Maximum Flow

Tim Olimpiade Komputer Indonesia

Pendahuluan

Melalui dokumen ini, kalian akan:

- Memahami persoalan *maximum flow*
- Memahami metode untuk menyelesaikan *maximum flow*
- Memahami aplikasi persoalan *maximum flow*

Persoalan Maximum Flow

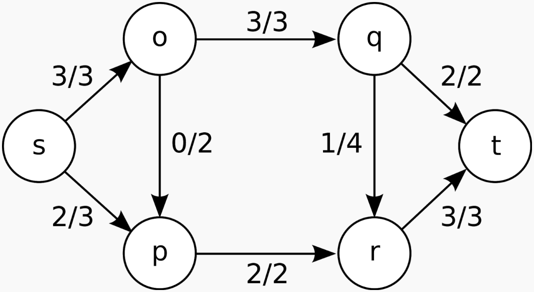
- Diberikan sebuah directed *graph*. Setiap *edge* memiliki kapasitas.
- Salah satu *node* merupakan **source** (sumber, biasanya dinotasikan *node* s), dan salah satu *node* lainnya merupakan **sink** (pembuangan, biasanya dinotasikan *node* t).
- Sebuah **flow** (aliran) adalah pemetaan dari setiap *edge* ke sebuah bilangan bulat (banyaknya aliran) tidak lebih dari kapasitasnya sehingga untuk seluruh *node* selain *source* dan *sink*, banyaknya aliran yang masuk ke *node* tersebut sama dengan banyaknya aliran yang keluar dari *node* tersebut.

Persoalan Maximum Flow (lanj.)

- Lebih resminya, sebuah *flow* adalah pemetaan $f : V \times V \to \mathbb{Z}$ sedemikian sehingga $0 \leq f(u, v) \leq c(u, v)$ dan untuk setiap *node* u selain *source* dan *sink*, $\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u)$.
- Nilai dari sebuah *flow* adalah banyaknya aliran yang keluar dari *source* dikurangi dengan banyaknya aliran yang masuk ke *source* (dengan kata lain, $\sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$).
- Persoalan **maximum flow** adalah mencari *flow* dengan nilai maksimum.

Contoh Maximum Flow

Ilustrasi berikut adalah contoh maximum flow.
Angka pada setiap edge menyatakan banyaknya aliran pada maximum flow dan kapasitas pada edge tersebut. Pada contoh ini, maximum flow memiliki nilai 5.

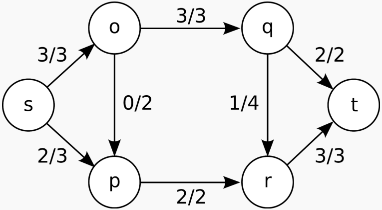


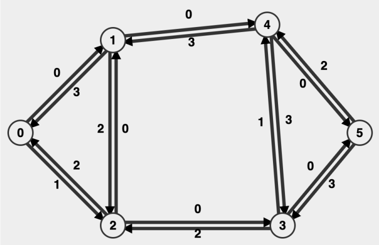
Residual capacity

- **Residual capacity** (kapasitas tersisa) dari setiap *edge* adalah sisa kapasitas *edge* tersebut, yaitu kapasitas awal dikurangi banyaknya aliran.
- Selain *edge* yang terdapat pada *graph* awal, kita juga menambahkan *back-edge* dengan arah kebalikan, dengan kapasitas sama dengan banyaknya aliran *edge* tersebut.
- *Back-edge* untuk mengembalikan aliran yang sudah dikirim sebelumnya (mengurangi banyaknya aliran pada *edge*). Bagian ini akan lebih jelas pada contoh metode Ford Fulkerson di bawah.

Contoh residual capacity

Sebagai contoh, *graph* di gambar kiri merupakan banyaknya aliran dan kapasitas setiap *edge*, sedangkan graph di gambar kanan merupakan *residual capacity* setiap *edge*.



