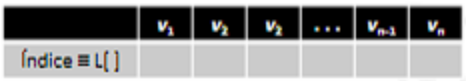
8 - Busca de largura

Introdução:

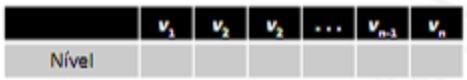
Deve-se escolher o vértice menos marcado na busca e continuar executando as buscas pelos vértices vizinhos a ele até finalizar.

Durante a busca diversos atributos são definidos para os vértices:

- Cada vértice tem um valor de tempo associado a ele, apenas o tempo de descoberta, visitado pela primeira vez.



- Necessita de um contador global que é incrementado cada vez que um vértice seja descoberto.
- Quando o vértice V é descoberto através do vértice W, o predecessor do vértice W na busca passa a ser o seu pai.
- Cada vértice tem um valor indicando a distância (menor caminho) entre ele e a raiz da busca.



- Define o vértice raiz da busca com o nível 1, com isso os outros vértices filhos do pai passam a ser nível do pai + 1.

Busca em largura em GRAFO NÃO DIRECIONADO:

As arestas podem ser rotuladas das seguintes formas:

- Aresta de árvore: quando a aresta {v, w} é usada para visitar pela primeira vez o W.

- Aresta de tio: nível dos vértices são diferentes e os pais são diferentes.
- Aresta de irmão: nível de vértice sejam iguais e os pais sejam iguais.
- Aresta de primo: nível de vértice sejam iguais e os pais sejam diferentes.

Algoritmo:

```
Inicialização / Chamada inicial
1.  t ← 0;   Fila ← ∅;                                   // Inicializar tempo e fila
2.  para todo vértice v ∈ V(G) faça
    a.  L[v] ← 0;                                        // Inicializar índice
    b.  nível[v] ← 0;                                    // Inicializar nível
    c.  pai[v] ← null;                                   // Inicializar predecessor ou pai
3.  enquanto existir algum vértice v tal que L[v] = 0 efetuar
    a.  t ← t + 1;  L[v] ← t;                            // Atualizar nível da raiz
    b.  Fila.Insere(v);                                  // Inserir a raiz na fila
    c.  Executar Busca_Largura();                        // Executar busca para raiz v
```

```
Busca_Largura()
  enquanto not Fila.Vazia() efetuar
    1.  v ← Fila.Remove();                               // Remover 1º elemento da fila
    2.  para todo vértice w ∈ Γ(v) faça                  // Para toda a vizinhança de v
        a.  se L[w] = 0 então                            // Se w é visitado pela 1ª vez
                Visitar aresta de árvore ou pai {v, w};  pai[w] ← v;
                nível[w] ← nível[v] + 1; t ← t + 1;  L[w] ← t;  Fila.Insere(w);
        b.  senão se nível[w] = nível[v] + 1 então
                Visitar aresta tio {v, w};
        c.  senão se nível[w] = nível[v] e pai[v] = pai[w] e L[w] > L[v] então
                Visitar aresta irmão {v, w};
        d.  senão se nível[w] = nível[v] e pai[v] ≠ pai[w] e L[w] > L[v] então
                Visitar aresta primo {v, w};
```

A condição 2.c (L[W] > L[V]) e a condição 2.d (L[W] > L[V]) é necessária para que a aresta seja explorada uma única vez.

O custo da busca é O(m + n), em que n = |V(G)| e m = |E(G)|.

Exemplo:

	a	b	c	d	e
L	1	2	3	4	5
nível	0	1	1	2	2
pai	∅	a	a	b	c

A representação formada por todos os vértices e apenas as arestas de árvore ou pai formam uma árvore de largura. Caso o grafo seja desconexo, a busca irá produzir várias árvores de largura.