Graph Traversals, Part 1

The Basics of DFS and BFS

Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Fall, 2022

Outline

Introduction

Depth First Search

Breadth First Search

Applications

Objectives

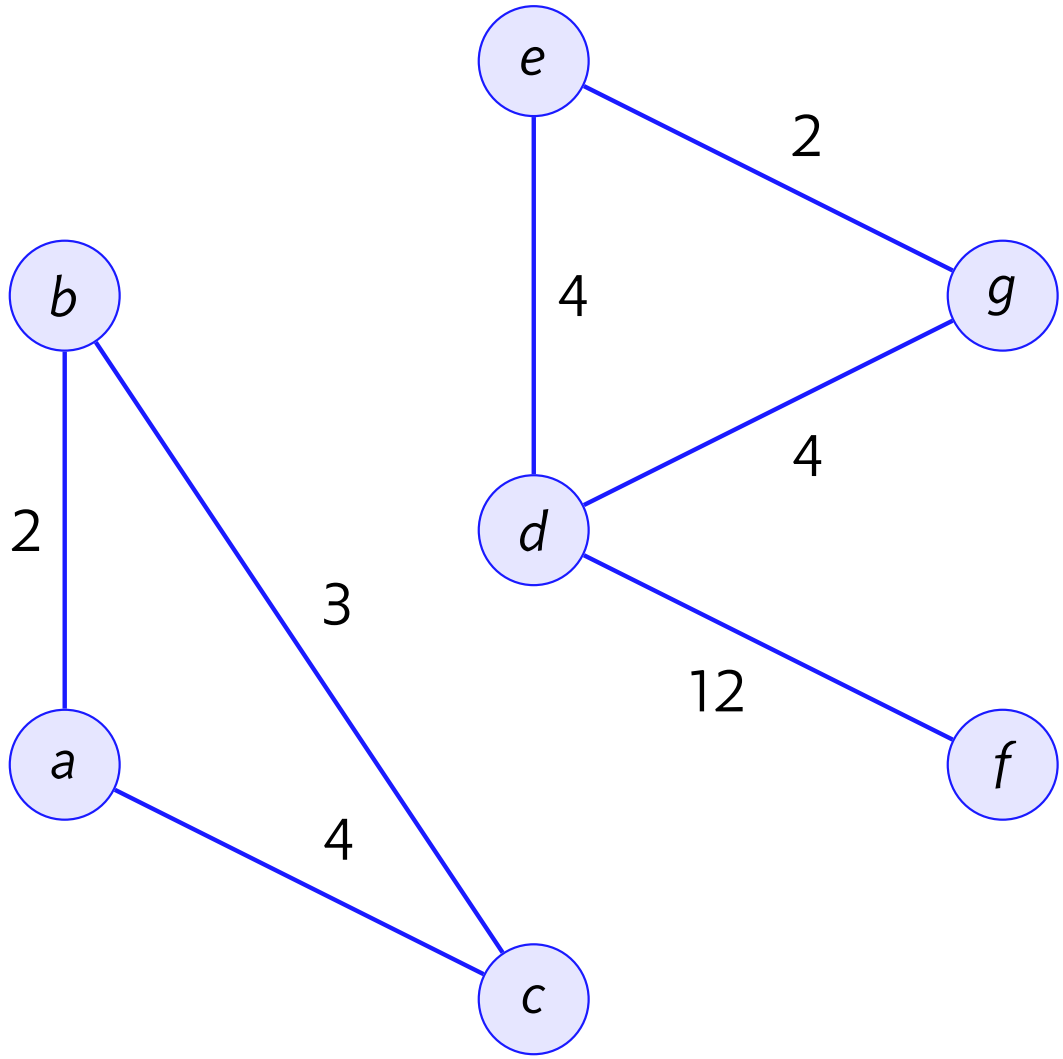
- Implement DFS and BFS
- Show how to use these to solve some classic graph problems:
  - connected components
  - flood fill

DFS Basics

- Step 1: Mark self as visited
- Step 2: Visit all unvisited children
- Step 3: ???
- Step 4: Profit!

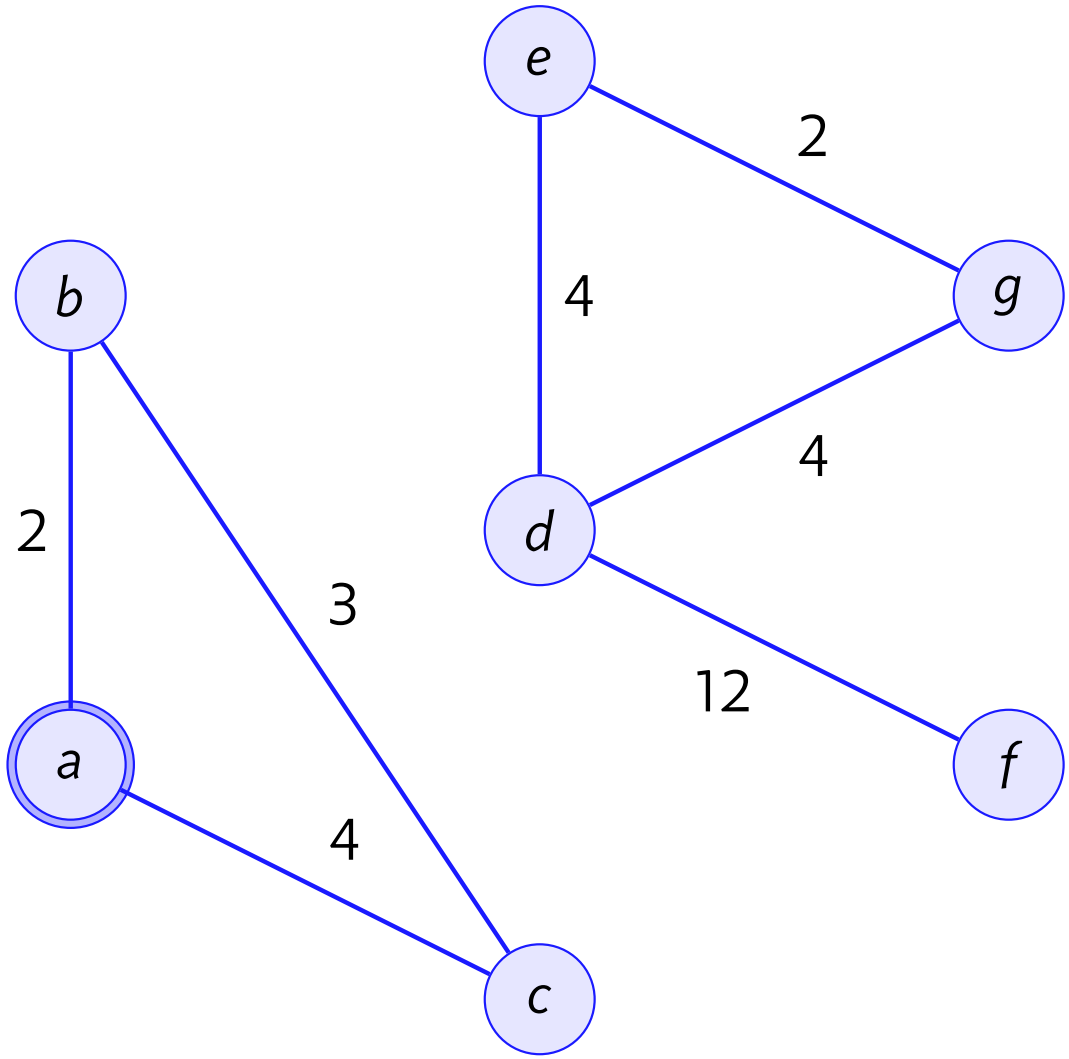
Example

- Here's our happy tree from last time.



