CHƯƠNG 6

Một số ứng dụng

GV: TRẦN QUỐC VIỆT

Một số ứng dụng

Giới thiệu 2 ứng dụng:

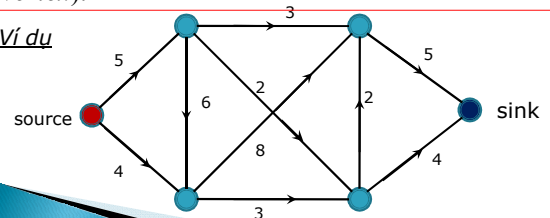
- Bài toán luồng cực đại (Max-flow problem)
- Bài toán ghép cặp (Matching problem)

1. Bài toán luồng cực đại
(Max flow problem)

Định nghĩa

- Mạng (network) là một đồ thị có hướng có trọng số G = (V,E) trên đó ta chọn một đỉnh gọi là đỉnh phát (*source vertex*) và 1 đỉnh gọi là đỉnh thu (*sink vertex*).

Ví dụ

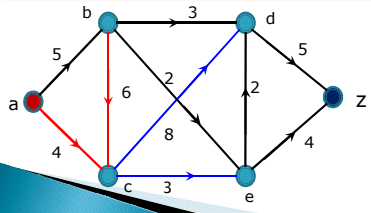


Định nghĩa

- Một mạng G = (V,E) với đỉnh phát là a, đỉnh thu là z, c(e) ∈ N là trọng số của cung e. Với mỗi đỉnh x, ta đặt:

  In(x) = {e ∈ E | e tới trong x}

  Out(x) = {e ∈ E | e tới ngoài x}



In(c)={$\overrightarrow{ac}$, $\overrightarrow{bc}$}

Out(c)={$\overrightarrow{cd}$, $\overrightarrow{ce}$}

5

Định nghĩa

- Một hàm tải (*flow function*) trên G được định nghĩa bởi ánh xạ:

$$\varphi: E \rightarrow N$$

thỏa các điều kiện

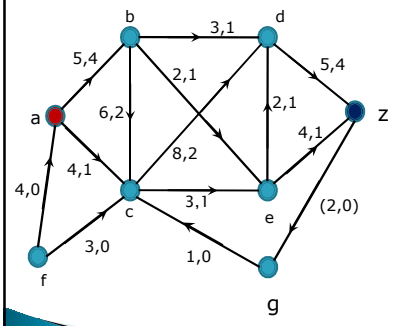
(i) $\varphi(e) \le c(e), \forall e \in E$

(ii) $\varphi(e) = 0, \forall e \in In(a) \cup Out(z)$

(iii) $\sum_{e \in In(x)} \varphi(e) = \sum_{e \in Out(x)} \varphi(e), \forall x \in V \setminus \{a, z\}$

6

Ví dụ về hàm tải
a:source, z:sink



$\varphi(\overrightarrow{fa}) = 0$
$\varphi(\overrightarrow{zg}) = 0$
$\varphi(\overrightarrow{ab}) = 4$
$\varphi(\overrightarrow{ac}) = 1$
$\varphi(\overrightarrow{fc}) = 0$
$\varphi(\overrightarrow{gc}) = 0$
$\varphi(\overrightarrow{bd}) = 1$
$\varphi(\overrightarrow{be}) = 1$
$\varphi(\overrightarrow{bc}) = 2$
$\varphi(\overrightarrow{cd}) = 2$
$\varphi(\overrightarrow{ce}) = 1$
$\varphi(\overrightarrow{dz}) = 4$
$\varphi(\overrightarrow{ez}) = 1$
$\varphi(\overrightarrow{ed}) = 1$

7

Định nghĩa

- Một phép cắt (*cut*) xác định bởi 1 tập hợp con P của V, ký hiệu $(P, \overline{P})$ là tập hợp:

$$(P,\overline{P}) = \{ \overrightarrow{xy} \mid x \in P \text{ và } y \in \overline{P} \}$$

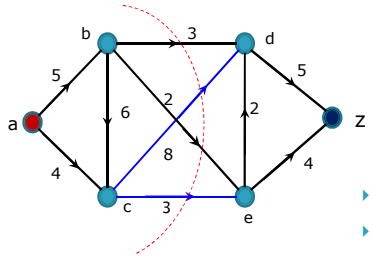
Trong đó $\overline{P} = V \setminus P$

- Phép cắt $(P, \overline{P})$ gọi là 1 phép cắt a-z nếu $a \in P$ và $z \in \overline{P}$
- Trọng số (*capacity*) của một phép cắt được định nghĩa là:

$$c(P,\overline{P}) = \sum_{e \in (P,\overline{P})} c(e)$$

8

Ví dụ:



