Algoritmos - Cormen

OBS: Essas anotações, como o próprio título já diz, se baseiam bastante no livro Algoritmos, Teoria e Prática.

Antes de discutir as representações, é fundamental entender o que é um grafo. Um **grafo G é uma estrutura matemática fundamental definida como um par (V, E)**.

• **V** representa o **conjunto finito de vértices** (também conhecidos como nós), que são os objetos individuais no grafo. O número de vértices em um grafo é denotado por **|V|**.

• **E** representa o **conjunto de arestas**, que são as conexões ou relações entre os vértices. O número de arestas em um grafo é denotado por **|E|**.

Tipos de grafos

• **Grafos dirigidos**: As arestas são pares ordenados (u, v), indicando uma conexão com uma direção específica, de u para v. Laços (arestas de um vértice para si mesmo) são possíveis.

• **Grafos não dirigidos**: As arestas são pares não ordenados {u, v}, significando que a conexão entre u e v não tem direção. Por convenção, também usamos a notação (u, v) para arestas não dirigidas, e (u, v) e (v, u) são consideradas a mesma aresta. Nesses grafos, **laços são proibidos**, e cada aresta conecta dois vértices distintos.

Grafo esparso e Grafo denso

• **Grafos esparsos** são aqueles em que o **número de arestas (|E|) é muito menor que o número máximo possível de arestas ($|V|^2$)**. Para esses grafos, a representação por listas de adjacências é geralmente preferida. Veremos o que são listas de adjacências em breve.

• **Grafos densos** são aqueles em que o **número de arestas (|E|) está próximo do número máximo possível de arestas ($|V|^2$)**. Para esses grafos, a representação por matrizes de adjacências pode ser mais adequada, especialmente quando se precisa verificar rapidamente a existência de uma aresta entre dois vértices dados. Veremos o que são matrizes de adjacências em breve.

O valor **$|V|^2$** é o **número máximo possível de arestas que um grafo com |V| vértices pode ter**. Esse valor serve como um ponto de referência para medir a "densidade" ou quão "cheio" um grafo é em termos de suas conexões.

• **Para entender em um grafo direcionado**:

◦ Se existem **|V| vértices**, e cada vértice pode ter uma aresta apontando para **qualquer outro dos |V| vértices** (incluindo ele mesmo, se laços forem permitidos), então o número total máximo de arestas possíveis é **|V| multiplicado por |V|**, o que resulta em **$|V|^2$** arestas.

◦ Mesmo se laços não forem permitidos (o que significa que um vértice não pode ter uma aresta para si mesmo), cada vértice ainda pode se conectar a |V|-1 outros vértices. O número total de arestas seria |V| * (|V|-1), que ainda é **assintoticamente da ordem de $|V|^2$**.

• **Para entender em um grafo não dirigido**:

◦ Nesse tipo de grafo, **laços são proibidos**, e uma aresta simplesmente conecta um par de vértices distintos.

◦ O número de maneiras de escolher dois vértices distintos de um conjunto de |V| vértices para formar uma aresta é dado pela fórmula de combinações: |V| * (|V|-1) / 2.

◦ Esta expressão também é **assintoticamente da ordem de $|V|^2$**.

Portanto, **$|V|^2$** fornece uma medida do potencial máximo de conexões de um grafo. Ao comparar o número real de arestas (|E|) com este valor de referência ($|V|^2$), podemos classificar um grafo como esparso ou denso, o que influencia a escolha da representação mais eficiente e a análise do desempenho dos algoritmos.

Lista de adjacência e Matriz de adjacência

Podemos escolher entre dois modos padrões para representar um grafo G = (V, E): como uma coleção de **listas de adjacências** ou como uma **matriz de adjacências**. Qualquer desses modos se aplica a grafos dirigidos e não dirigidos.

Geralmente, **nosso método preferido são as listas de adjacência**. Elas **se destacam em situações em que estamos lidando com grafos esparsos**. No entanto, **para grafos densos**, a **matriz de adjacência pode ser preferível**, ou quando **precisamos saber rapidamente se há uma aresta conectando dois vértices** dados.

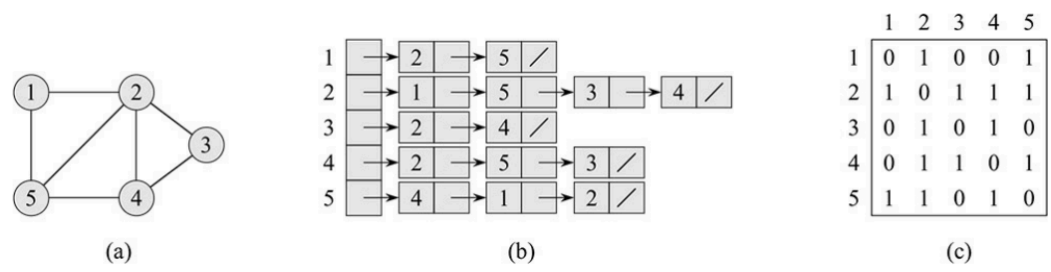


Figura 22.1 Duas representações de um grafo não dirigido. (**a**) Um grafo não dirigido *G* com cinco vértices e sete arestas. (**b**) Uma representação de *G* por lista de adjacências. (**c**) A representação de *G* por matriz de adjacências.