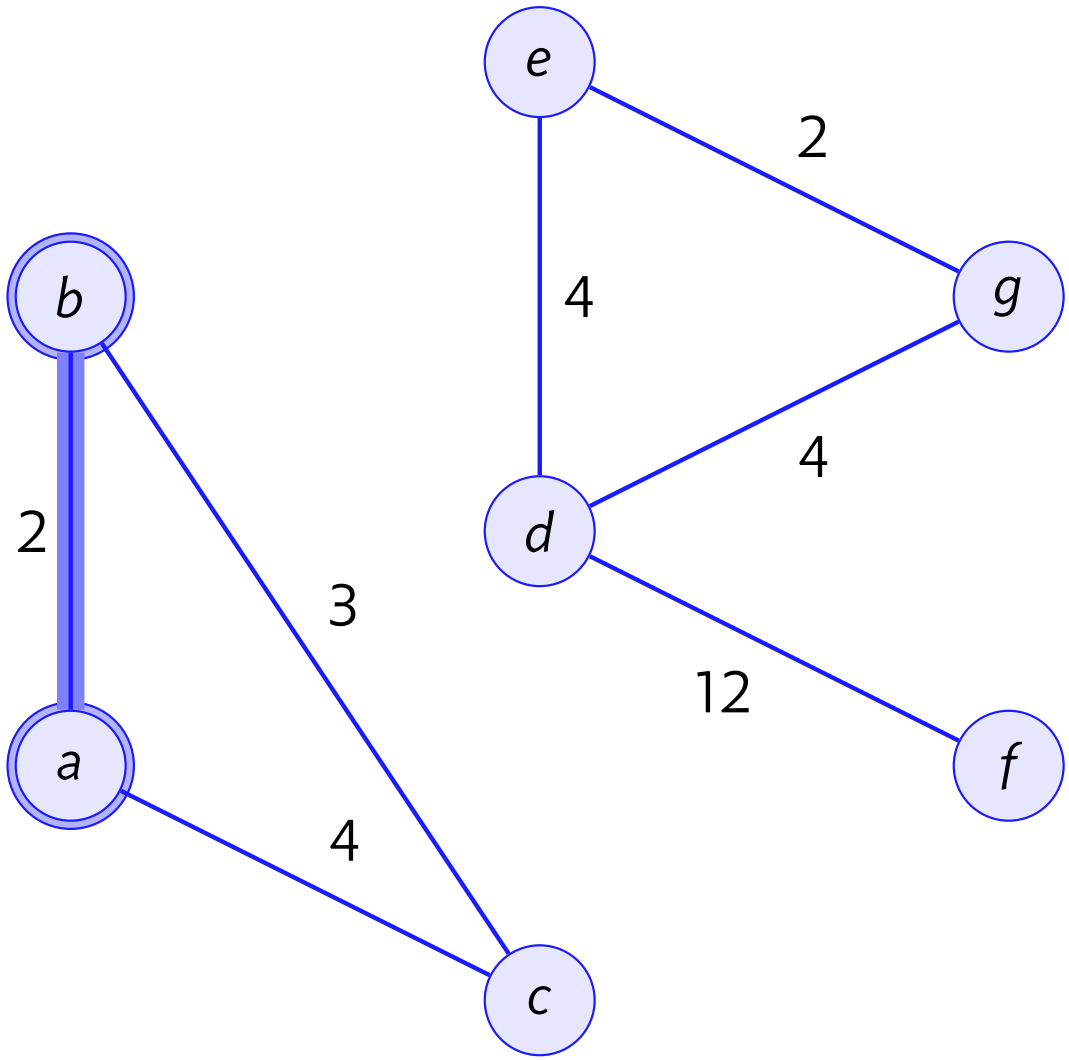
Example

- Here's our happy tree from last time.



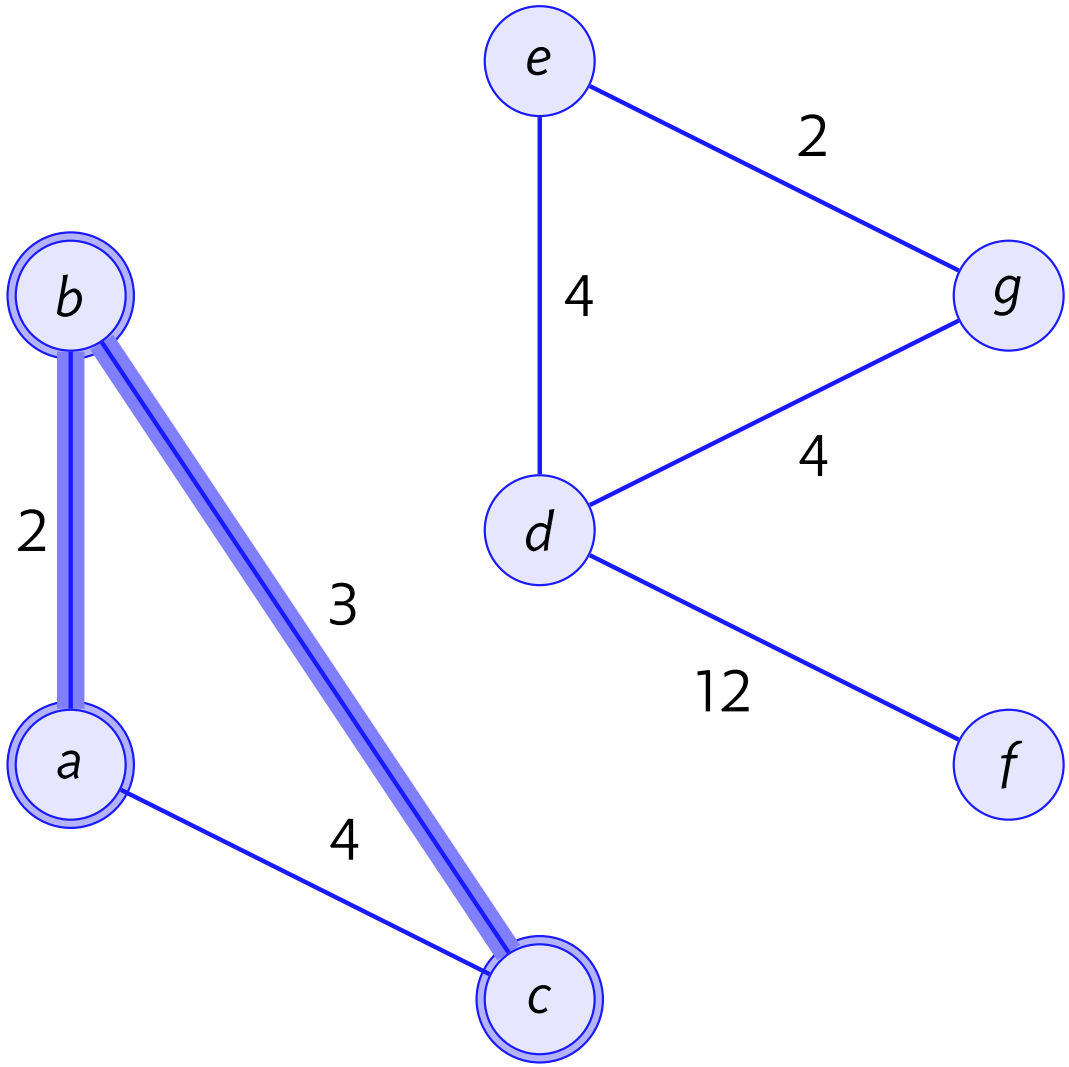
Example

- Here's our happy tree from last time.