- P={a,b, c}
- $\overrightarrow{P}$={d, e,z}
- (P,$\overline{P}$)={$\overrightarrow{bd}$,$\overrightarrow{be}$,$\overrightarrow{cd}$,$\overrightarrow{ce}$}
- c(P,$\overline{P}$)=16

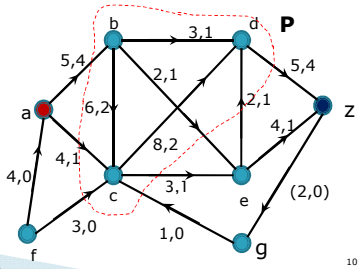
9

Định lý 6.1

Gọi φ là một hàm tải trên mạng G và P ⊂ V\{a,z} thì:

$$\sum_{e\in(P,\overline{P})}\varphi(e) = \sum_{e\in(\overline{P},P)}\varphi(e)$$

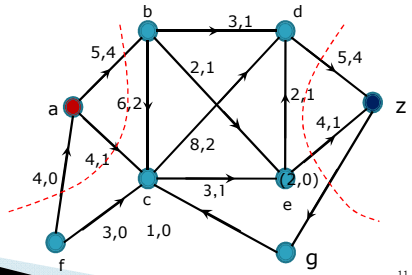
Ví dụ:



10

Định lý 6.2

- Với mọi hàm tải φ trên mạng G, lượng tải khỏi a bằng lượng tải vào z, nghĩa là: $\sum_{e\in Out(a)}\varphi(e) = \sum_{e\in In(z)}\varphi(e)$



11

Chứng minh định lý 6.2

- Đặt P = V \ {a,z}, khi đó:

$$\sum_{e\in Out(a)}\varphi(e) = \sum_{e\in(\overline{P},P)}\varphi(e) = \sum_{e\in(P,\overline{P})}\varphi(e) = \sum_{e\in In(z)}\varphi(e)$$

12

Định lý 6.3

- Với mọi hàm tải φ và với mọi phép cắt a-z trong mạng G, ta có:

$$|\varphi| \le c(P, \overline{P})$$

13

Chứng minh định lý 6.3

- Thêm vào G một đỉnh a_0 và cạnh a_0a (hướng từ a_0 đến a), $c(a_0a)=\infty$. Ta được mạng G'. Trong G' đặt $\varphi'(a_0a) = |\varphi|$ và $\varphi'(e) = \varphi(e)$, $\forall e \in E$

Ta có:

$$|\varphi| = |\varphi'| \le \sum_{e\in(\overline{P}\cup\{a_0\},P)} \varphi'(e) = \sum_{e\in(P,\overline{P}\cup\{a_0\})} \varphi'(e)$$

$$= \sum_{e\in(P,\overline{P})} \varphi(e) \le \sum_{e\in(P,\overline{P})} c(e) = c(P,\overline{P})$$

14

Hệ quả

- Với mọi hàm tải φ và mọi phép cắt a-z trong mạng G. $|\varphi| = c(P, \overline{P})$ nếu và chỉ nếu thỏa 2 điều kiện:

(i) $\forall e \in (\overline{P}, P),\ \varphi(e) = 0$

(ii) $\forall e \in (P, \overline{P}),\ \varphi(e) = c(e)$

- Khi $|\varphi| = c(P, \overline{P})$ thì φ là hàm tải có tải trọng lớn nhất và $(P, \overline{P})$ là phép cắt a-z có trọng số nhỏ nhất

15

Định nghĩa:

- Cho một mạng G, đỉnh phát a và đỉnh thu z, với một phép căt a-z $(P, \overline{P})$
- Một chuyền a-z K là một đường đi vô hướng nối a với z
- Ký hiệu s(e) = c(e)-φ(e) gọi là độ lệch tải của e
- Ta định nghĩa:

$$\varphi_K(e) = \begin{cases} 0 & : e \notin K \\ 1 & : e \in K \text{ và có hướng từ a đến z} \\ -1 & : e \in K \text{ và e có hướng từ z đến a} \end{cases}$$

16

Thuật toán Ford-Fulkerson (Tìm một phép cắt a-z tối thiểu)

Input: Mạng G, đỉnh phát a và đỉnh thu z
Output: Tập P của phép cắt a-z tối thiểu $(P, \overline{P})$

