Bab 4: Representasi Graf untuk Perhitungan

Pada bab-bab sebelumnya, graf didefinisikan secara matematis sebagai

$$G = (V, E),$$

dengan V himpunan simpul dan E himpunan sisi. Definisi ini sangat bersih untuk pembuktian. Namun ketika kita ingin menghitung derajat, mencari tetangga, menjalankan BFS, menghitung jarak terpendek, atau membangun model statistik jaringan, komputer tidak langsung “memahami” pasangan himpunan abstrak G=(V,E). Kita harus menerjemahkannya menjadi struktur data.

Bab ini membahas dua representasi graf paling dasar:

1. matriks ketetanggaan, yaitu representasi berbentuk tabel n× n;

2. daftar ketetanggaan, yaitu representasi berbentuk daftar tetangga untuk setiap simpul.

Keduanya adalah jembatan antara definisi matematika dan algoritma. Dalam teori graf dan ilmu komputer, pemilihan representasi graf memengaruhi biaya memori dan waktu komputasi secara langsung (Cormen et al., 2009). Dalam statistik jaringan, pilihan ini juga memengaruhi apakah analisis dapat dijalankan pada data besar, terutama ketika graf memiliki banyak simpul tetapi relatif sedikit sisi, suatu pola yang sering muncul pada jaringan empiris meskipun tidak berlaku universal untuk semua data jaringan (Newman, 2010).

Kita akan menggunakan notasi standar:

$$n = |V|, \qquad m = |E|.$$

Di sini n adalah order graf, yaitu banyak simpul, dan m adalah size graf, yaitu banyak sisi.

4.1 Dari graf abstrak ke struktur komputasi

Misalkan kita mempunyai graf tak berarah sederhana

$$G = (V, E),$$

dengan

$$V = \{A, B, C, D, E\}$$

dan

$$E = \{\{A, B\}, \{A, C\}, \{B, D\}, \{C, D\}, \{D, E\}\}.$$

Secara matematis, penulisan ini sudah cukup. Kita tahu bahwa A bertetangga dengan B dan C, bahwa D bertetangga dengan B,C,E, dan bahwa graf memiliki

$$n = 5, \qquad m = 5.$$

Namun komputer memerlukan bentuk yang lebih eksplisit. Ada dua pertanyaan dasar yang hampir selalu muncul dalam algoritma graf.

Pertama, apakah dua simpul tertentu bertetangga?

Contoh:

$$\text{apakah } A \text{ bertetangga dengan } D?$$

Kedua, siapa saja tetangga dari suatu simpul?

Contoh:

$$\text{siapa saja tetangga dari } D?$$

Matriks ketetanggaan sangat baik untuk pertanyaan pertama. Daftar ketetanggaan sangat baik untuk pertanyaan kedua, terutama pada graf jarang. Istilah graf jarang berarti graf yang jumlah sisinya jauh lebih kecil daripada jumlah sisi maksimum yang mungkin. Untuk graf tak berarah sederhana dengan n simpul, jumlah sisi maksimum adalah

$$\binom{n}{2} = \frac{n(n-1)}{2}.$$

Jadi graf dengan jutaan simpul tetapi hanya beberapa juta sisi biasanya dianggap jarang, karena jumlah sisi maksimum teoretisnya berorde n^2. Sebaliknya, graf padat adalah graf yang jumlah sisinya mendekati jumlah sisi maksimum.

Istilah “berorde” di sini mengarah pada ukuran pertumbuhan. Dalam analisis algoritma, kita sering menulis $O(n^2)$, $O(n+m)$, atau $O(\deg(v))$. Notasi $O(\cdot)$, disebut notasi Big-O, menggambarkan batas atas pertumbuhan biaya komputasi atau memori ketika ukuran input membesar. Misalnya, mengatakan bahwa suatu representasi memerlukan memori $O(n^2)$ berarti kebutuhan memorinya tumbuh sebanding dengan jumlah pasangan simpul, bukan hanya sebanding dengan jumlah simpul atau sisi. Notasi ini adalah bahasa standar untuk membandingkan algoritma dan struktur data (Cormen et al., 2009).

4.2 Pengindeksan simpul

Sebelum membuat matriks atau daftar, kita biasanya memberi indeks numerik pada simpul.