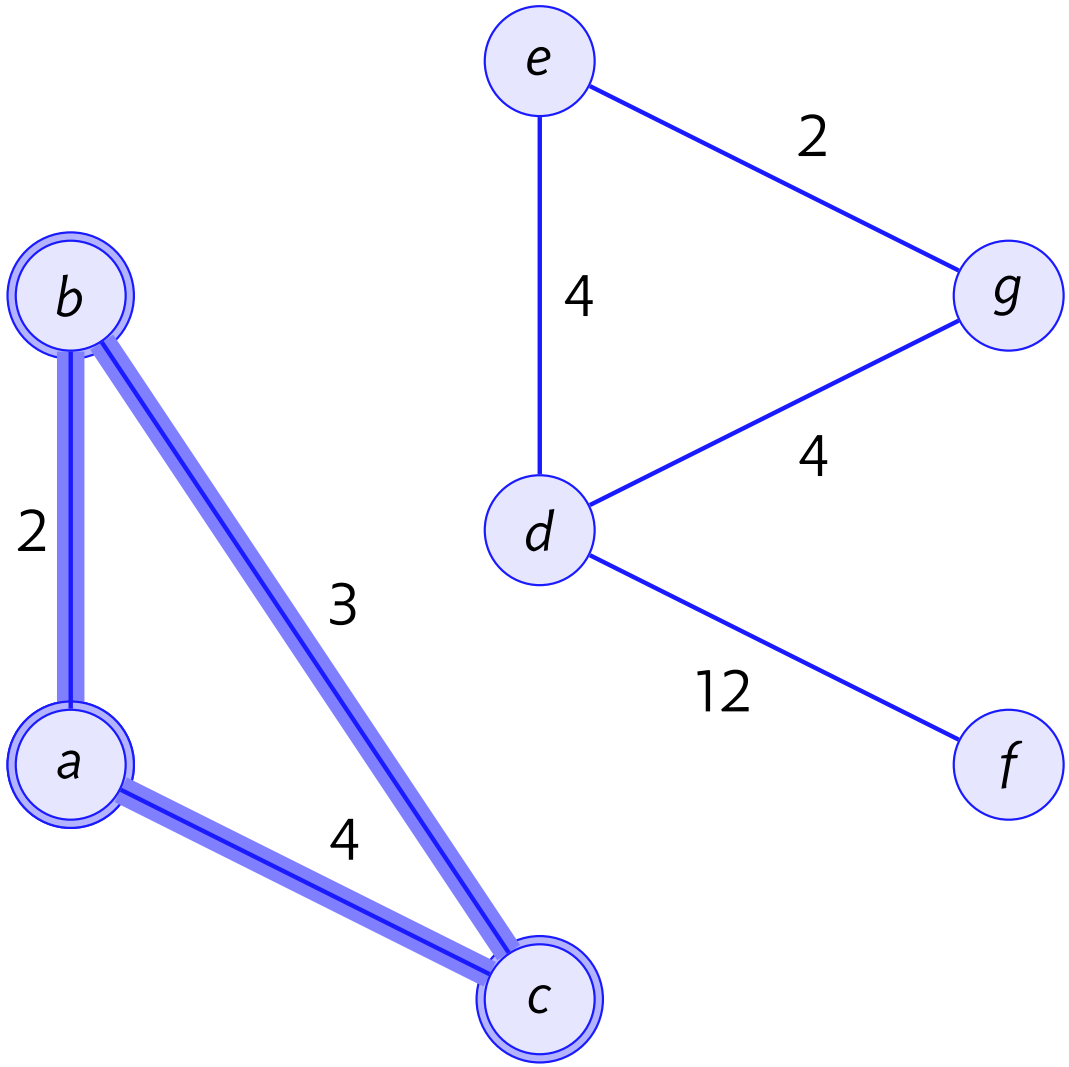
Example

- Here's our happy tree from last time.



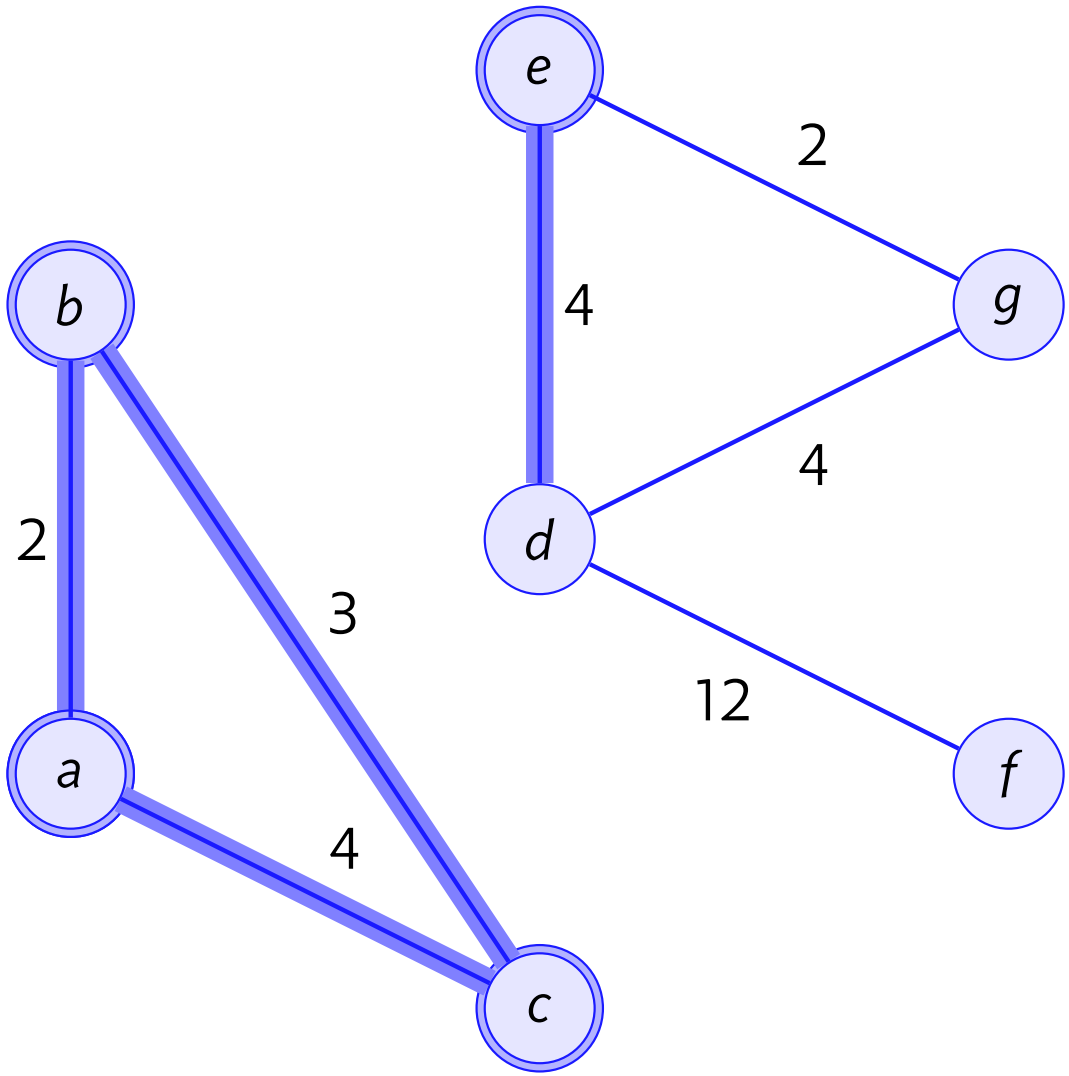
Example

- Here's our happy tree from last time.



