TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN TOÁN TỔ HỢP VÀ ĐỒ THỊ**

**Project: Combinatorial Optimization**

*Người hướng dẫn*:

*Người thực hiện*: **NGUYỄN TRỌNG THẮNG – 51704014**

**HỒ TRẦN CÔNG THUẬN – 51703194**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn thầy đã nhiệt tình trong việc tận tình chỉ dạy chúng em, cung cấp cho chúng em nhiều kiến thức hay và bổ ích cho môn học. Chúng em hiểu là thầy luôn muốn tốt cho chúng em.

Chúng em xin chân thành cảm ơn thầy. Chúc thầy nhiều sức khỏe và thành công trên con đường giảng dạy của mình.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của TS Vũ Đình Hồng. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Quốc Thắng*

*Hồ Trần Công Thuận*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc20257)

[ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG ii](#_Toc7049)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc28907)

[TÓM TẮT iv](#_Toc11816)

[MỤC LỤC 1](#_Toc14005)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc13036)

[CHƯƠNG 1 5](#_Toc14492)

[1.1 Maximum Network Flow 5](#_Toc17597)

[1.1 Thuật toán Ford-Fulkerson 6](#_Toc17597)

[CHƯƠNG 2 8](#_Toc14492)

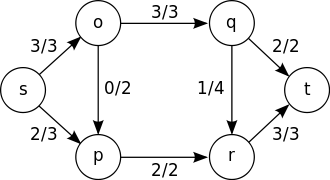
[2.1 Bài toán Max Flow 8](#_Toc17597)

[2.2 Shortest Path 15](#_Toc25660)

CHƯƠNG 1

1.1 Maximum Network Flow

Luồng cực đại là một trong những bài toán tối ưu trên đồ thị tìm được những ứng dụng rất rộng rãi trong cả thực tế cũng như trong lý thuyết tổ hợp. Bài toán được đề xuất vào đầu những năm 1950 và gắn liền với tên tuổi của 2 nhà toán học Mỹ [Lester Randolph Ford](https://vi.wikipedia.org/w/index.php?title=Lester_Randolph_Ford&action=edit&redlink=1) và [Delbert Ray Fulkerson](https://vi.wikipedia.org/wiki/Delbert_Ray_Fulkerson).



Mạng (network) là một [đồ thị có hướng](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_th%E1%BB%8B_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)#%C4%91%E1%BB%93_th%E1%BB%8B_c%C3%B3_h%C6%B0%E1%BB%9Bng) G = (V, E) trong đó:

Có duy nhất một đỉnh s không có cung đi vào, được gọi là đỉnh phát (source)

Có duy nhất một đỉnh t không có cung đi ra, được gọi là đỉnh thu (sink)

Mỗi cạnh e = (u, v) ∈ E được gán một số nguyên không âm c(e) = c[u, v] và gọi là khả năng thông qua của cung đó (capacity).

Ta quy ước nếu mạng không có cung (u, v) thì ta thêm vào cung (u, v) với khả năng thông qua c[u, v] bằng 0.

Với một mạng G = (V, E, c), ta ký hiệu:

W-(x) = {(u, v) ∈ E | u ∈ V}: tập các cung đi vào đỉnh v.

W+(x) = {(v, u) ∈ E | u ∈ V}: tập các cung đi ra khỏi đỉnh v.

Tính chất của luồng

Với tập B ⊆ V, ký hiệu:

W-(B) = { (a, b) ∈ E | a ∉ B, b ∈ B } - tập cạnh từ ngoài B đi vào B.

W+(B) = { (a, b) ∈ E | a ∈ B, b ∉ B } - tập cạnh từ B đi ra khỏi B.

Khi đó nếu tập con các đỉnh B không chứa x0 và z thì: t (W-(B)) =t (W+(B)).Theo tính chất b) của luồng: ∑ t (W-(x)) =∑ t (W+(x)).