Untuk contoh di atas, kita dapat memilih:

$$A \mapsto 1, \quad B \mapsto 2, \quad C \mapsto 3, \quad D \mapsto 4, \quad E \mapsto 5.$$

Pengindeksan ini tampak sepele, tetapi penting. Matriks dan array komputer bekerja dengan indeks seperti 1,2,…,n atau, dalam banyak bahasa pemrograman, 0,1,…,n-1. Jika data asli memakai label seperti nama orang, kode provinsi, ID pasien, atau nama gen, maka kita memerlukan peta dari label asli ke indeks numerik.

Dalam analisis statistik, langkah ini harus dilakukan hati-hati. Label simpul bukan sekadar dekorasi. Jika simpul mewakili pasien, maka kesalahan memetakan ID pasien ke indeks dapat membuat sisi menghubungkan individu yang salah. Jika simpul mewakili variabel dalam model grafis probabilistik, kesalahan indeks dapat mengubah struktur dependensi yang dimodelkan.

Secara praktis, kita sering menyimpan dua objek:

1. kamus label-ke-indeks, misalnya A↦ 1;
2. kamus indeks-ke-label, misalnya 1↦ A.

Dengan begitu, algoritma dapat bekerja pada indeks numerik, sedangkan interpretasi statistik tetap kembali pada label asli.

4.3 Matriks ketetanggaan

4.3.1 Definisi untuk graf tak berarah sederhana

Untuk graf tak berarah sederhana dengan simpul

$$V = \{v_1, v_2, \ldots, v_n\},$$

matriks ketetanggaan adalah matriks A berukuran n× n dengan entri

$$A_{ij} = \begin{cases} 1, & \text{jika } \{v_i, v_j\} \in E, \\ 0, & \text{jika } \{v_i, v_j\} \notin E. \end{cases}$$

Matriks ini disebut “ketetanggaan” karena entri A_{ij} menjawab apakah v_i dan v_j bertetangga.

Untuk contoh

$$V = \{A, B, C, D, E\}$$

dengan urutan A,B,C,D,E, kita memperoleh:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Baris pertama menyatakan tetangga A. Karena baris pertama adalah

$$(0, 1, 1, 0, 0),$$

maka A bertetangga dengan B dan C, tetapi tidak dengan A,D,E.

Untuk graf tak berarah sederhana, matriks ketetanggaan selalu simetris, yaitu

$$A_{ij} = A_{ji}$$

untuk semua i,j. Alasannya sederhana: jika v_i bertetangga dengan v_j, maka v_j juga bertetangga dengan v_i. Sifat simetri ini adalah ciri aljabar dari graf tak berarah.

Selain itu, untuk graf sederhana tanpa gelang, diagonal matriks berisi nol:

$$A_{ii} = 0.$$

Diagonal A_{ii} mewakili sisi dari v_i ke dirinya sendiri. Karena graf sederhana tidak memiliki gelang, semua entri diagonal bernilai nol.

4.3.2 Membaca derajat dari matriks ketetanggaan

Untuk graf tak berarah sederhana, derajat simpul v_i dapat dihitung dengan menjumlahkan baris ke-i:

$$\deg(v_i) = \sum_{j=1}^{n} A_{ij}.$$

Dalam contoh kita, baris untuk D adalah

$$(0, 1, 1, 0, 1).$$

Maka

$$\deg(D) = 0 + 1 + 1 + 0 + 1 = 3.$$

Ini cocok dengan definisi derajat: D bersisian dengan tiga sisi, yaitu B,D\, C,D\, dan D,E\.

Jika kita menjumlahkan semua entri matriks ketetanggaan graf tak berarah sederhana, setiap sisi dihitung dua kali: sekali pada posisi A_{ij}, sekali lagi pada posisi A_{ji}. Karena itu,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} = 2m.$$

Ini adalah bentuk matriks dari Handshaking Lemma yang telah dibuktikan pada Bab 2:

$$\sum_{v \in V} \deg(v) = 2|E|.$$

Di sini terlihat hubungan penting antara representasi komputasi dan pembuktian matematis. Matriks bukan hanya format penyimpanan; ia juga memungkinkan kita menulis ulang identitas graf dalam bahasa aljabar linear. Hubungan antara graf dan matriks akan dibahas lebih jauh pada Bab 15 dan Bab 16.

4.3.3 Keunggulan matriks ketetanggaan

Keunggulan utama matriks ketetanggaan adalah akses cepat untuk pertanyaan:

$$\text{apakah } v_i \text{ bertetangga dengan } v_j?$$

Jika matriks sudah tersedia, kita cukup memeriksa entri A_{ij}. Dalam model komputasi standar, akses ke satu entri array dua dimensi dianggap waktu konstan, yaitu O(1) (Cormen et al., 2009).