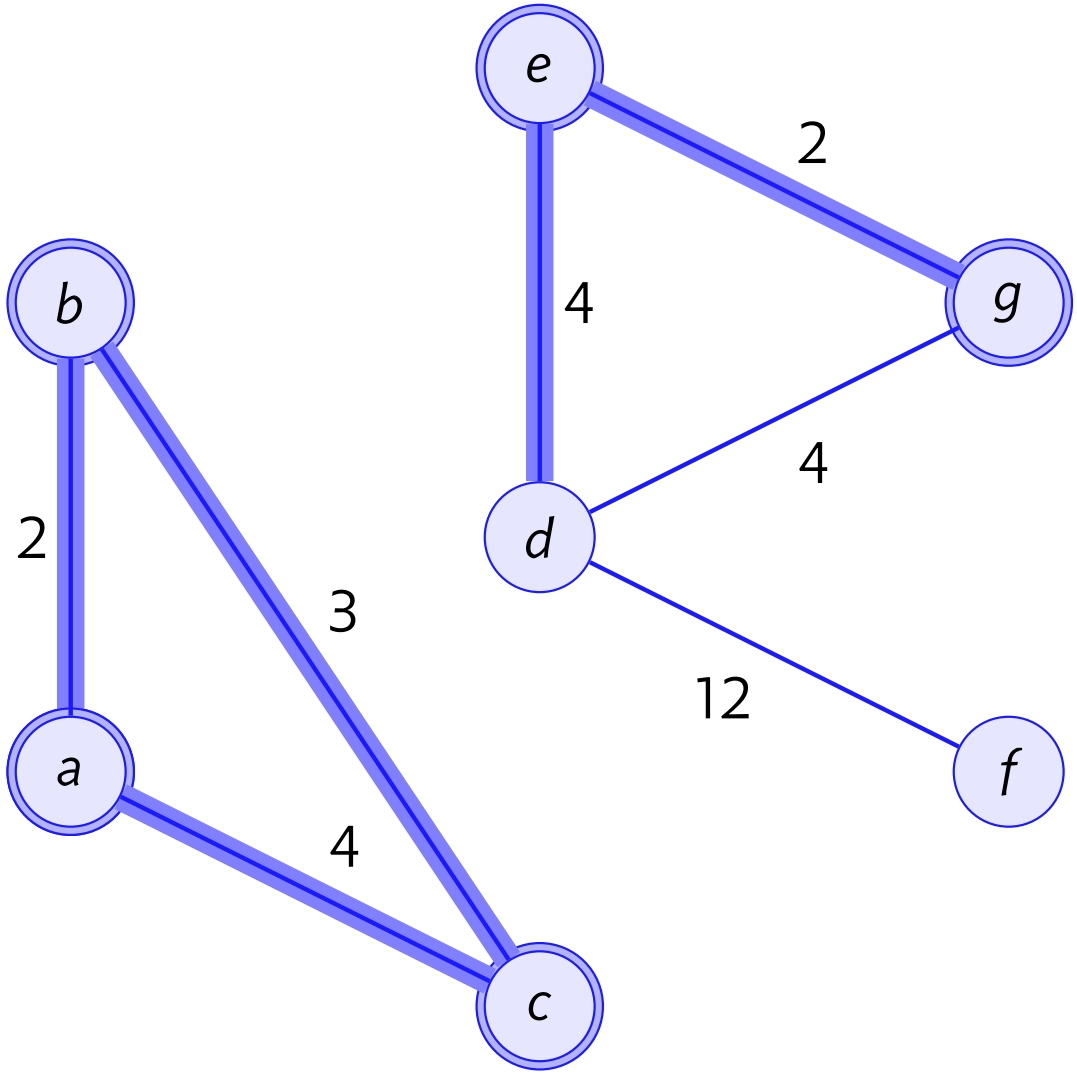
Example

- Here's our happy tree from last time.



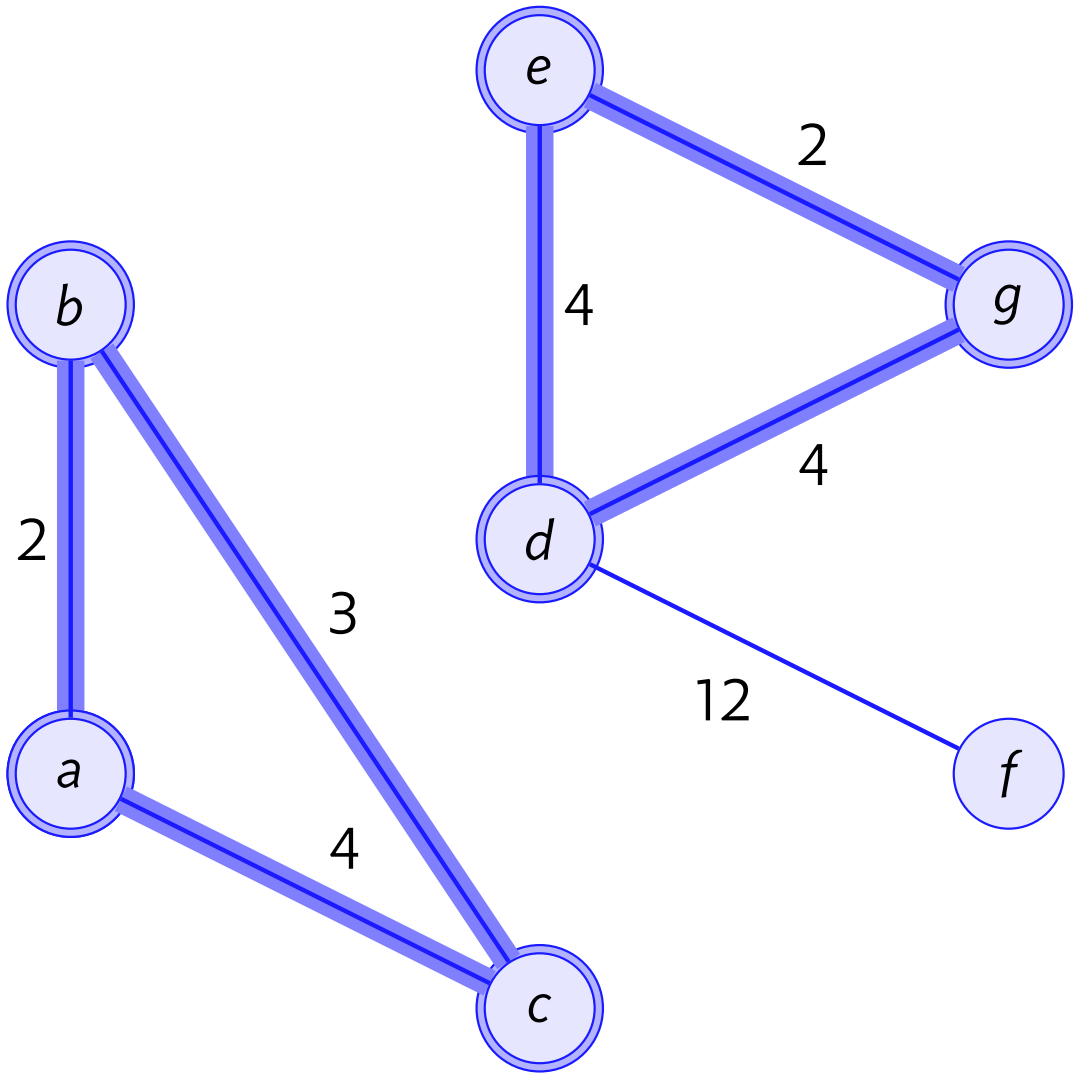
Example

- Here's our happy tree from last time.



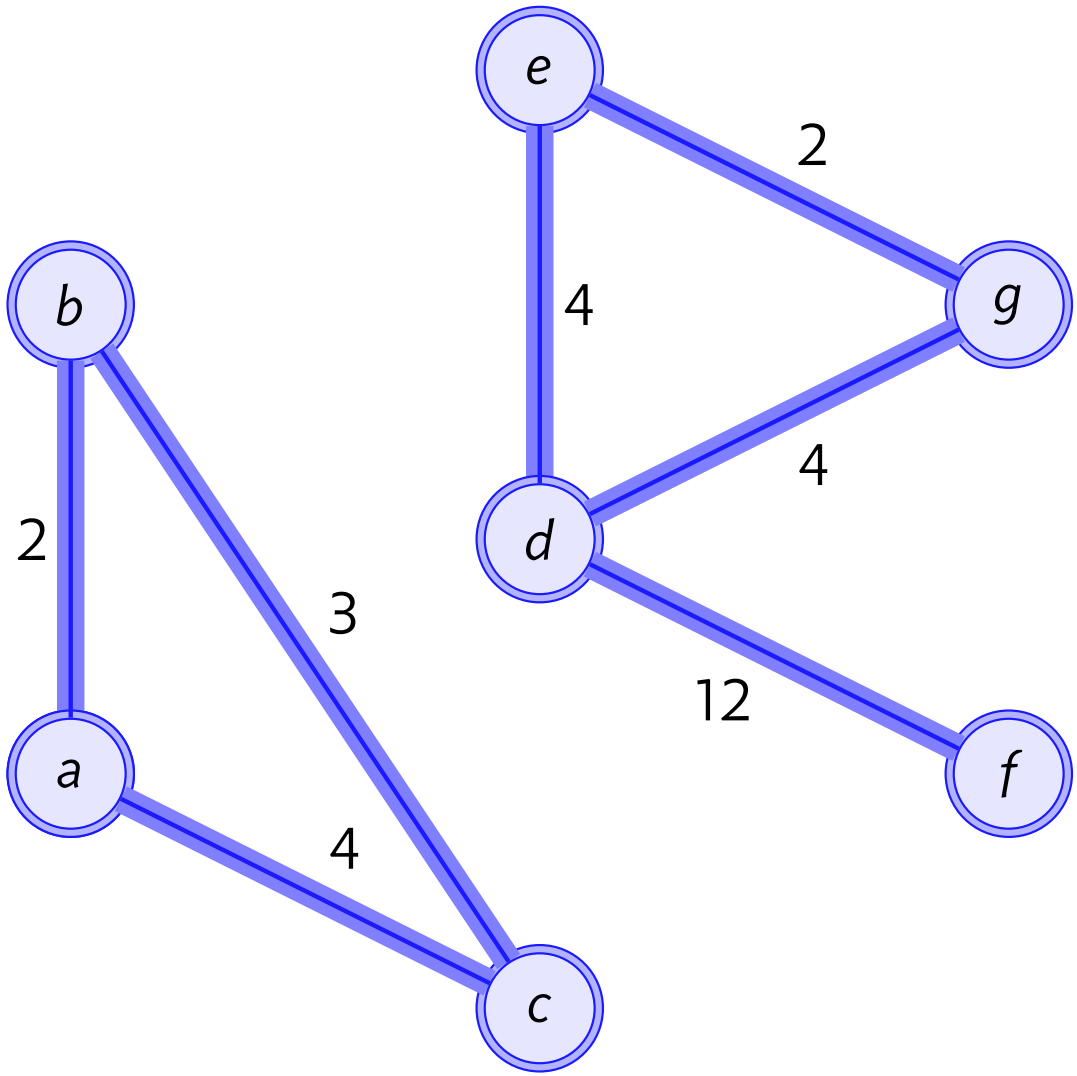
Example

- Here's our happy tree from last time.