Figuras do livro Algoritmos - Teoria e Prática de Cormen.

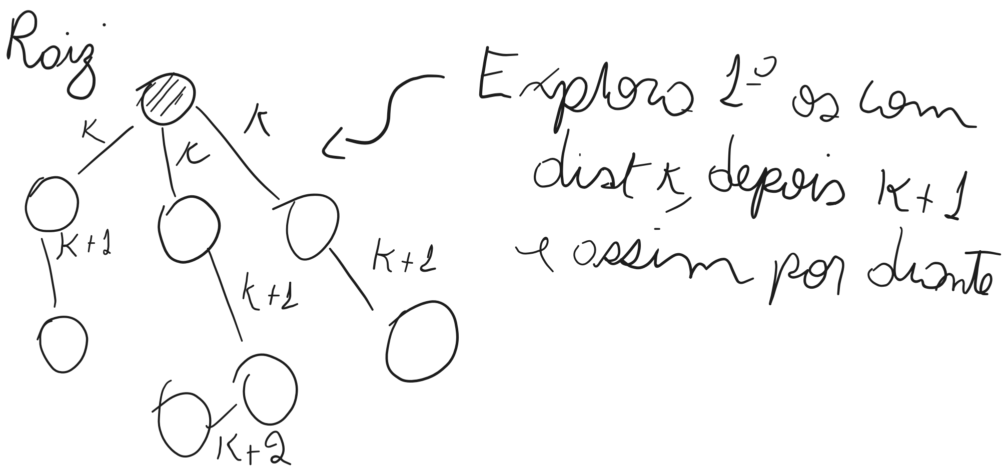
Temos também as matrizes de incidência, mas não irei trabalhar com elas aqui.

Busca em Largura

Dado um grafo G = (V, E) e um vértice fonte (**raiz**) s, a busca em largura explora sistematicamente as arestas de G para “descobrir” cada vértice que pode ser alcançado a partir de s. O algoritmo calcula a distância (menor número de arestas) de s até cada vértice que pode ser alcançado. Produz também uma “árvore de busca em largura” com raiz s que contém todos os vértices que podem ser alcançados.

Para qualquer vértice v que pode ser alcançado de s, o caminho simples na árvore de busca em largura de s até v corresponde a um caminho que contém o menor número de aresta de s a v. O algoritmo funciona em grafos dirigidos, bem como em grafos não dirigidos.

A busca em largura tem esse nome porque expande a fronteira entre vértices descobertos e não descobertos uniformemente ao longo da extensão da fronteira. Isto é, o algoritmo descobre todos os vértices à distância k de s, antes de descobrir quaisquer vértices à distância k + 1. Expansão é uniforme.



Para controlar o progresso, a busca em largura pinta cada vértice de **branco, cinzento ou preto**. No início, todos os vértices são brancos, e mais tarde eles podem se tornar **cinzentos** e **depois pretos**. Um vértice é descoberto na primeira vez em que é encontrado durante a busca, e nesse momento ele se torna não branco. Portanto, vértices cinzentos e pretos são vértices descobertos.

Se (u, v) ∈ E e o vértice u é preto, então o vértice v é cinzento ou preto; isto é, **todos os vértices adjacentes a vértices pretos foram descobertos**. **Vértices cinzentos podem ter alguns vértices adjacentes brancos**; eles **representam a fronteira entre vértices descobertos e não descobertos**.

A busca em largura **constrói uma árvore em largura**, que **contém inicialmente apenas sua raiz**, que é o vértice de fonte s. **Sempre que a busca descobre um vértice branco v no curso da varredura da lista de adjacências de um vértice u já descoberto, o vértice**

v e a aresta (u, v) são acrescentados à árvore. Dizemos que **u é o predecessor ou pai de v** na árvore de busca em largura.

Um vértice é descoberto apenas uma vez e, portanto, tem no máximo um pai.

Relações de ancestral e descendente na árvore de busca em largura são definidas em relação à raiz s da maneira usual: **se u está em um caminho simples na árvore que vai da raiz s até o vértice v, então u é um ancestral de v,e v é um descendente de u**.

```
BFS(G, s)
    for cada vértice u ∈ V[G] − {s}
        u.cor = BRANCO
        u.d = ∞
        u.π = NIL
    s.cor = CINZENTO
    s.d = 0
    s.π = NIL
    Q = ∅
    ENQUEUE(Q, s)
    while Q ≠ ∅
        u = DEQUEUE(Q)
        for cada v = Adj[u]
            if v.cor == BRANCO
                v.cor == CINZENTO
                v.d = u.d + 1
                v.π = u
                ENQUEUE(Q, v)
        u.cor = PRETO
```

Explicações sobre o pseudocódigo acima: Armazenamos a cor de cada vértice u ∈ V no atributo u.cor e o predecessor de u no atributo u.p. Se u não tem nenhum predecessor (por exemplo, se u = s ou se u não foi descoberto), então u.p = NIL. O atributo u.d mantém a distância da fonte s ao vértice u em número de arestas. O algoritmo também utiliza uma fila Q para gerenciar o conjunto de vértices cinzentos.