Contoh: apakah B bertetangga dengan E?

Dengan urutan A,B,C,D,E, kita memeriksa entri

$$A_{2,5}.$$

Dari matriks,

$$A_{2,5} = 0.$$

Jadi B tidak bertetangga dengan E.

Apakah C bertetangga dengan D?

Kita memeriksa

$$A_{3,4} = 1.$$

Jadi C bertetangga dengan D.

Akses cepat ini sangat berguna dalam algoritma yang sering menanyakan keberadaan sisi antara dua simpul tertentu. Misalnya, pada perhitungan motif jaringan atau pemeriksaan apakah tiga simpul membentuk segitiga, kita sering perlu memeriksa banyak pasangan simpul. Matriks ketetanggaan membuat operasi semacam itu sederhana dan cepat.

4.3.4 Biaya memori matriks ketetanggaan

Kelemahan utama matriks ketetanggaan adalah kebutuhan memorinya.

Matriks n× n memiliki

$$n^2$$

entri. Jadi kebutuhan memori dasarnya adalah O(n²), terlepas dari berapa banyak sisi yang benar-benar ada.

Jika n=5, ini tidak menjadi masalah. Kita hanya menyimpan 25 entri.

Namun jika n=1,000,000, maka matriks memiliki

$$10^{12}$$

entri. Bahkan jika setiap entri hanya memakai satu byte, kebutuhan memori mentahnya sekitar satu terabyte. Dalam praktik, representasi aktual dapat lebih besar bergantung pada tipe data dan bahasa pemrograman.

Masalahnya menjadi jelas pada graf jarang. Misalkan graf memiliki satu juta simpul dan dua juta sisi. Untuk graf tak berarah, daftar sisi aktual hanya berjumlah dua juta, tetapi matriks ketetanggaan tetap harus menyediakan ruang untuk 10^{12} kemungkinan pasangan simpul. Sebagian besar entri bernilai nol.

Dalam statistik jaringan, situasi seperti ini sering muncul. Misalnya, jaringan pertemanan, komunikasi, sitasi, atau transaksi biasanya tidak menghubungkan setiap pasangan unit. Banyak pasangan tidak memiliki sisi. Untuk data seperti itu, menyimpan semua nol dalam matriks padat biasanya tidak efisien. Literatur jaringan modern sering menekankan bahwa banyak jaringan empiris berskala besar bersifat jarang dibandingkan jumlah pasangan simpul yang mungkin, meskipun tingkat kejarangan bergantung pada domain dan cara pengukuran jaringan (Newman, 2010).

4.4 Daftar ketetanggaan

4.4.1 Definisi dasar

Daftar ketetanggaan menyimpan, untuk setiap simpul, daftar simpul-simpul yang bertetangga dengannya.

Untuk contoh kita, daftar ketetanggaannya adalah:

```
A: B, C
B: A, D
C: A, D
D: B, C, E
E: D
```

Representasi ini sangat dekat dengan cara kita berbicara tentang graf: “tetangga D adalah B,C,E.”

Secara matematis, untuk setiap simpul v, daftar ketetanggaan menyimpan himpunan atau daftar

$$N(v) = \{u \in V : \{u, v\} \in E\}.$$

Himpunan N(v) disebut lingkungan atau neighborhood dari v. Untuk contoh di atas,

$$N(D) = \{B, C, E\}.$$

Karena itu,

$$\deg(D) = |N(D)| = 3.$$

Untuk graf tak berarah sederhana, setiap sisi u,v\ muncul dua kali dalam daftar ketetanggaan:

- v muncul dalam daftar u;
- u muncul dalam daftar v.

Misalnya, sisi A,B\ muncul sebagai B dalam daftar A, dan A dalam daftar B.

Dengan menggunakan Handshaking Lemma,

$$\sum_{v \in V} \deg(v) = 2m,$$

jumlah total entri tetangga dalam semua daftar adalah 2m. Maka kebutuhan memori utama daftar ketetanggaan untuk graf tak berarah sederhana adalah O(n+m): ada n daftar, dan total isi daftar berorde 2m. Dalam notasi Big-O, faktor 2 diabaikan karena kita memperhatikan laju pertumbuhan asimtotik.

4.4.2 Keunggulan daftar ketetanggaan

Keunggulan utama daftar ketetanggaan adalah efisiensi ketika kita ingin menelusuri tetangga suatu simpul.

