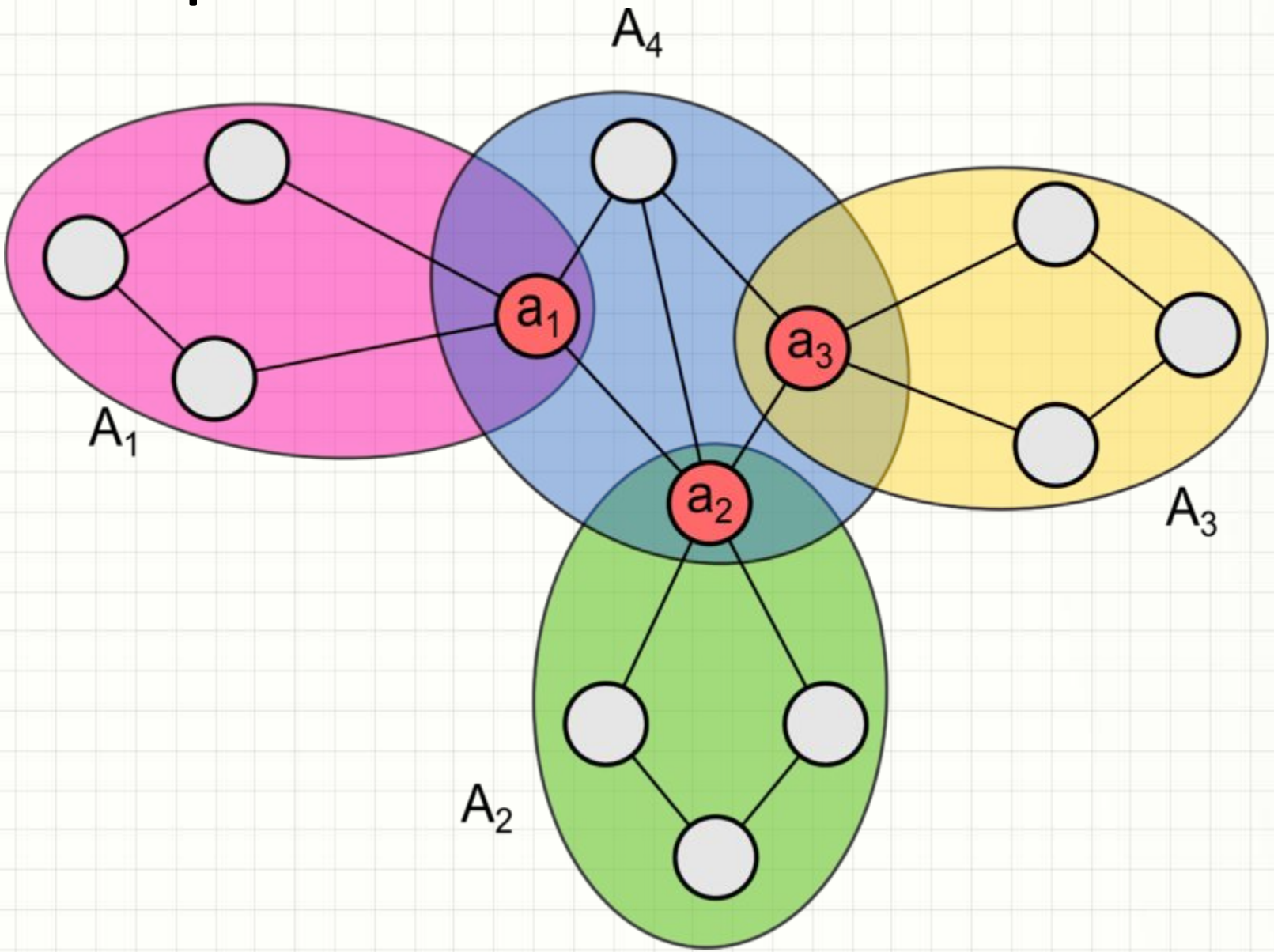
Definitions

- *Bridge* – an edge of a graph whose deletion increases the number of connected components.
- *Cut vertex* – a vertex whose deletion increases the number of connected components.