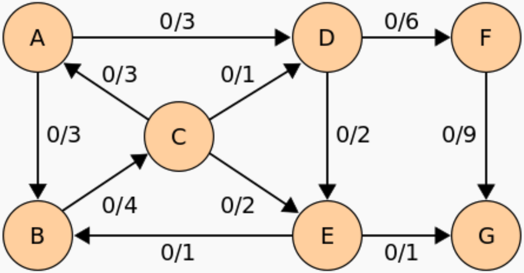


Metode Ford Fulkerson

- Metode **Ford Fulkerson** mencari **augmenting path** (sebuah *path* yang melewati *edge* dengan *residual capacity* positif) secara terus menerus, dan menambahkan aliran melalui *path* tersebut.
- Kita dapat menambahkan aliran sebanyak **bottleneck** (*residual capacity* terkecil dari seluruh *edge* yang dikunjungi) dari *path* tersebut, sehingga nilai *maximum flow* dapat ditambahkan dengan banyaknya aliran.
- Untuk seluruh *edge* (u, v) yang dikunjungi oleh *path*, kita harus mengurangi *residual capacity edge* (u, v) dengan banyaknya aliran dan menambahkan *residual capacity edge* (v, u) (*back-edge*) dengan banyaknya aliran.

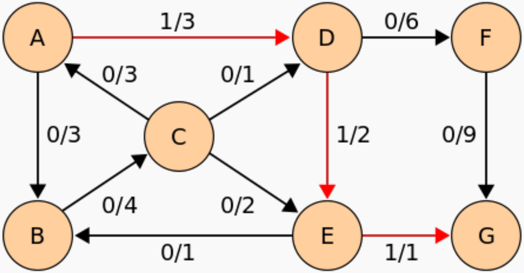
Contoh Ford Fulkerson

Sebagai contoh, diberikan *graph* berikut, dengan *node A* sebagai *source* dan *node G* sebagai *sink*.



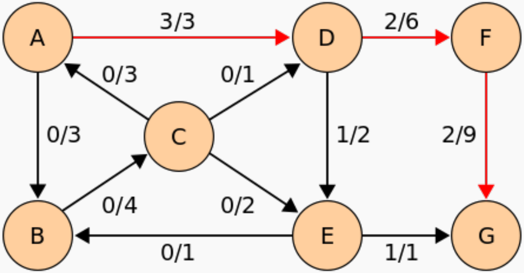
Contoh Ford Fulkerson (lanj.)

Berikut adalah salah satu *augmenting path*, melewati *node* A, D, E, dan G. *Bottleneck* dari *path* ini adalah 1 (*edge* (E, G)), sehingga nilai *maximum flow* kita tambahkan dengan 1 dan memperbaharui *residual capacity* setiap *edge* yang dilewati.



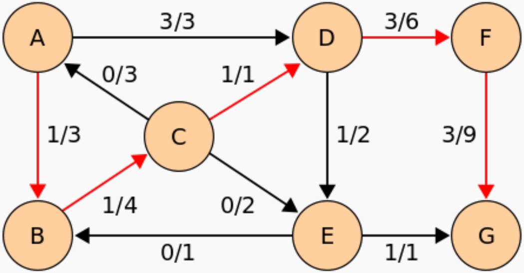
Contoh Ford Fulkerson (lanj.)

Masih terdapat *augmenting path*, salah satunya yang melewati *node* A, D, F, dan G. *Bottleneck* dari *path* ini adalah 2 (edge (A, D)), sehingga kita menambahkan nilai *maximum flow* dengan 2. *Maximum flow* sekarang adalah 3.



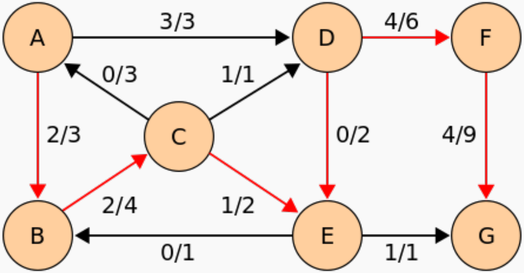
Contoh Ford Fulkerson (lanj.)