Misalnya, jika kita ingin memproses semua tetangga D, kita langsung membaca:

```
D: B, C, E
```

Biayanya sebanding dengan banyak tetangga D, yaitu

$$O(\deg(D)).$$

Secara umum, menelusuri semua tetangga v memerlukan waktu

$$O(\deg(v)).$$

Ini sangat efisien untuk algoritma traversal seperti BFS dan DFS, yang akan dipelajari pada Bab 7. Algoritma-algoritma tersebut pada dasarnya bergerak dari simpul ke tetangganya. Dengan daftar ketetanggaan, total waktu untuk menelusuri seluruh graf dapat dibuat O(n+m), karena setiap simpul diproses dan setiap sisi diperiksa dalam jumlah konstan kali (Cormen et al., 2009).

Sebaliknya, jika kita memakai matriks ketetanggaan dan ingin mencari semua tetangga v_i, kita harus memindai seluruh baris ke-i:

$$A_{i1}, A_{i2}, \ldots, A_{in}.$$

Biayanya O(n), bahkan jika v_i hanya memiliki dua tetangga.

Jadi, untuk graf jarang, daftar ketetanggaan biasanya jauh lebih hemat.

4.4.3 Kelemahan daftar ketetanggaan

Kelemahan utama daftar ketetanggaan adalah pertanyaan keberadaan sisi tidak selalu dapat dijawab dalam waktu konstan.

Misalkan kita ingin tahu apakah A bertetangga dengan D. Dalam daftar ketetanggaan, kita melihat daftar A:

```
A: B, C
```

Kita harus mencari apakah D muncul dalam daftar tersebut. Jika daftar disimpan sebagai array atau linked list biasa, pencarian ini memerlukan waktu

$$O(\deg(A)).$$

Untuk simpul berderajat kecil, ini murah. Tetapi untuk simpul dengan derajat sangat besar, pencarian dapat menjadi mahal jika dilakukan berulang kali.

Ada beberapa variasi implementasi.

Jika setiap daftar tetangga disimpan dalam bentuk terurut, maka pencarian dapat dilakukan dengan pencarian biner dalam waktu

$$O(\log \deg(v)).$$

Jika setiap daftar disimpan sebagai hash set, pencarian keanggotaan rata-rata dapat mendekati O(1), meskipun ada biaya memori tambahan dan detail teknis tentang hashing. Karena itu, “daftar ketetanggaan” bukan satu struktur tunggal; ia adalah keluarga representasi yang ide dasarnya sama: menyimpan tetangga aktual, bukan semua pasangan simpul yang mungkin.

4.5 Perbandingan matriks dan daftar ketetanggaan

Perbandingan inti dapat diringkas sebagai berikut.

Operasi atau biaya	Matriks ketetanggaan	Daftar ketetanggaan
Memori	$O(n^2)$	$O(n+m)$
Cek apakah (u,v) adalah sisi	O(1)	O(deg(u)) jika daftar biasa
Menelusuri semua tetangga u	O(n)	O(deg(u))
Menelusuri seluruh graf	$O(n^2)$	$O(n+m)$
Cocok untuk	graf padat, akses pasangan cepat	graf jarang, traversal efisien

Tabel ini bukan aturan mutlak. Ia adalah panduan awal. Dalam praktik, pilihan representasi bergantung pada:

- ukuran graf;
- kepadatan graf;
- operasi yang paling sering dilakukan;
- apakah graf berarah atau tak berarah;
- apakah sisi memiliki bobot;
- apakah data memiliki nilai hilang;
- apakah kita memakai algoritma berbasis matriks atau traversal;
- batas memori perangkat keras.

Untuk graf padat, m mendekati n^2, sehingga perbedaan antara $O(n^2)$ dan $O(n+m)$ tidak terlalu besar secara orde. Matriks ketetanggaan bisa menjadi pilihan alami, terutama jika algoritma membutuhkan banyak operasi aljabar linear.

Untuk graf jarang, m jauh lebih kecil daripada n^2. Daftar ketetanggaan biasanya lebih hemat memori dan lebih cepat untuk traversal.

4.6 Contoh statistik: jaringan korelasi

Misalkan kita memiliki n=1000 variabel biologis, misalnya ekspresi gen. Kita menghitung korelasi antara setiap pasangan variabel, lalu membuat graf dengan aturan:

$$\{i,j\} \in E \quad \text{jika dan hanya jika} \quad |\operatorname{corr}(X_i, X_j)| > 0.8.$$

Dalam graf ini:

- simpul mewakili variabel;
- sisi mewakili korelasi absolut yang melewati ambang tertentu.