Example

- Here's our happy tree from last time.



DFS Code

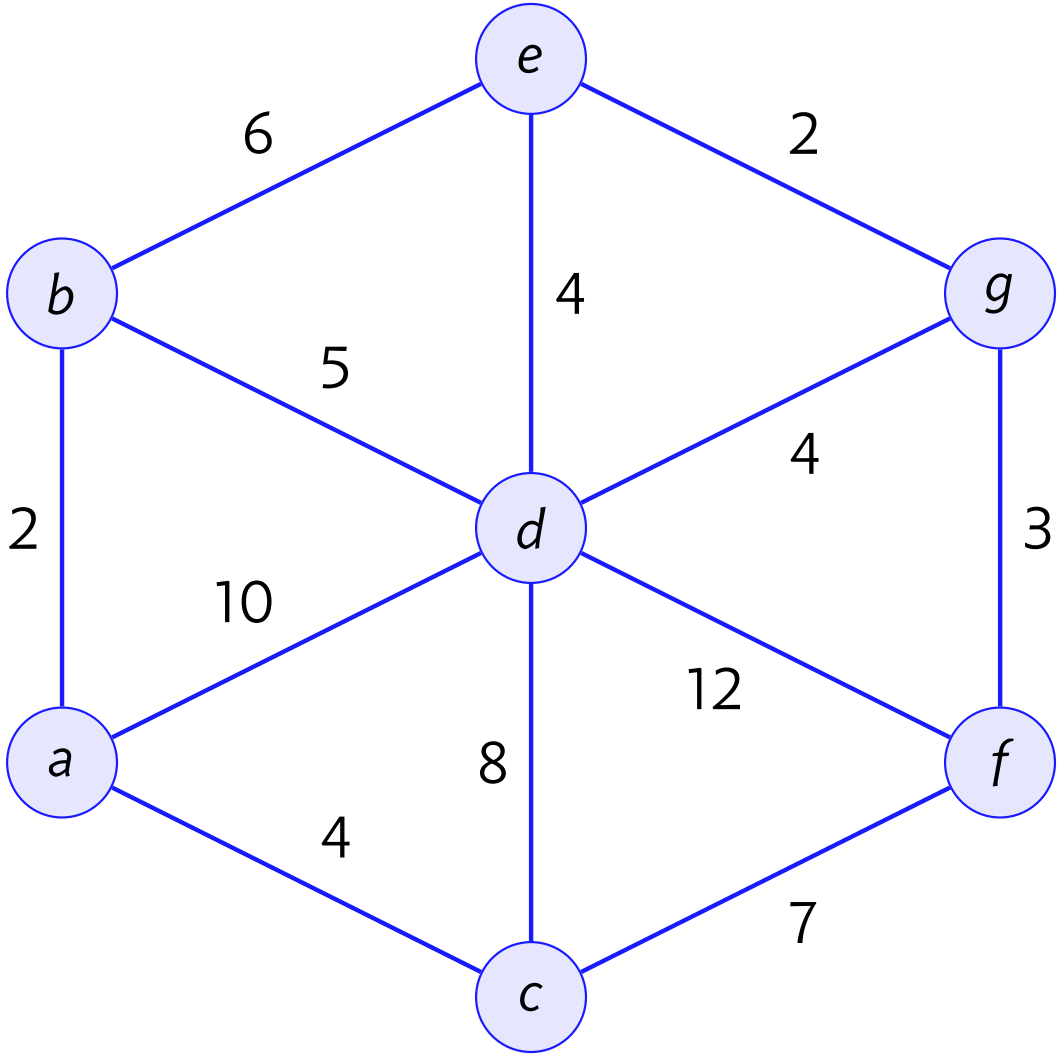
- Converted to more modern C++ from the book.

```
1  typedef pair<int, int> ii;
2  typedef vector<ii> vii; // edge is (neighbor, weight) pair
3  typedef vector<int> vi;
4
5  vi dfs_num;
6
7  void dfs(int u) {
8    // we mark the vertex as visited
9    dfs_num[u] = VISITED; // == 1,  UNVISITED == -1
10   for (auto v = AdjList[u].begin();
11        v != AdjList[u].edu(); ++v) {
12     if (dfs_num[v->first] == UNVISITED)
13       dfs(v->first);
14 } }
```

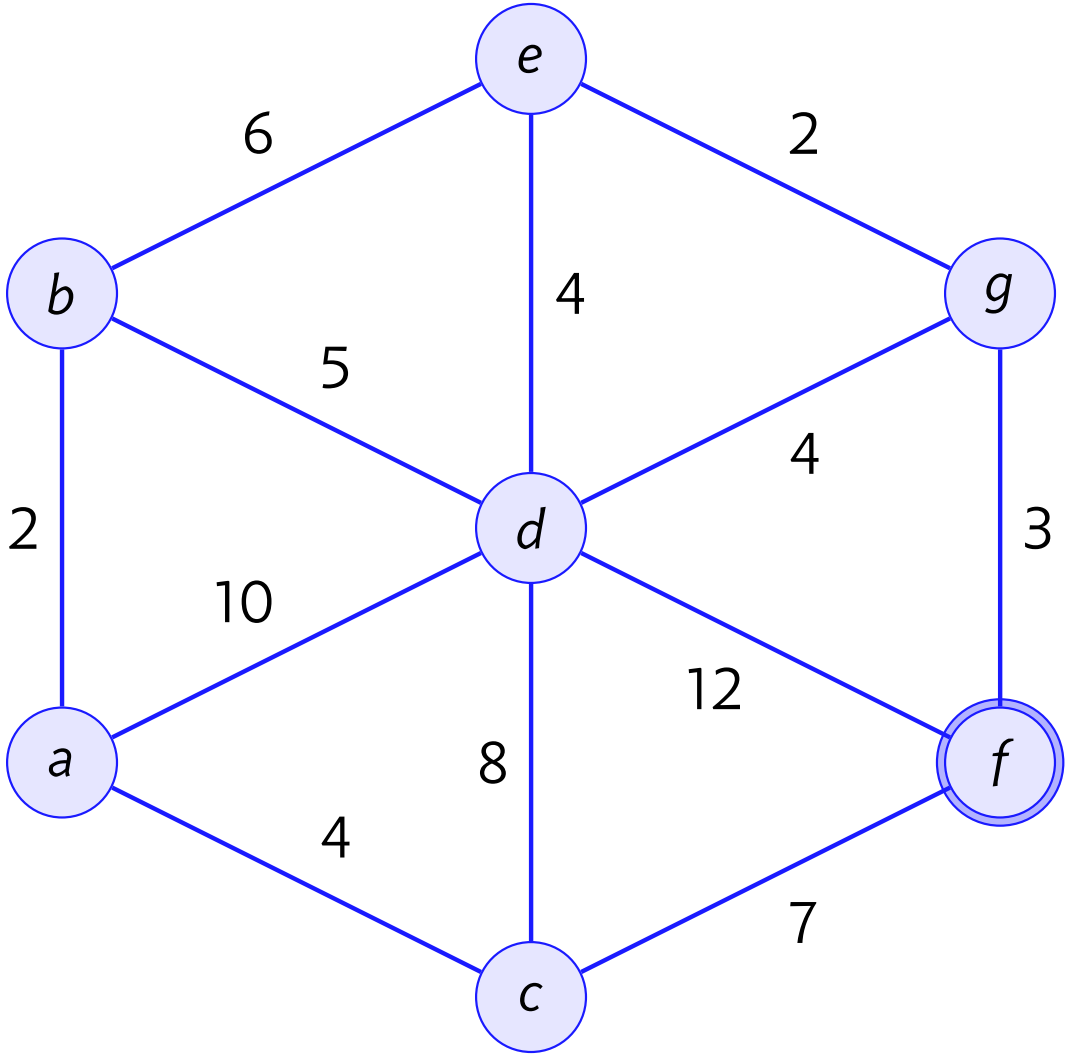
BFS Basics

- Step 1: Mark self as visited
- Step 2: Enqueue all unvisited children
- Step 3: Dequeue next child and visit
- Step 4: ???
- Step 5: Profit!