Cạnh kề với đỉnh x nếu có đỉnh đầu và đỉnh cuối đều nằm trong tập B thì nó sẽ có mặt ở cả hai vế củađẳng thức đúng một lần, do đó có thể giản ước. Giá trị của luồng : Giá trị của một luồng được tính bằng tổng giá trị trên các cung đi ra từ đỉnh nguồn s, hoặc tổng giá trị trên các cung đi vào đỉnh thứ t : val(f) = t(W+(s)) = t(W-(t)).

1.2 Thuật toán Ford-Fulkerson

Thuật toán Ford- Fulkerson (đặt theo [L. R. Ford](https://vi.wikipedia.org/w/index.php?title=L._R._Ford&action=edit&redlink=1) và [D. R. Fulkerson](https://vi.wikipedia.org/w/index.php?title=D._R._Fulkerson&action=edit&redlink=1)) tính toán [luồng cực đại](https://vi.wikipedia.org/wiki/B%C3%A0i_to%C3%A1n_lu%E1%BB%93ng_c%E1%BB%B1c_%C4%91%E1%BA%A1i) trong một [mạng vận tải](https://vi.wikipedia.org/w/index.php?title=M%E1%BA%A1ng_v%E1%BA%ADn_t%E1%BA%A3i&action=edit&redlink=1). Tên Ford-Fulkerson cũng thường được sử dụng cho [thuật toán Edmonds-Karp](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_Edmonds%E2%80%93Karp), một trường hợp đặc biệt của thuật toán Ford-Fulkerson.

Ý tưởng đằng sau [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) rất đơn giản: miễn là tồn tại một đường đi từ nguồn (nút bắt đầu) đến điểm xả (nút cuối), với điều kiện tất cả các cung trên đường đi đó vẫn còn khả năng thông qua, thì ta sẽ gửi đi một luồng dọc theo đường đi đó. Sau đó chúng ta tìm một đường đi khác, và tiếp tục như vậy. Một đường đi còn khả năng thông qua là một đường đi có khả năng mở rộng thêm hay một đường đi mà luồng qua đó còn khả năng tăng thêm - gọi tắt là [đường tăng](https://vi.wikipedia.org/wiki/Lu%E1%BB%93ng_tr%C3%AAn_m%E1%BA%A1ng#%C4%90%E1%BB%8Bnh_ngh%C4%A9a).

Cho một đồ thị {\displaystyle G(V,E)}G(V,E), với các khả năng thông qua {\displaystyle c(u,v)}c(u,v) và luồng f(u,v)=0 {\displaystyle f(u,v)=0}trên các cung từ **u**{\displaystyle u} đến **v** {\displaystyle v}, ta muốn tìm luồng cực đại từ đầu nguồn **s{\displaystyle s}**đến điểm thoát **t**{\displaystyle t}. Sau mỗi bước, các điều kiện sau đây được duy trì:

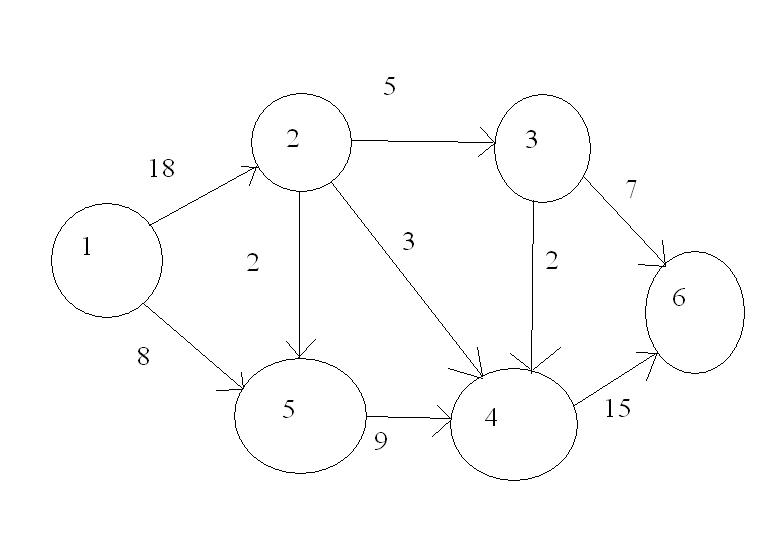
* f(u,v) ≤ c(u,v) {\displaystyle \ f(u,v)\leq c(u,v)}Luồng từ {\displaystyle u} tới {\displaystyle v} không vượt quá khả năng thông qua.
* f(u,v) = -f(v,u) {\displaystyle \ f(u,v)=-f(v,u)}Ta duy trì cân bằng luồng.
* ∑f(u,v) = 0 cho tất cả các nút ngoại trừ **s** và **t**. {\displaystyle \ \sum \_{v}f(u,v)=0} {\displaystyle t}Lượng dòng chảy vào một nút bằng lượng chảy ra khỏi một nút.