Jika hanya 5000 pasangan yang melewati ambang, maka

$$m = 5000.$$

Jumlah pasangan maksimum adalah

$$\binom{1000}{2} = 499{,}500.$$

Graf ini relatif jarang. Daftar ketetanggaan akan menyimpan sekitar 2m=10,000 entri tetangga untuk graf tak berarah, ditambah struktur untuk 1000 simpul. Matriks ketetanggaan akan menyimpan

$$1000^2 = 1{,}000{,}000$$

entri.

Jika analisis utama adalah mencari komponen terhubung, menjalankan BFS, atau menghitung derajat, daftar ketetanggaan lebih efisien.

Namun jika analisis utama adalah melakukan operasi spektral pada matriks ketetanggaan atau Laplacian, representasi matriks dapat lebih alami. Dalam praktik modern, graf jarang sering disimpan sebagai matriks sparse, yaitu format matriks yang hanya menyimpan entri tak nol. Secara konseptual, matriks sparse berada di antara dunia matriks dan daftar ketetanggaan: ia tetap dipandang sebagai matriks, tetapi tidak menyimpan semua nol.

Kita belum membahas matriks sparse secara teknis dalam bab ini, tetapi ide dasarnya penting: representasi matematis dan representasi memori tidak selalu identik. Sebuah objek dapat ditulis sebagai matriks dalam rumus, tetapi disimpan dalam format hemat memori ketika dihitung.

4.7 Representasi untuk graf berarah

Pada Bab 3, kita telah memperkenalkan graf berarah atau digraf. Dalam graf berarah, sisi memiliki arah. Sisi dari u ke v ditulis sebagai pasangan berurutan

$$(u, v).$$

Pasangan ini berbeda dari

$$(v, u).$$

Contoh data statistik yang alami untuk graf berarah adalah jaringan pengiriman pesan. Jika A mengirim pesan kepada B, belum tentu B mengirim pesan kepada A.

Misalkan

$$V = \{A, B, C, D\}$$

dan

$$E = \{(A, B), (A, C), (B, C), (C, A), (D, C)\}.$$

Matriks ketetanggaan untuk digraf

Untuk digraf, matriks ketetanggaan didefinisikan sebagai

$$A_{ij} = \begin{cases} 1, & \text{jika } (v_i, v_j) \in E, \\ 0, & \text{jika } (v_i, v_j) \notin E. \end{cases}$$

Dengan urutan A,B,C,D, matriksnya adalah

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Matriks ini tidak harus simetris. Misalnya,

$$A_{1,2} = 1$$

karena ada sisi (A,B), tetapi

$$A_{2,1} = 0$$

karena tidak ada sisi (B,A).

Untuk digraf, ada dua jenis derajat:

- derajat keluar, ditulis $\deg^+(v)$, yaitu banyak sisi yang keluar dari v;
- derajat masuk, ditulis $\deg^-(v)$, yaitu banyak sisi yang masuk ke v.

Dalam matriks ketetanggaan digraf:

$$\deg^+(v_i) = \sum_{j=1}^{n} A_{ij},$$

yaitu jumlah baris ke-i, sedangkan

$$\deg^-(v_i) = \sum_{j=1}^{n} A_{ji},$$

yaitu jumlah kolom ke-i.

Untuk contoh di atas, baris A adalah

$$(0, 1, 1, 0),$$

maka

$$\deg^+(A) = 2.$$

Kolom C adalah

$$(1, 1, 0, 1)^\top,$$

maka

$$\deg^-(C) = 3.$$

Daftar ketetanggaan untuk digraf

Untuk digraf, daftar ketetanggaan biasanya menyimpan tetangga keluar:

```
A: B, C
B: C
C: A
D: C
```

Daftar ini menjawab pertanyaan: “ke mana simpul ini memiliki sisi keluar?”

Jika algoritma sering membutuhkan sisi masuk, kita dapat menyimpan daftar masuk juga:

```
A: C
B: A
C: A, B, D
D:
```

Jadi untuk digraf besar, kadang kita menyimpan dua struktur:

1. out-neighbor list, daftar tetangga keluar;
2. in-neighbor list, daftar tetangga masuk.

Biaya memori masing-masing adalah O(n+m). Jika keduanya disimpan, biaya tetap O(n+m) secara asimtotik, tetapi dengan konstanta lebih besar.