Bắt đầu bằng 1 hàm tải φ bất kỳ trên G

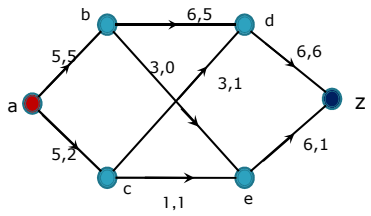
1. Đánh dấu mọi đỉnh đều chưa xét, gán nhãn cho a là $(-, \Delta(a))$ với $\Delta(a)=\infty$. Đặt $p_0=a$.
2. Xét p_0.
   a. Cạnh $e=\overrightarrow{p_0q}$ với q chưa có nhãn và $s(e)>0$ thì gán nhãn cho q là $(p_0^+, \min(\Delta(p_0), s(e)))$
   b. Cạnh $e=\overrightarrow{qp_0}$ với q chưa có nhãn và $\varphi(e)>0$ thì gán nhãn cho q là $(p_0^-, \min(\Delta(p_0), \varphi(e)))$
3. Nếu đỉnh z đã được gán nhãn → 4, ngược lại → 5.
4. Xác định một dây chuyền (vô hướng) từ a đến z dựa vào thành phần thứ 1 của nhãn. Cập nhật lại φ như sau: $\varphi(e) = \varphi(e) + \Delta(z) \times \varphi_K(e)$. Về bước 1.
5. Tìm 1 đỉnh p đã có nhãn nhưng chưa xét. Nếu tồn tại p, đặt $p_0 = p$, → bước 2. Ngược lại dừng.

Thuật toán Ford-Fulkerson

- Sau khi thuật toán kết thúc. P là tập hợp các đỉnh đã có nhãn và đã xét.

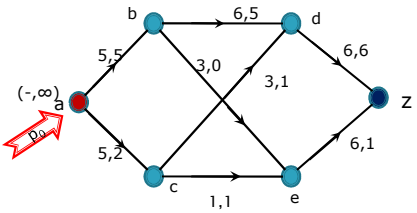
Ví dụ: Tìm một hàm tại cực đại trên mạng G

➢G với hàm tải ban đầu:



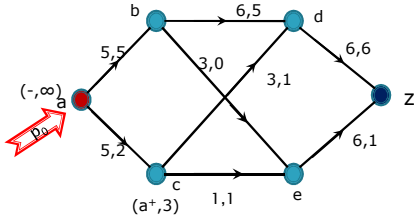
☞Lặp lần 1:

➢Gán nhãn cho đỉnh a là (-,Δ(a)), với Δ(a)=∞
➢Đặt p_0=a



21

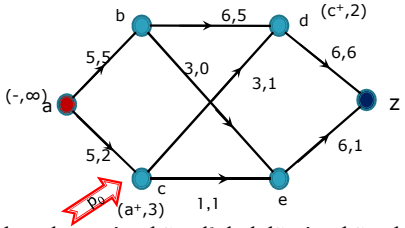
Xét các đỉnh kề với p_0:
➢Cạnh $e_1=(\overrightarrow{a,b})$ có $s(e_1)=0$ nên không xét
➢Cạnh $e_2=(\overrightarrow{a,c})$ có $s(e_2)=3>0$ nên gán nhãn cho đỉnh c là: **$(a^+,\min\{\Delta(p_0),s(e_2)\})=(a^+,3)$**



➢Đỉnh z chưa được gán nhãn, đỉnh c đã gán nhãn nhưng chưa xét, đặt p_0=c

22

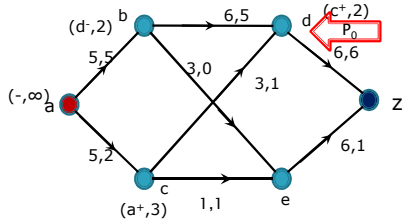
Xét các đỉnh kề với p_0:
➢Cạnh $e_3=(\overrightarrow{c,e})$ có $s(e_3)=0$ nên không xét
➢Cạnh $e_4=(\overrightarrow{c,d})$ có $s(e_4)=2>0$ nên gán nhãn cho đỉnh d là: **$(c^+,\min\{\Delta(p_0),s(e_4)\})=(c^+,2)$**



➢Đỉnh z chưa được gán nhãn, đỉnh d đã gán nhãn nhưng chưa xét, đặt p_0=d

23

p_0=d:
➢Cạnh $e_5=(\overrightarrow{d,z})$ có $s(e_5)=0$ nên không xét
➢Cạnh $e_6=(\overrightarrow{b,d})$ có $\varphi(e_6)=6>0$ nên gán nhãn cho đỉnh b là: **$(d^-,\min\{\Delta(p_0),\varphi(e_6)\})=(d^-,2)$**