Điều này có nghĩa là một luồng đi qua một mạng là một luồng *hợp lệ* sau mỗi vòng của thuật toán. Chúng ta định nghĩa mạng còn dư {\displaystyle G\_{f}(V,E\_{f})} là mạng với sức chứa  {\displaystyle c\_{f}(u,v)=c(u,v)-f(u,v)} và luồng bằng không. Chú ý rằng không chắc chắn là  {\displaystyle E=E\_{f}}, bởi vì việc gửi luồng theo cung u,v{\displaystyle u,v} có thể làm ngắt u,v{\displaystyle u,v} (làm nó bão hòa), nhưng lại mở một cung mới v,u{\displaystyle v,u} trong mạng còn dư.

CHƯƠNG 2

2.1 Bài toán Max Flow

Xét bài toán tìm luồng cực đại trong mạng sau:



Đầu tiên, ta khởi tạo luồng f = 0.

Đồ thị tăng luồng của luồng f giống như trên hiện chỉ có các cung thuận; các cung nghịch hiện đều có giá trị bằng 0.

Chọn 1 đường đi bất kì từ điểm phát 1 tới điểm thu 6.

Ở đây mình chọn đường đi P1: 1 – 2 – 3 – 6.

Tiếp theo, tìm trong các cung của đường đi P1 giá trị trọng số nhỏ nhất, ta gọi là e. Ở đây đó là cung (2,3) với trọng số là 5. Vậy e = 5

Tiến hành tăng luồng dọc theo đường P1. Ta xây dựng luồng f1 như sau:

f1 (1,2) = 0 + e = 5

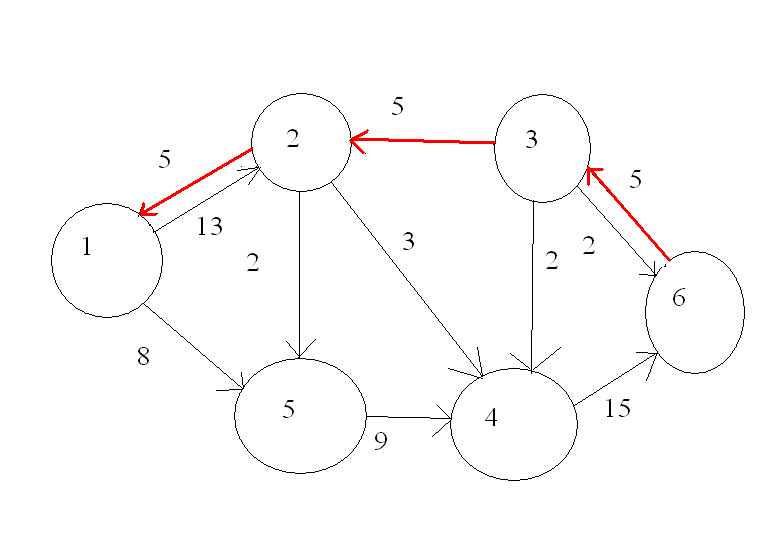
f1 (2,3) = 0 + e = 5

f1 (3,6) = 0 + e = 5

(do các cung trên đều là cung thuận nên ta cộng thêm e vào, nếu là cung nghịch thì ta trừ đi e)

Giá trị luồng qua các cung còn lại vẫn như cũ, bằng 0.

Ta được đồ thị tăng luồng của của f1:



Ta thấy bây giờ đã xuất hiện các cung nghịch, có chiều ngược với chiều cung thuận ban đầu. Giá trị của cung nghịch bằng giá trị của luồng đã đi qua đoạn tương ứng với cung đó. Nếu giá trị của cung bằng 0 thì mặc định là nó không được vẽ ra.