Example - Bridges

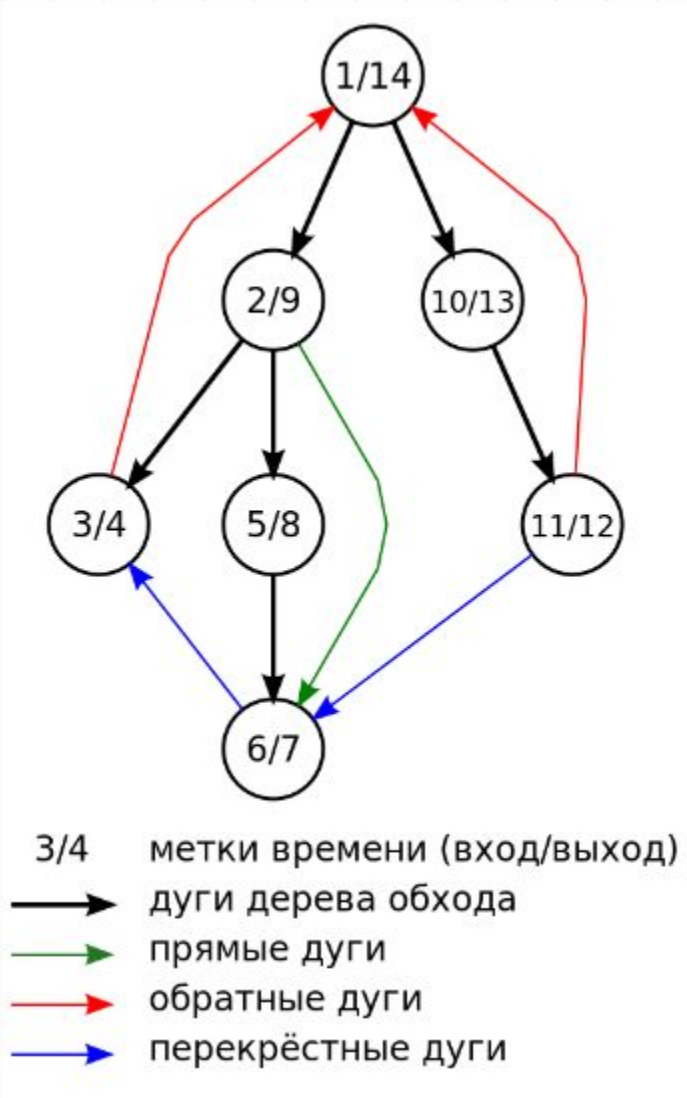


Example – Cut Vertices



Definitions for Depth-First-Search in undirected graph

- *Tree edge* –
  edge to unvisited vertex.
- *Back edge* –
  edge to visited vertex.
- *Parent edge* –
  edge to parent vertex.



Algorithm - Bridges

- Start DFS from unvisited vertex.
- $tin[v]$ – entry time for vertex.
- $fup[v] = min \begin{cases} tin[v] \\ tin[p], \ for\ all\ (v,p) - back\ edge \\ fup[to], \ for\ all\ (v,to) - tree\ edge \end{cases}$
- If $fup[to] > tin[v]$, then (v, to) is bridge.

Implementation - Bridges

```
void dfs(int v, int p = -1) {
    used[v] = true;
    tin[v] = fup[v] = timer++;
    for (size_t i = 0; i<g[v].size(); ++i) {
        int to = g[v][i];
        if (to == p)  continue; // Parent edge
        if (used[to]) // Back edge
            fup[v] = min(fup[v], tin[to]);
        else { // Tree edge
            dfs(to, v);
            fup[v] = min(fup[v], fup[to]);
            if (fup[to] > tin[v])
                IS_BRIDGE(v, to);
        }
    }
}
```

Algorithm – Cut Vertices

- Start DFS from unvisited vertex.
- $tin[v]$ – entry time for vertex.
- $$fup[v] = min \begin{cases} tin[v] \\ tin[p], \ for\ all\ (v,p) - back\ edge \\ fup[to], \ for\ all\ (v,to) - tree\ edge \end{cases}$$
- If $fup[to] \geq tin[v]$ and $v \neq root$, then v is cut vertex.
- If $v = root$ and $children > 1$, then v is cut vertex.

Implementation – Cut Vertices

```
void dfs(int v, int p = -1) {
    used[v] = true;
    tin[v] = fup[v] = timer++;
    int children = 0; // Number of children
    for (size_t i = 0; i<g[v].size(); ++i) {
        int to = g[v][i];
        if (to == p)  continue; // Parent edge
        if (used[to]) // Back edge
            fup[v] = min(fup[v], tin[to]);
        else { // Tree edge
            dfs(to, v);
            fup[v] = min(fup[v], fup[to]);
            if (fup[to] >= tin[v] && p != -1)
                IS_CUTPOINT(v);
            ++children;
        }
    }
    if (p == -1 && children > 1) // If root
        IS_CUTPOINT(v);
}
```

Additional links and home task

- Bridge searching
  http://e-maxx.ru/algo/bridge_searching
- Cut vertex searching
  http://e-maxx.ru/algo/cutpoints
- Tasks
  http://codeforces.com/group/Hq4vrJcA4s/contest/100703/problem/E