Auxílio ByteBusters - Grafos

Busca em Largura (BFS)

Explora o grafo nível por nível utilizando uma fila (FIFO). Ideal para encontrar o menor caminho em grafos não ponderados.

Exemplo em Python:

```
def bfs(grafo, inicio):
    visitado = set()
    fila = [inicio]
    while fila:
        atual = fila.pop(0)
        if atual not in visitado:
            print(atual, end=' ')
            visitado.add(atual)
            fila.extend(grafo.get(atual, []))
```

Este código percorre os vértices conectados a partir do vértice inicial, visitando cada um apenas uma vez.

Busca em Profundidade (DFS)

Explora profundamente antes de voltar, utilizando recursão. Ideal para detecção de ciclos e componentes conexos.

Exemplo em Python:

```
def dfs(grafo, v, visitado):
    if v not in visitado:
```

```
    print(v, end=' ')
    visitado.add(v)
    for viz in grafo.get(v, []):
        dfs(grafo, viz, visitado)
```

Este código visita os vértices de maneira recursiva, indo o mais fundo possível antes de voltar.

Algoritmo de Dijkstra

Encontra o menor caminho ponderado em grafos, funcionando somente com pesos positivos. Utiliza uma fila de prioridade.

Exemplo em Python:

```
import heapq

def dijkstra(grafo, inicio):
    dist = {v: float('inf') for v in grafo}
    dist[inicio] = 0
    fila = [(0, inicio)]
    while fila:
        custo, vertice = heapq.heappop(fila)
        if custo > dist[vertice]:
            continue
        for viz, peso in grafo[vertice]:
            nova_dist = custo + peso
            if nova_dist < dist[viz]:
                dist[viz] = nova_dist
                heapq.heappush(fila, (nova_dist, viz))
```

```
    return dist
```

Este código calcula o menor custo de cada vértice a partir do vértice inicial, considerando os pesos das arestas.