Masih terdapat *augmenting path* lagi, salah satunya yang melewati *node* A, B, C, D, F, dan G. *Bottleneck* dari *path* ini adalah 1 (*edge* (C, D)), sehingga kita menambahkan nilai *maximum flow* dengan 1. *Maximum flow* sekarang adalah 4.



Contoh Ford Fulkerson (lanj.)

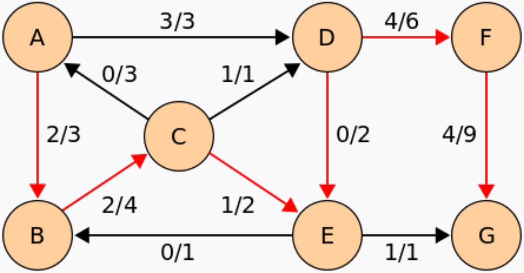
Masih terdapat *augmenting path* lagi. *Augmenting path* ini memanfaatkan *back-edge* (D, E) dengan *residual capacity* 1. *Augmenting path* ini mengembalikan aliran yang melewati *edge* (D, E) (aliran pertama) dan menggantinya untuk melewati *edge* (D, F).



Contoh Ford Fulkerson (lanj.)

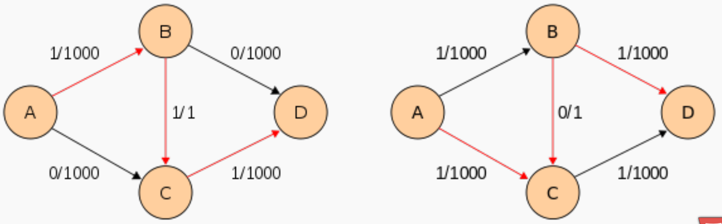
Bottleneck dari *augmenting path* terakhir adalah 1 (*back-edge* (D, E)), sehingga kita menambahkan nilai *maximum flow* dengan 1. *Maximum flow* sekarang adalah 5.

Tidak terdapat lagi *augmenting path*, sehingga nilai *maximum flow* adalah 5.



Mencari Augmenting Path

- Jika pencarian *augmenting path* dilakukan dengan DFS, maka kompleksitas solusinya adalah $O(E * |f|)$, dengan $|f|$ adalah nilai *maximum flow*.
- Ini disebabkan karena sekali pencarian *augmenting path* membutuhkan waktu $O(E)$ dan dapat menambahkkan 1 pada nilai *maximum flow*.
- Berikut contoh *graph* yang memerlukan $|f|$ pencarian *augmenting path*.



Algoritme Edmonds-Karp

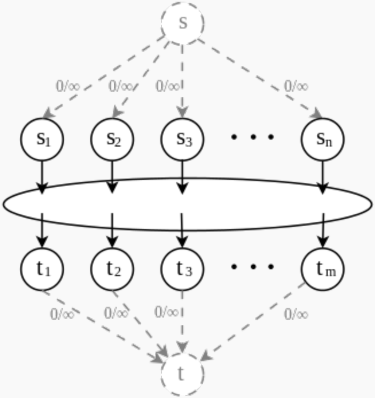
- Algoritme **Edmonds-Karp** adalah implementasi metode Ford Fulkerson dengan menggunakan BFS untuk mencari *augmenting path*.
- Jika kita menggunakan BFS untuk mencari *augmenting path* (yang akan mengembalikan *augmenting path* dengan menggunakan paling sedikit *edge*), maka kita hanya akan mencari paling banyak $min(O(VE), |f|)$ *augmenting path*.
- Sehingga kompleksitas waktu dari algoritme ini adalah $O(min(|V||E|^2, |E| \times |f|))$.

Variasi Maximum Flow: Minimum Cut

- Diberikan sebuah *weighted directed graph* dengan salah satu *node* diberi label s dan salah satu *node* lainnya diberi label t.
- Persoalan **minimum cut** adalah persoalan membagi himpunan *node* menjadi dua buah subhimpunan *node* S dan T (dengan $s \in S$ dan $t \in T$) yang meminimumkan total bobot perpotongan, yaitu $\sum_{u \in S} \sum_{v \in T} w(u, v)$.
- Teorema **max-flow min-cut** menyatakan bahwa nilai *minimum cut* ini sama dengan nilai *maximum flow* dengan *source* s dan *sink* t.