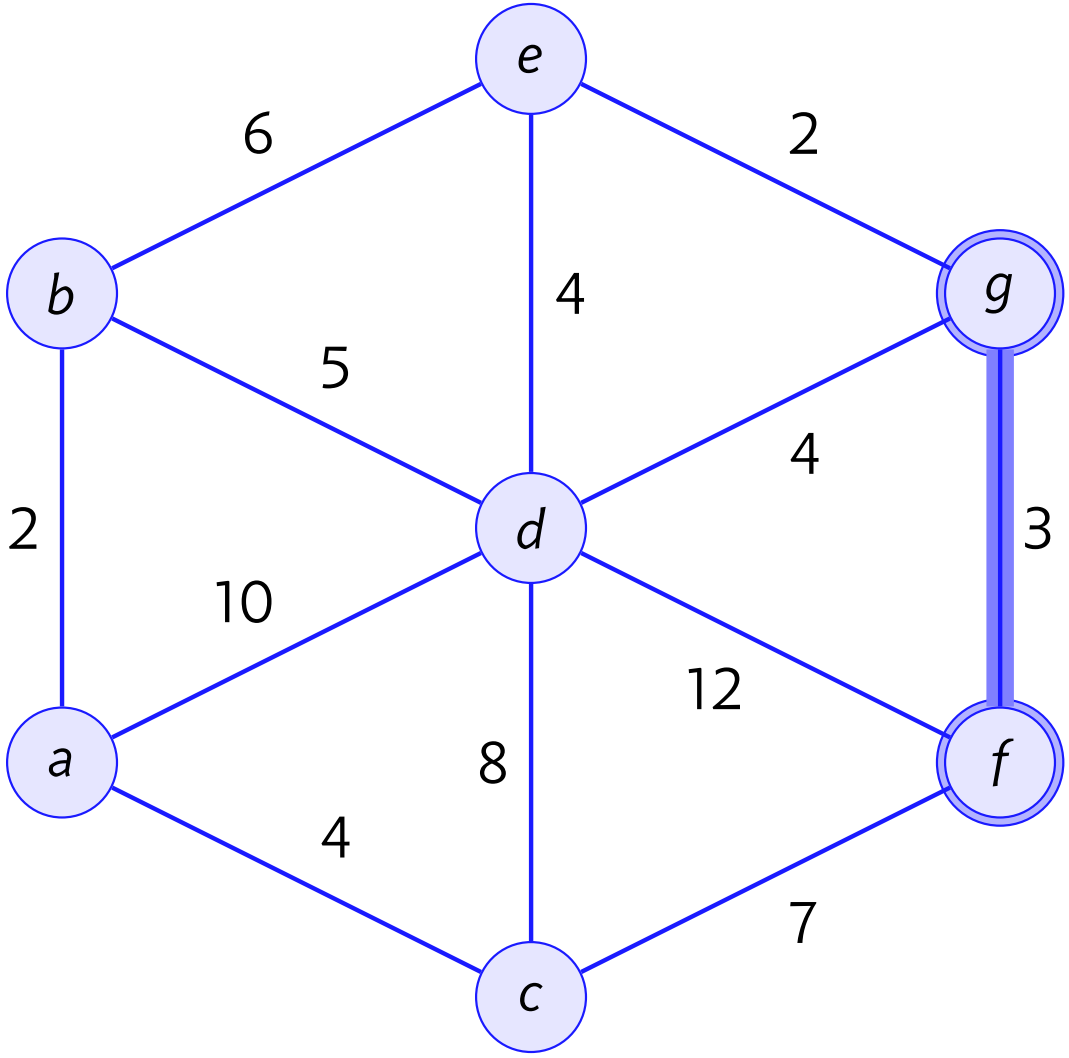
Example



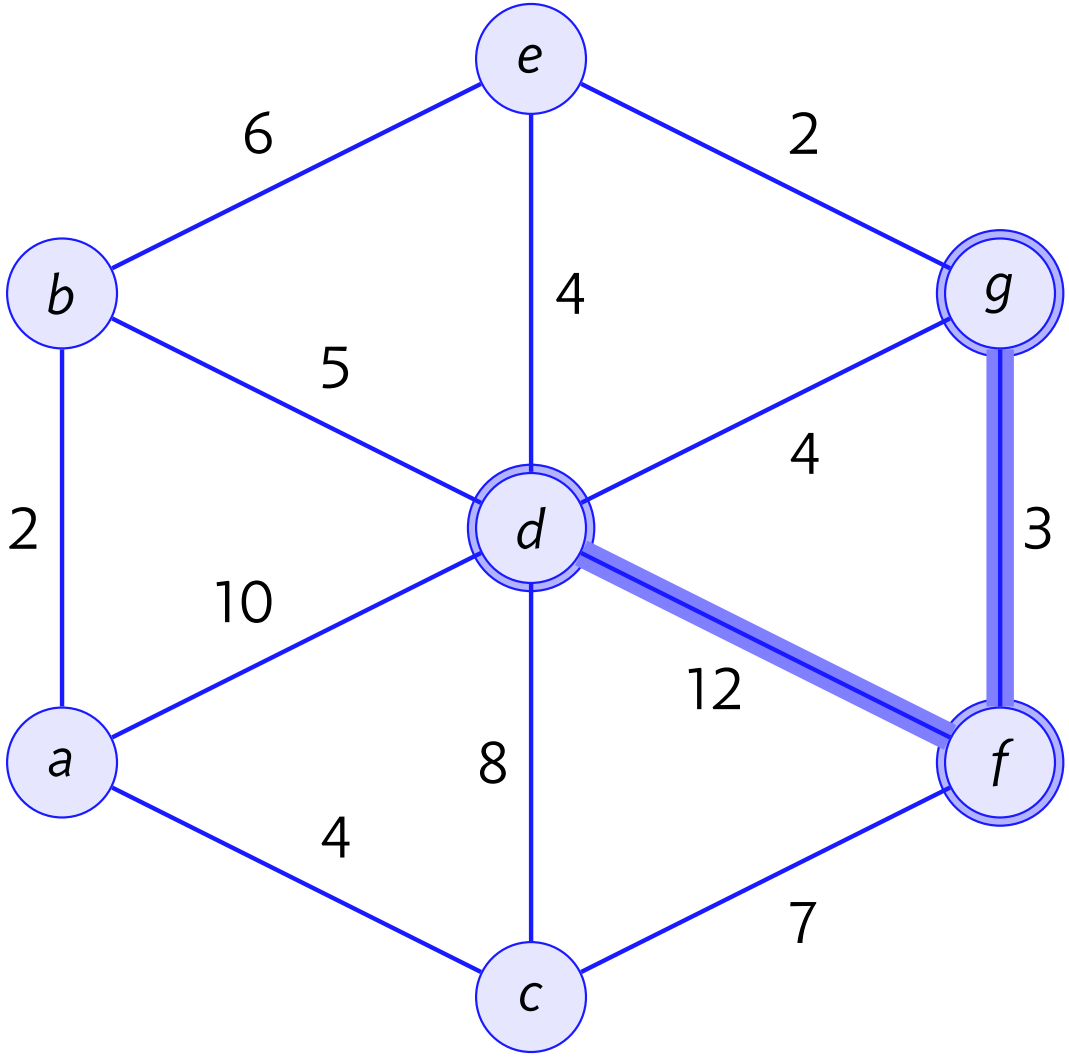
Example



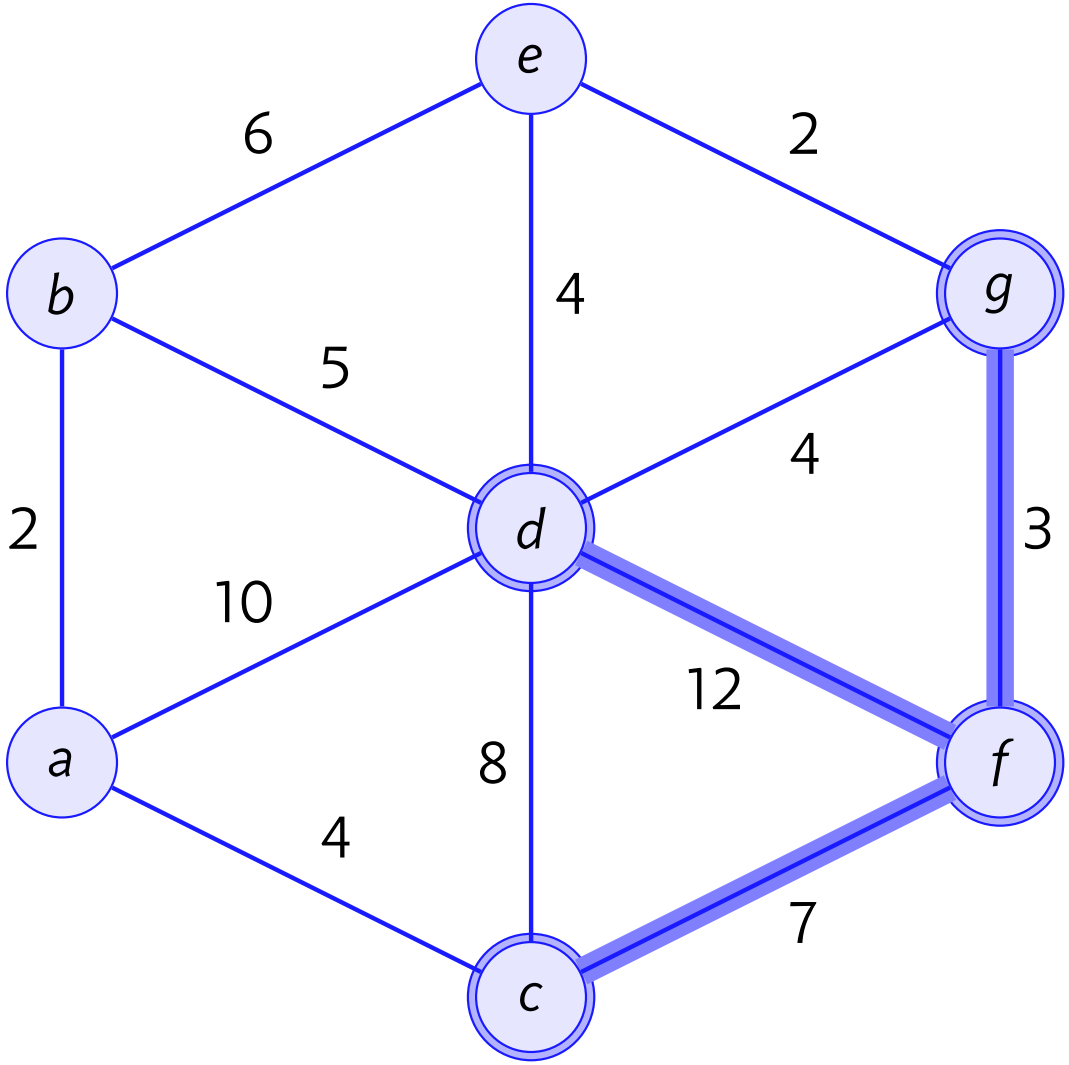
Example



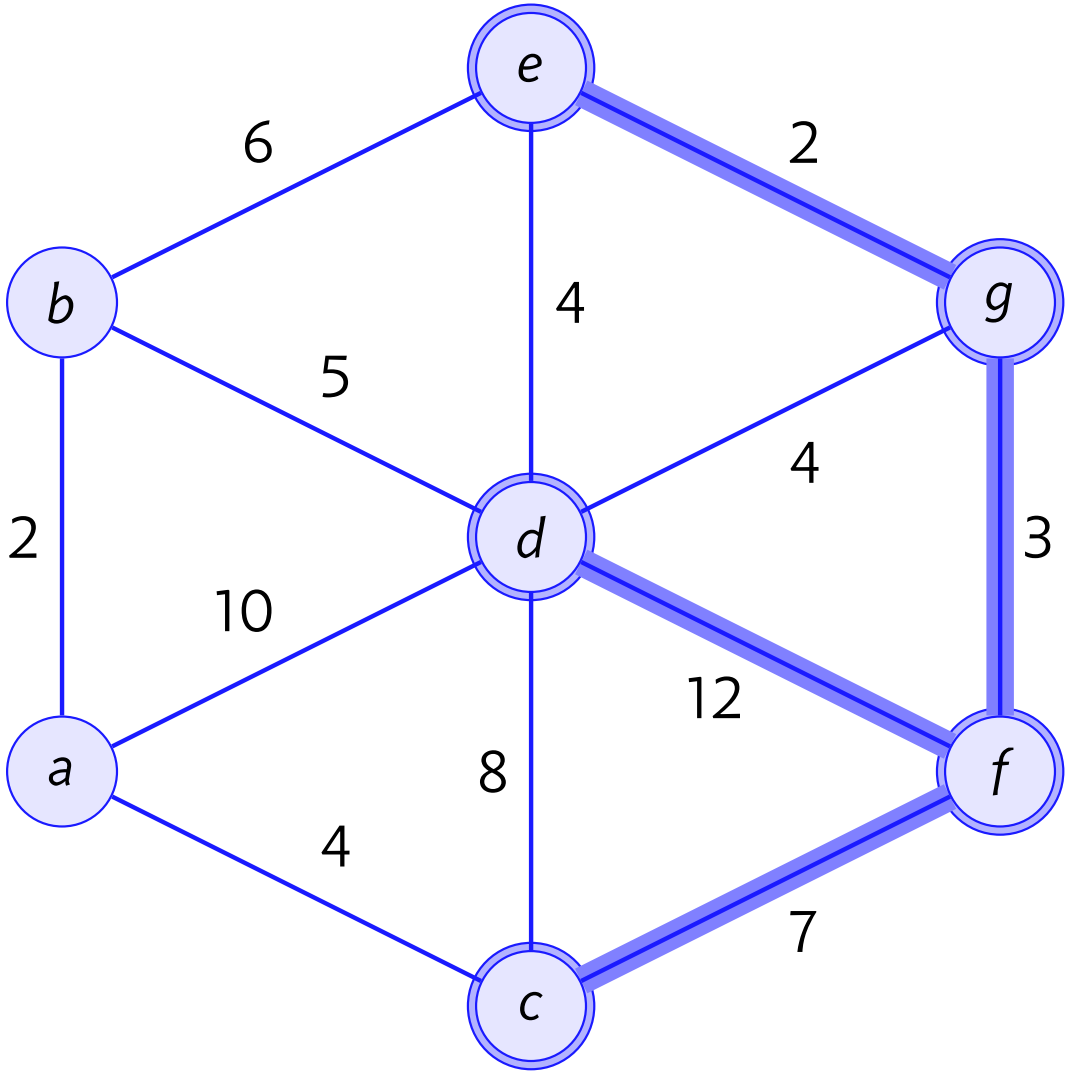
Example



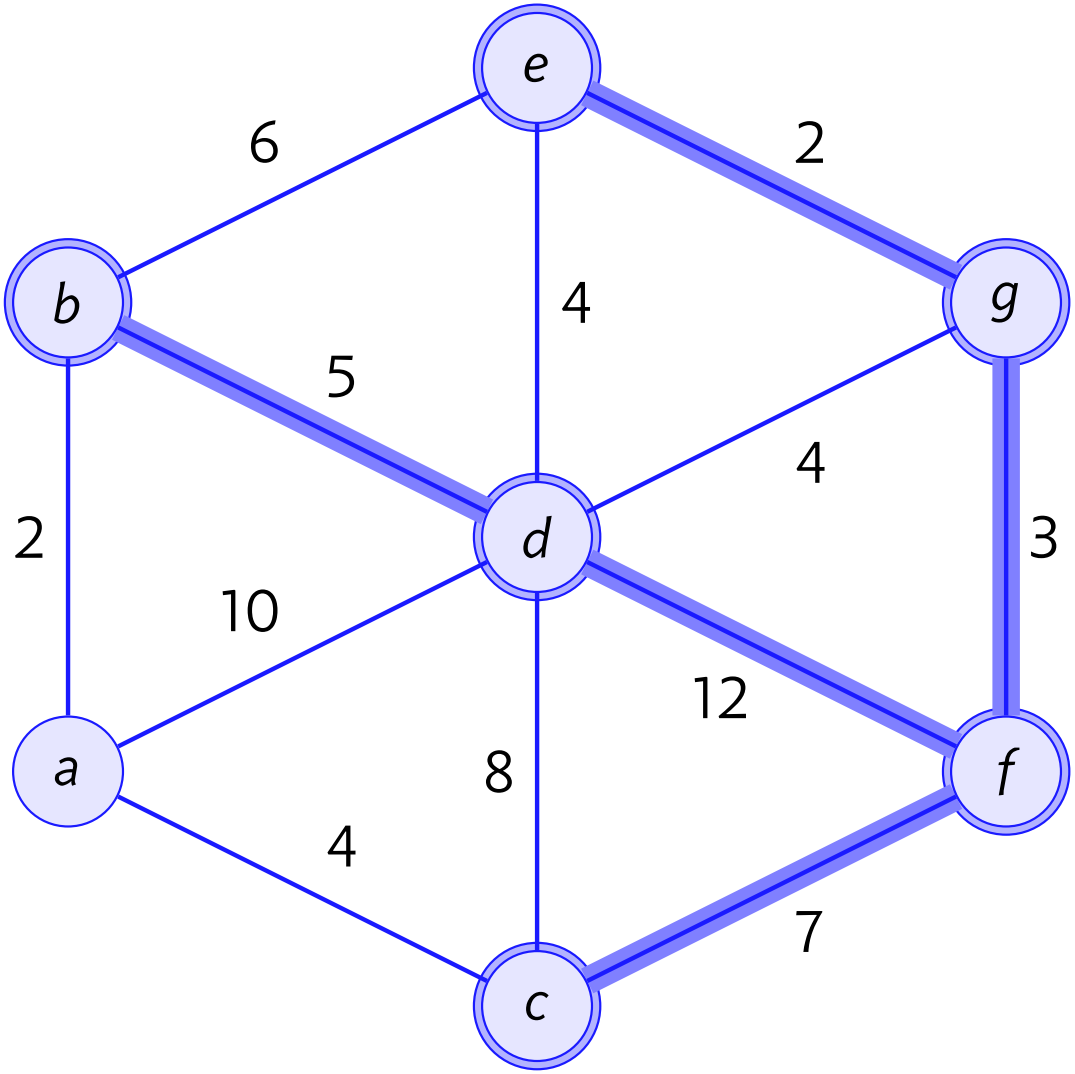
Example



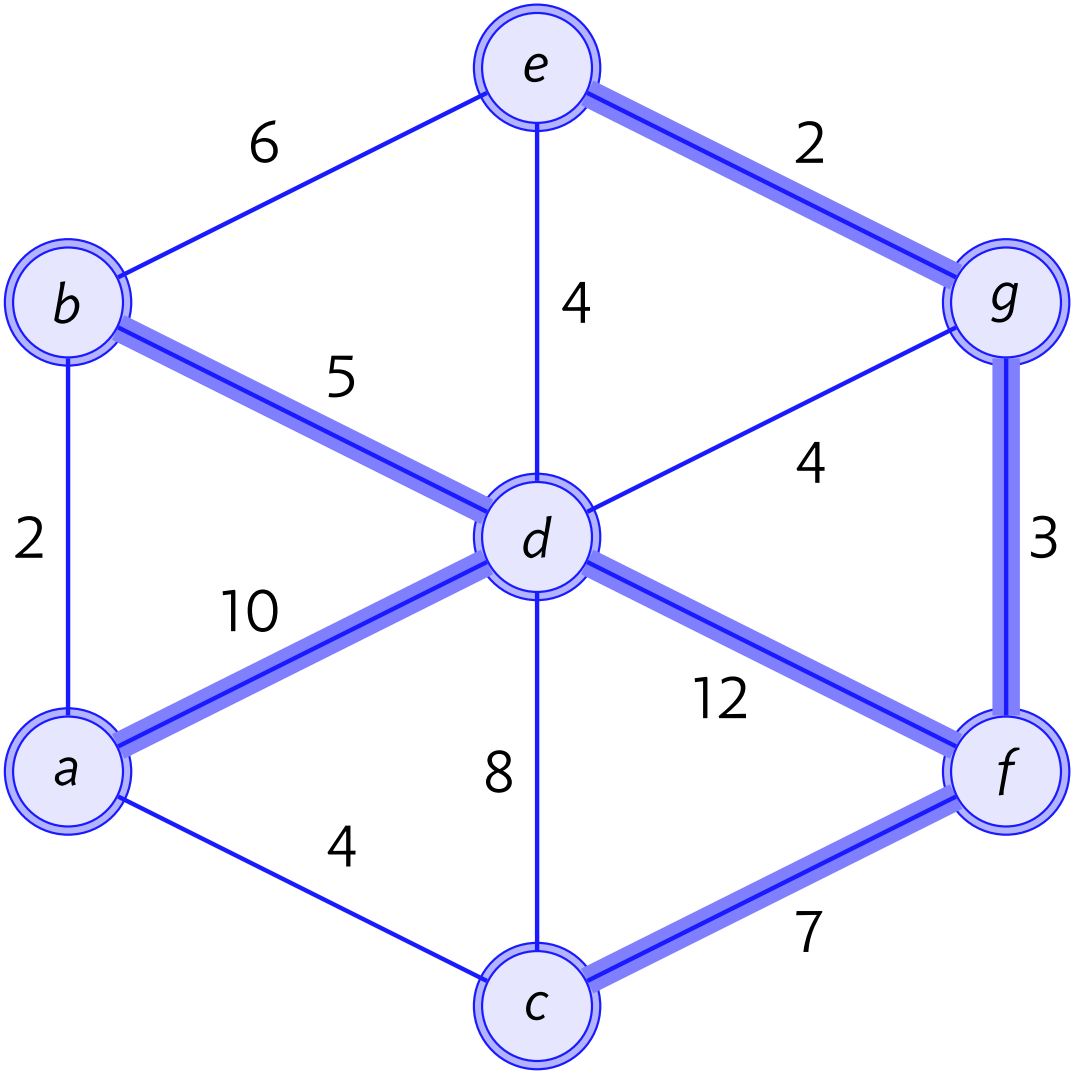
Example



Example



Example



BFS Code

```
1  vi d(V, INF); d[s] = 0;  // initialize source distance
2  queue<int> q; q.push(s); // start from source
3  while (!q.empty()) {
4    int u = q.front(); q.pop();
5    for (auto v = AdjList[u].begin();
6         v !+ AdjList[u].end();
7         ++j) {
8      if (d[v->first] == INF) {
9        d[v->first] = d[u] + 1;
10       q.push(v->first);
11 } } }
```

Connected Components

- You can use DFS to determine the number of connected components.
- Loop through the vertices; if you encounter an unvisited one, increment the CC count.

```
1  numCC = 0;
2  dfs_num.assign(V, UNVISITED);
3  for (int i = 0; i < V; i++)
4     if (dfs_num[i] == UNVISITED) {
5        printf("CC %d:", ++numCC);
6        dfs(i);
7        printf("\n");
8      }
```

Flood Fill

- This is a special case of DFS.

```
1  int dr[] = {1,1,0,-1,-1,-1, 0, 1};
2  int dc[] = {0,1,1, 1, 0,-1,-1,-1};
3
4  int floodfill(int r, int c, char c1, char c2) {
5    if (r < 0 || r >= R || c < 0 || c >= C) return 0;
6    if (grid[r][c] != c1) return 0;
7    int ans = 1;
8    grid[r][c] = c2;
9    for (int d = 0; d < 8; d++)
10       ans += floodfill(r + dr[d], c + dc[d], c1, c2);
11   return ans;
12 }
```