4.8 Representasi untuk graf berbobot

Dalam graf berbobot, setiap sisi memiliki nilai numerik yang disebut bobot. Bobot dapat berarti jarak, biaya, kapasitas, frekuensi, risiko, kekuatan hubungan, atau ukuran kemiripan.

Misalnya, dalam jaringan transportasi:

$$w(A, B) = 12.5$$

dapat berarti jarak antara kota A dan B adalah 12,5 kilometer. Dalam jaringan komunikasi:

$$w(A, B) = 37$$

dapat berarti terdapat 37 pesan dari A ke B. Dalam jaringan statistik antarvariabel:

$$w(i, j) = \mathrm{corr}(X_i, X_j)$$

dapat berarti bobot sisi adalah korelasi antara variabel X_i dan X_j.

Matriks bobot

Untuk graf berbobot, matriks ketetanggaan dapat diganti menjadi matriks bobot W, dengan

$$W_{ij} = w(v_i, v_j)$$

jika sisi ada.

Pertanyaan penting adalah: apa nilai W_{ij} jika sisi tidak ada?

Ada beberapa konvensi:

1. memakai 0;
2. memakai ∞, terutama untuk masalah jarak terpendek;
3. memakai nilai khusus seperti NA, NaN, atau null.

Pilihan ini tidak boleh sembarangan.

Jika bobot adalah frekuensi interaksi, maka 0 dapat berarti “tidak ada interaksi”, sehingga masuk akal memakai nol untuk tidak ada sisi.

Tetapi jika bobot adalah korelasi, bobot nol berarti korelasi tepat nol, bukan nilai hilang dan bukan otomatis “tidak ada sisi”. Jika kita memakai nol untuk menyatakan tidak ada sisi, kita harus memastikan bahwa nol tidak juga merupakan bobot sah yang bermakna lain.

Dalam statistik, perbedaan antara nilai nol, tidak ada sisi, dan data hilang sangat penting. “Tidak ada sisi” adalah klaim struktural. “Data hilang” adalah klaim observasional: kita tidak tahu apakah sisi ada. Keduanya tidak boleh dicampur tanpa asumsi eksplisit.

Daftar ketetanggaan berbobot

Dalam daftar ketetanggaan berbobot, setiap tetangga disimpan bersama bobotnya.

Contoh:

```
A: (B, 0.72), (C, 0.91)
B: (A, 0.72), (D, 0.64)
C: (A, 0.91), (D, 0.83)
D: (B, 0.64), (C, 0.83), (E, 0.58)
E: (D, 0.58)
```

Untuk graf tak berarah berbobot, pasangan bobot muncul dua kali jika kita menyimpan daftar dua arah:

```
A: (B, 0.72)
B: (A, 0.72)
```

Untuk graf berarah berbobot, bobot w(u,v) tidak harus sama dengan w(v,u), dan salah satunya bisa ada tanpa yang lain.

Representasi berbobot ini akan menjadi penting pada Bab 9 ketika kita membahas masalah jarak terpendek, dan pada Bab 10 ketika kita membahas pohon merentang minimum.

4.9 Graf sederhana, multigraf, dan gelang dalam representasi

Pada Bab 3, kita membedakan graf sederhana, multigraf, dan graf dengan gelang. Sekarang kita lihat konsekuensinya bagi representasi.

Graf sederhana

Untuk graf sederhana tak berarah:

- tidak ada gelang;
- tidak ada sisi rangkap;
- matriks ketetanggaan berisi 0 atau 1;
- matriks simetris;
- diagonal matriks bernilai 0.

Daftar ketetanggaan tidak perlu menyimpan tetangga yang sama lebih dari sekali.

Contoh:

```
A: B, C
```

berarti ada satu sisi dari A ke B, dan satu sisi dari A ke C.

Multigraf

Dalam multigraf, dua simpul dapat dihubungkan oleh lebih dari satu sisi. Misalnya, antara dua kota A dan B terdapat tiga jalur transportasi berbeda: jalan raya, kereta, dan penerbangan.

Jika kita memakai matriks, entri A_{ij} dapat menyimpan banyak sisi antara v_i dan v_j:

$$A_{ij} = \text{jumlah sisi antara } v_i \text{ dan } v_j.$$

Maka entri tidak lagi hanya 0 atau 1.