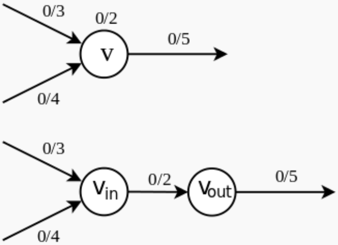
Variasi Maximum Flow: Lebih dari satu source/sink

- Jika terdapat lebih dari satu *source*/*sink*, maka kita dapat menambahkan dua *node* baru, super *source* dan super *sink*, lalu menghubungkan super *source* ke seluruh *source* dan seluruh *sink* ke super *sink* (dengan kapasitas tak terhingga).
- Jalankan algoritme *maximum flow* dengan memilih super *source* sebagai *source* dan super *sink* sebagai *sink*.



Variasi Maximum Flow: Kapasitas Node

- Jika setiap *node* juga memiliki kapasitas banyaknya aliran yang dapat melewati *node* tersebut, maka kita dapat memisahkan *node* tersebut menjadi dua *node* dan menambahkan *edge* yang menghubungkan keduanya dengan kapasitas *node* tersebut.
- Sebagai contoh, jika *node* v pada gambar di samping (atas) memiliki kapasitas 2, maka kita dapat memisahnya seperti gambar di samping (bawah)



Contoh Aplikasi: COCI 2017/2018 Round #7 (PRIGLAVCI)

- Terdapat N murid dan M halte pada koordinat 2 dimensi.
- Terdapat K rute bus yang melewati beberapa halte. Hanya terdapat 1 bus pada setiap rute.
- Setiap bus dapat berisi maksimal C murid.
- Setiap murid dapat naik bus dari halte manapun di suatu rute, namun hanya dapat turun dari bus di halte terakhir rute tersebut.
- Kita ingin mengatur setiap murid harus naik bus dari halte mana, sehingga kapasitas bus mencukupi untuk semua murid dan “kelemahan”-nya minimum.
- “Kelemahan” suatu pengaturan didefinisikan sebagai jarak Euclidean yang paling maksimum antara seorang murid ke halte yang harus dikunjunginya.

Solusi: COCI 2017/2018 Round #7 (PRIGLAVCI)

- Kita dapat menyelesaikan soal ini menggunakan Binary Search the Answer
- Untuk setiap x, kita dapat memeriksa apakah terdapat solusi dengan "kelemahan" tidak lebih dari x dengan cara mengkonstruksi *graph* berikut:
  - Tambahkan satu node untuk setiap murid, rute, dan halte.
  - Tambahkan *node source* dan *sink*.
  - Tambahkan *edge* dari *source* ke setiap murid dengan kapasitas 1.
  - Tambahkan *edge* dari setiap rute ke *sink* dengan kapasitas C.
  - Untuk setiap murid m dan rute r, jika terdapat halte h pada rute r dengan jarak Euclidean dari murid m ke halte h tidak lebih dari x, maka tambahkan *edge* dari murid m ke rute r dengan kapasitas 1.

Solusi: COCI 2017/2018 Round #7 (PRIGLAVCI) (lanj.)

- Jika nilai *maximum flow* dari *graph* yang dikonstruksi adalah N, maka terdapat pemetaan untuk setiap siswa ke rute bis, dengan jarak Euclidean murid dan halte tempat murid tersebut naik bis tidak lebih dari x, dan seluruh bis tidak melebihi kapasitasnya.
  - Dalam kasus ini, kita dapat mencoba untuk menurunkan nilai x pada binary search.
- Kita dapat menginterpretasikan sebuah aliran dari *source* $\rightarrow$ murid $m \rightarrow$ rute $r \rightarrow$ *sink* sebagai murid m mengambil rute r.