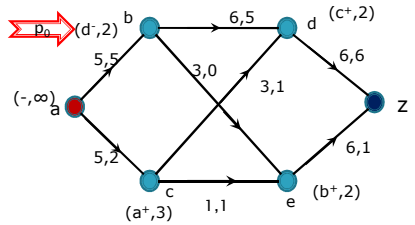


➢Đỉnh z chưa được gán nhãn, đỉnh b đã gán nhãn nhưng chưa xét, đặt p_0=b

24

p_0=b:

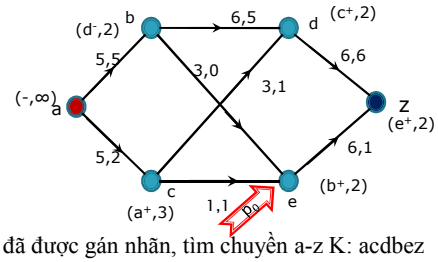
➢Cạnh $e_7=(\overrightarrow{b,e})$ có $s(e_7)=3>0$ nên gán nhãn cho đỉnh e là: **$(b^+,\min\{\Delta(p_0), s(e_7)\}) = (b^+,2)$**



➢Đỉnh z chưa được gán nhãn, đỉnh e đã gán nhãn nhưng chưa xét, đặt p_0=e

p_0=e:

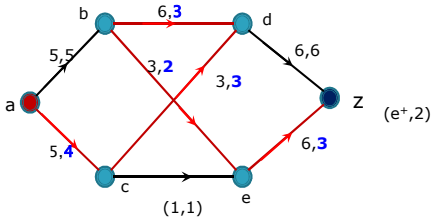
➢Cạnh $e_8=(\overrightarrow{e,z})$ có $s(e_8)=5>0$ nên gán nhãn cho đỉnh z là: **$(e^+,\min\{\Delta(p_0), s(e_8)\}) = (e^+,2)$**



➢Đỉnh z đã được gán nhãn, tìm chuyền a-z K: acdbez

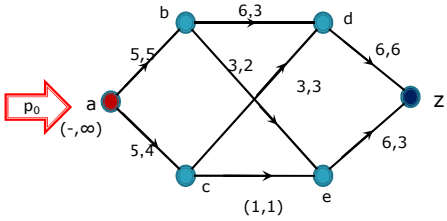
➢Cập nhật lại hàm tải: $\varphi = \varphi + \Delta(z)\varphi_K$

$$\varphi_K(e) = \begin{cases} 0 & : e \notin K \\ 1 & : e \in K \text{ và có hướng từ a đến z} \\ -1 & : e \in K \text{ và e có hướng từ z đến a} \end{cases}$$



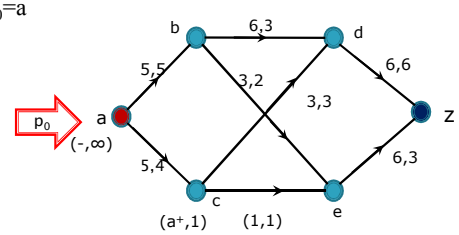
☛Lặp lần 2:

- Gán nhãn cho đỉnh a là (-,Δ(a)), với Δ(a)=∞
- Đặt p_0=a



☛Lặp lần 2:

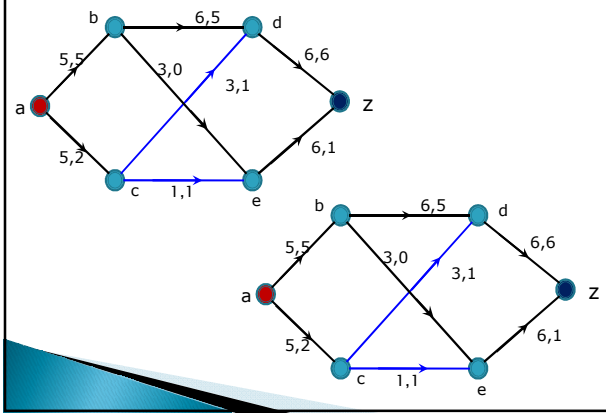
- Gán nhãn cho đỉnh a là (-,Δ(a)), với Δ(a)=∞
- Đặt p_0=a



- Đỉnh z chưa gán nhãn, đỉnh c đã gán nhãn nhưng chưa được xét, đặt p_0=c

29

Ví dụ: Tìm một hàm tại cực đại trên mạng G sau:



30

Định lý 6.4

- Khi kết thúc thuật toán Ford-Fulkerson thì φ là 1 hàm tải tối đại và $(P,\overline{P})$ là 1 phép cắt a-z tối tiểu.