Jika kita memakai daftar ketetanggaan, kita dapat menyimpan tetangga berulang:

```
A: B, B, B
B: A, A, A
```

atau menyimpan pasangan dengan multiplicity:

```
A: (B, 3)
B: (A, 3)
```

Pilihan kedua biasanya lebih ringkas jika sisi rangkap hanya dibedakan oleh jumlahnya. Namun jika setiap sisi rangkap memiliki atribut berbeda, misalnya jenis transportasi dan biaya, kita perlu menyimpan daftar objek sisi:

```
A: (B, jalan, 120), (B, kereta, 95), (B, pesawat, 300)
```

Gelang

Gelang adalah sisi dari simpul ke dirinya sendiri. Dalam matriks ketetanggaan, gelang pada v_i muncul pada diagonal A_{ii}.

Untuk graf sederhana, diagonal nol. Untuk graf dengan gelang, diagonal dapat bernilai positif atau satu, tergantung konvensi.

Perlu kehati-hatian saat menghitung derajat. Dalam teori graf tak berarah, sebuah gelang biasanya menyumbang 2 terhadap derajat simpul, karena gelang bersisian dua kali dengan simpul yang sama. Konvensi ini adalah bagian dari definisi standar derajat pada graf dengan gelang (West, 2001). Karena itu, jika matriks menyimpan $A_{ii}=1$ untuk satu gelang, maka rumus derajat tidak cukup hanya menjumlahkan baris secara biasa kecuali diagonal diperlakukan khusus. Untuk graf dengan gelang, salah satu cara menulisnya adalah:

$$\deg(v_i) = 2A_{ii} + \sum_{j \neq i} A_{ij}$$

jika A_{ii} menghitung jumlah gelang di v_i.

Dalam banyak aplikasi statistik awal, kita memilih graf sederhana tanpa gelang agar interpretasi lebih jelas. Namun dalam beberapa konteks, gelang dapat bermakna. Misalnya, dalam rantai Markov, transisi dari suatu keadaan ke dirinya sendiri adalah objek penting. Jika keadaan dimodelkan sebagai simpul dan transisi sebagai sisi berarah berbobot, maka entri diagonal matriks transisi tidak boleh otomatis dihapus.

4.10 Membangun representasi dari data sisi

Dalam praktik, graf sering masuk sebagai edge list, yaitu tabel sisi. Misalnya:

sumber	target	bobot
A	B	0.72
A	C	0.91
B	D	0.64
C	D	0.83
D	E	0.58

Untuk graf tak berarah, kolom sumber dan target tidak menunjukkan arah; keduanya hanya menyatakan pasangan simpul. Untuk graf berarah, sumber dan target memiliki arti arah.

Dari tabel ini, kita dapat membangun daftar ketetanggaan:

```
A: (B, 0.72), (C, 0.91)
B: (A, 0.72), (D, 0.64)
C: (A, 0.91), (D, 0.83)
D: (B, 0.64), (C, 0.83), (E, 0.58)
E: (D, 0.58)
```

atau matriks bobot:

$$W = \begin{pmatrix} 0 & 0.72 & 0.91 & 0 & 0 \\ 0.72 & 0 & 0 & 0.64 & 0 \\ 0.91 & 0 & 0 & 0.83 & 0 \\ 0 & 0.64 & 0.83 & 0 & 0.58 \\ 0 & 0 & 0 & 0.58 & 0 \end{pmatrix}.$$

Namun sebelum membangun representasi, kita harus memutuskan beberapa hal.

Pertanyaan pemodelan sebelum implementasi

Pertama, apakah graf berarah?

Jika data adalah “A mengirim pesan ke B”, maka arah penting. Jika data adalah “A dan B berada dalam rumah tangga yang sama”, arah tidak penting.

Kedua, apakah sisi berbobot?

Jika data mencatat jumlah transaksi, frekuensi komunikasi, jarak, atau korelasi, bobot mungkin penting. Jika kita hanya peduli ada atau tidaknya relasi, kita dapat membinarisasi sisi.

Ketiga, apakah sisi rangkap dipertahankan?

Misalnya, jika A mengirim 20 pesan ke B, apakah itu:

- 20 sisi rangkap;
- satu sisi berbobot 20;
- satu sisi tak berbobot?

Ketiga pilihan itu menghasilkan graf berbeda. Dalam statistik, pilihan tersebut adalah keputusan pemodelan, bukan sekadar keputusan teknis.

Keempat, bagaimana menangani nilai hilang?

Jika pasangan A,B tidak muncul dalam tabel, apakah berarti tidak ada hubungan, atau hubungan belum diobservasi? Dalam data survei jaringan, tidak adanya laporan hubungan dapat berarti tidak ada hubungan, lupa melaporkan, batas desain survei, atau nonrespons. Representasi graf harus mencerminkan asumsi yang dipilih.

