Grafos

1. Bfs

1.1. Tamanho do maior caminho entre dois nós em uma árvore sem peso e com arestas não direcionadas

1. Rode um bfs partindo de qualquer nó. O ultimo nó descoberto é então chamado de t.
2. Rode um bfs partindo de t. A distância entre t e o ultimo nó descoberto pelo segundo bfs é o maior caminho dentro da árvore/grafo.

```
int bfs1(int idx)
{
        queue<int> myq;
        myq.push(idx);
        int node = idx;
        visited[idx] = true;
        while (not myq.empty()) {
                node = myq.front(); myq.pop();
                visited[node] = true;
                for (auto it : graph[node]) {
                        if (not visited[it] ) {
                                visited[it] = true;
                                myq.push(it);
                        }
                }
        }

        return node;
}
```

```
int bfs2(int t)
{
        queue<ii> myq;
        myq.push(ii(0, t));
        memset(visited, 0, sizeof visited);
        visited[t] = true;
        int val = 0;
        while (not myq.empty()) {
                val = myq.front().first;
                int node = myq.front().second;
                myq.pop();

                visited[node] = true;
                for (auto it : graph[node]) {
                        if (not visited[it]) {
                                myq.push(ii(val+1, it));
                        }
                }
        }
        return val;
}

int main()
  cout << bfs2( bfs1(0) ) << "\n";
```

2. Dfs

2.1. Detectar ciclo em um grafo direcionado

Tome G(u,v) como um Grafo(G) composto de u(arestas) e v(vértices).

- Marque todas as arestas como não visitadas e não presentes na pilha.

- Visite todas as arestas não visitadas com a função `dfs(idx)`. Se das chamadas para a função retornar verdadeiro, significa que existe pelo menos um ciclo
- A função `dfs(idx)` pode ser definida como:

```
bool dfs(int idx)
{
  if (not visited[idx]) {
    visited[idx] = 1;
    instack[idx] = 1;
    bool ans = 0;
    for (auto it : G[idx]) {
      if (instack[idx]) {
        return true;
      } else {
        ans = max(ans, dfs(it));
      }
    }
    instack[idx] = false;
    return ans;
  }
  return false;
}
```