Ví dụ đoạn [1,2] đã đi qua 5 đơn vị trong khi khả năng thông qua là 18. Vậy ta còn có thể tăng thêm tối đa 13 đơn vị qua đây nữa (biểu diễn bằng 13 đơn vị ở cung thuận). Đồng thời, ta cũng có thể giảm đi tối đa 5 đơn vị (biểu diễn bằng 5 đơn vị ở cung nghịch).

Tiếp tục lần lặp tiếp theo. Ta lại tìm 1 đường đi từ 1 đến 6. Đường đi P1 ở trên không thể đi được nữa vì cung (2,3) đã không còn để đi từ 2 đến 3. Ta phải đi từ 1 đến 6 bằng con đường khác. Ở đây mình chọn đường sau:

P2: 1 – 2 – 4 – 6

Giá trị trọng số nhỏ nhất của các cung trên đường đi P2 là e = 3 ứng với cung (2,4)

Tăng luồng theo đường P2, ta xây dựng luồng f2 như sau:

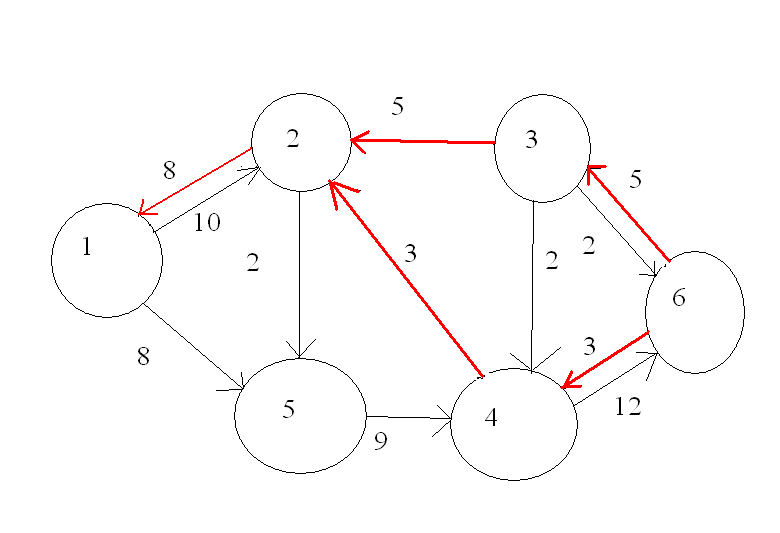
f2 (1,2) = 5 + e = 8

f2 (2,4) = 0 + e = 3

f2 (4,6) = 0 + e = 3

Giá trị luồng ở các cung khác vẫn giữ nguyên như ở luồng f1

Ta có đồ thị tăng luồng ứng với luồng f2 như sau:



Lặp lại quá trình, ta đi tìm đường đi từ 1 đến 6.

Ở đây mình chọn đường đi:

P3: 1 – 2 – 5 – 4 – 6

Giá trị trọng số nhỏ nhất trong các cung của đường đi P3 là e = 2 ứng với cạnh (2,5)

Các cung của đường đi vẫn đều là cung thuận nên ta tiến hành cộng thêm e vào để xây dựng luồng f3:

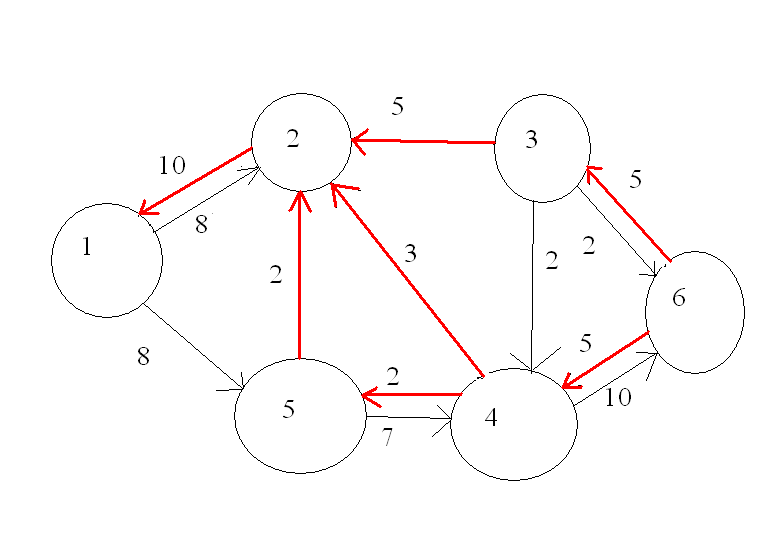
f3 (1,2) = 8 + e = 10

f3 (2,5) = 0 + e = 2

f3 (5,4) = 0 + e = 2

f3 (4,6) = 3 + e = 5

Ta có đồ thị tăng luồng của f3:



Dễ nhận thấy bây giờ chỉ còn 1 đường để đi từ 1 đến 6, dĩ nhiên ta sẽ chọn đường đó để tăng luồng.

P4: 1 – 5 – 4 – 6

Giá trị e = 7 ứng với cung (5,4)

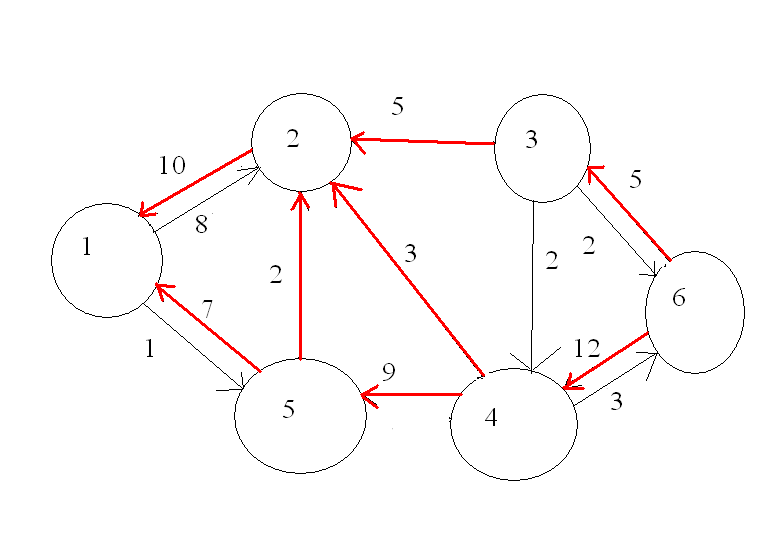
Các cung ở đây vẫn chỉ đều là cung thuận nên ta cộng thêm e vào để xây dựng luồng f4:

f4 (1,5) = 0 + e = 7

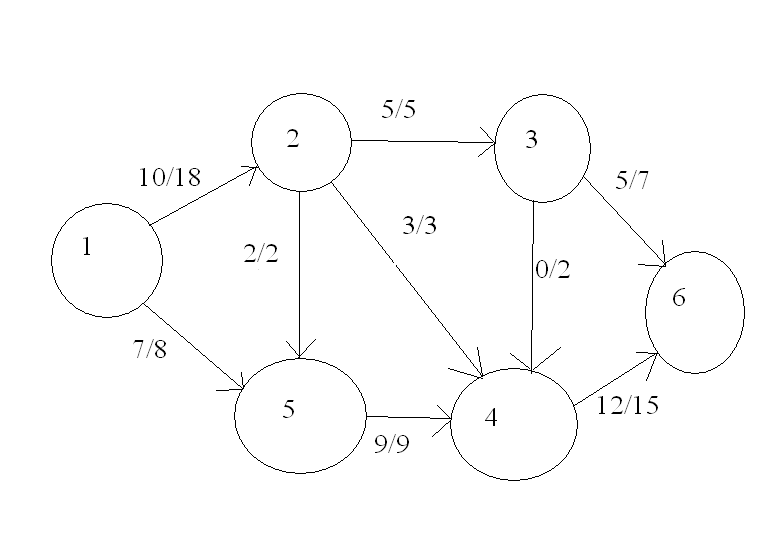
f4 (5,4) = 2 + e = 9

f4 (4,6) = 5 + e = 12

Đồ thị tăng luồng của f4 là:



Đến đây, do không còn có thể tìm được đường tăng luồng, tức là 1 đường đi từ 1 đến 6 nữa, nên ta có giá trị luồng f4 chính là giá trị luồng cực đại của bài toán. Giá trị luồng cực đại là f\* = f4 = 17



2.2 Shortest Path

Thuật toán Dijkstra, mang tên của nhà khoa học máy tính người Hà Lan [Edsger Dijkstra](https://vi.wikipedia.org/wiki/Edsger_Dijkstra) vào năm 1956 và ấn bản năm 1959, là một [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) giải quyết [bài toán đường đi ngắn nhất](https://vi.wikipedia.org/wiki/B%C3%A0i_to%C3%A1n_%C4%91%C6%B0%E1%BB%9Dng_%C4%91i_ng%E1%BA%AFn_nh%E1%BA%A5t) nguồn đơn trong một [đồ thị có hướng](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_th%E1%BB%8B_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)#%C4%91%E1%BB%93_th%E1%BB%8B_c%C3%B3_h%C6%B0%E1%BB%9Bng) không có [cạnh](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_th%E1%BB%8B_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)) mang trọng số âm. Thuật toán thường được sử dụng trong [định tuyến](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%8Bnh_tuy%E1%BA%BFn) với một [chương trình con](https://vi.wikipedia.org/wiki/Ch%C6%B0%C6%A1ng_tr%C3%ACnh_con) trong các thuật toán đồ thị hay trong công nghệ [Hệ thống định vị toàn cầu](https://vi.wikipedia.org/wiki/H%E1%BB%87_th%E1%BB%91ng_%C4%91%E1%BB%8Bnh_v%E1%BB%8B_to%C3%A0n_c%E1%BA%A7u) ([GPS](https://vi.wikipedia.org/wiki/GPS)).

Xét đồ thị G =(X,E) với các cạnh có trọng số không âm. - Dữ liệu nhập cho thuật toán là ma trận trọng số L (với qui ước Lhk = +∞ nếu không có cạnh nối từ đỉnh h đến đỉnh k)và hai đỉnh x,y cho trước. - Dữ liệu xuất là đường đi ngắn nhất từ x đến y.

Hàm d(u) dùng để lưu trữ độ dài đường đi ngắn nhất từ đỉnh nguồn s đến đỉnh u. Rõ ràng d(s)= 0. Ký hiệu {\displaystyle X^{-}(u)} là tập tất cả các đỉnh có cạnh đi tới đỉnh u. Nếu với mọi {\displaystyle v\in X^{-}(u)} đã xác định được d(v) thì:

{\displaystyle d(u)=min\{d(v)+w(v,u),v\in X^{-}(u)\}}

.

Để tính được giá trị nhỏ nhất này, như thông thường khi khởi tạo ta phải gán cho d(v)={\displaystyle \infty } , sau đó gặp giá trị nhỏ hơn thì thay thế lại.

Những đỉnh đã tính được d(v)hữu hạn được cho vào một hàng đợi có ưu tiên. Hàng đợi này luôn được bổ sung và sắp xếp lại nên một cấu trúc hợp lý là cấu trúc đống nhị phân (heap)...

Để theo dõi trạng thái của các đỉnh trong quá trình xét, ta dùng hàm COLOR(u) xác định với mọi {\displaystyle v\in V}. Lúc đầu các đỉnh được tô màu trắng (WHITE), khi cho vào hàng đợi nó được tô màu xám (GRAY), khi đã tính xong khoảng cách nó được tô màu đen(BLACK).

Nếu cần ghi lại đường đi ta sẽ phải dùng một hàm con trỏ PRE(u) để chỉ đỉnh đứng ngay trước đỉnh u trên đường đi ngắn nhất từ s tới u.

**TÀI LIỆU THAM KHẢO**

**Tiếng Anh: https://www.geeksforgeeks.org/**

**https://www.hackerearth.com**

**wikipedia.org**