4.11 Pseudokode sederhana

Untuk memperjelas hubungan antara definisi dan implementasi, kita tuliskan pseudokode. Pseudokode bukan bahasa pemrograman tertentu, tetapi deskripsi algoritmik yang cukup rinci untuk diterjemahkan ke bahasa seperti Python, R, Julia, C++, atau Java.

Membuat matriks ketetanggaan untuk graf tak berarah sederhana

Input:

- daftar simpul $V = v_1, \ldots, v_n$;
- daftar sisi E.

Langkah:

```
buat matriks A berukuran n x n berisi 0

untuk setiap sisi {u, v} dalam E:
i = indeks[u]
j = indeks[v]
A[i][j] = 1
A[j][i] = 1
```

Untuk graf berarah, baris terakhir diubah menjadi:

```
A[i][j] = 1
```

tanpa otomatis mengisi A[j][i].

Untuk graf berbobot, kita mengisi bobot:

```
A[i][j] = bobot(u, v)
```

```
A[j][i] = bobot(u, v) jika graf tak berarah
```

Biaya awal membuat matriks nol adalah O(n²), karena ada n² entri yang diinisialisasi. Setelah itu, memproses semua sisi memerlukan O(m). Jadi total waktu pembangunan matriks padat adalah

$$O(n^2 + m).$$

Karena m≤ n² untuk banyak kelas graf sederhana, ini sering ditulis sebagai O(n²) dalam pembahasan kasar.

Membuat daftar ketetanggaan untuk graf tak berarah sederhana

Input:

- daftar simpul V;
- daftar sisi E.

Langkah:

```
untuk setiap simpul u dalam V:
Adj[u] = daftar kosong

untuk setiap sisi {u, v} dalam E:
tambahkan v ke Adj[u]
tambahkan u ke Adj[v]
```

Untuk graf berarah:

```
tambahkan v ke Adj[u]
```

karena hanya ada sisi dari u ke v.

Untuk graf berbobot:

```
tambahkan (v, w) ke Adj[u]
tambahkan (u, w) ke Adj[v] jika graf tak berarah
```

Inisialisasi daftar untuk semua simpul memerlukan O(n). Memproses semua sisi memerlukan O(m). Jadi total waktu pembangunan daftar ketetanggaan adalah

$$O(n + m).$$

Analisis semacam ini merupakan contoh awal bagaimana definisi graf, struktur data, dan kompleksitas algoritma saling terkait. Buku algoritma standar seperti Cormen et al. membahas representasi matriks dan daftar ketetanggaan sebagai fondasi untuk algoritma graf dasar seperti BFS, DFS, dan jalur terpendek (Cormen et al., 2009).

4.12 Kapan memilih matriks dan kapan memilih daftar?

Tidak ada representasi yang selalu terbaik. Pilihan tergantung pada pertanyaan.

Pilih matriks ketetanggaan jika:

- graf relatif padat;
- n tidak terlalu besar;
- kita sering menanyakan apakah dua simpul bertetangga;
- kita memakai metode aljabar linear;
- kita ingin bekerja langsung dengan matriks ketetanggaan, matriks derajat, atau Laplacian;
- kita menghitung kuantitas yang melibatkan banyak pasangan simpul.

Contoh statistik:

Jika kita memiliki 200 variabel dan ingin membangun graf berdasarkan korelasi pasangan variabel, matriks 200× 200 masih mudah ditangani. Jika analisis berikutnya berbasis eigenvalue atau spectral clustering, representasi matriks sangat alami.

Pilih daftar ketetanggaan jika:

- graf jarang;
- n besar;
- kita sering menelusuri tetangga;
- kita menjalankan BFS, DFS, pencarian komponen, atau algoritma lokal;
- kita ingin menghemat memori;
- sisi memiliki atribut dan hanya sebagian kecil pasangan simpul terhubung.

Contoh statistik:

Jika kita memiliki jaringan komunikasi dengan 10 juta pengguna dan 50 juta interaksi teragregasi, matriks \(10^7\times

Document information

Bab 4: Representasi Graf untuk Perhitungan

Project	Teori Graf untuk Statistik
Document	Document 1.8
Author	Harizahayu
Verifier	Not verified
Downloaded	July 08, 2026 10:38 KST
Status	Working
Document link	https://theorytrace.com/projects/teori-graf-untuk-statistik/documents/bab-4-representasi-graf-untuk-perhitungan/