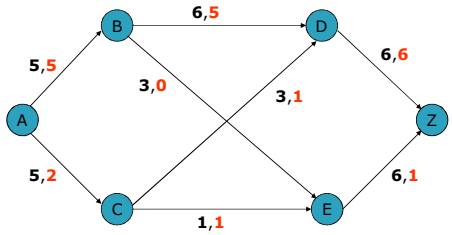
31

Định lý 6.5

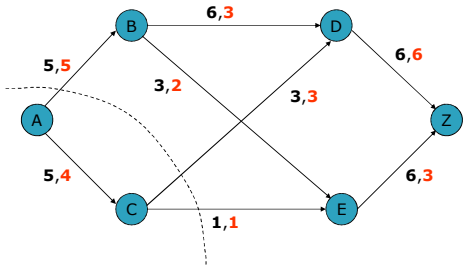
- Trong một mạng G, tải trọng của 1 hàm tải tối đại bằng trọng số của một phép cắt a-z tối tiểu.

32

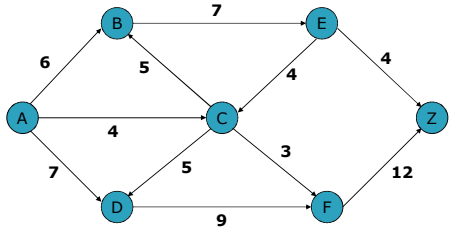
Một số ví dụ khác



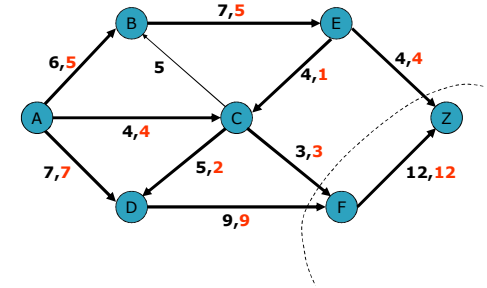
33



34



35



36

Bài toán ghép cặp

- Cho một đồ thị lưỡng phân G = (X,Y,E) với X là tập hợp các đỉnh trái và Y là tập hợp các đỉnh phải của G. Một bộ ghép (matching) của G là một tập hợp các cạnh của G đôi một không có đỉnh chung. Bài toán cặp ghép (matching problem) của G là tìm một bộ ghép tối đại (có số lượng các cạnh là lớn nhất) của G.

37

Một số khái niệm

- Xét 1 bộ ghép M của G. Khi đó:
  - Các đỉnh trong M được gọi là các đỉnh đã được ghép.
  - Một đường pha (*alternating path*) là một đường trong G bắt đầu bằng 1 đỉnh chưa ghép thuộc X và các cạnh lần lượt là thuộc rồi không thuộc M.
  - Một đường mở (*augmenting path*) là 1 đường pha kết thúc bằng một đỉnh chưa ghép thuộc Y.

38

Một số khái niệm (tt)

- Từ 1 đỉnh u chưa ghép thuộc X, ta có thể xây dựng 1 cây pha (*alternating tree*) gốc u gồm tất cả các đường pha bắt đầu từ u.
- Một cây pha chứa ít nhất 1 đường mở được gọi là 1 cây mở (*augmenting tree*). Ngược lại sẽ được gọi là một cây đóng (*Hungarian tree*), gốc u của cây đóng này gọi là đỉnh đóng (*Hungarian acorn*).

39

Thuật toán xác định bộ ghép tối đại M – Thuật toán Hungarian

1. Đặt mọi đỉnh thuộc X là chưa kiểm tra. Đặt M=∅.
2. Nếu mọi đỉnh thuộc X chưa ghép đều đã kiểm tra thì dừng. Nếu không, chọn một đỉnh u∈X chưa ghép và chưa kiểm tra để xây dựng 1 cây pha gốc u.
3. Nếu cây pha này là cây mở thì → bước 4. Nếu không, đánh dấu u là đã kiểm tra → bước 2.
4. Mở rộng M bằng cây mở như sau: Trên đường mở, loại bỏ các cạnh trong M và thêm vào các cạnh ngoài M. Đánh dấu mọi đỉnh thuộc X là chưa kiểm tra. Quay về bước 2.

40

Định lý 6.5

- Bộ ghép nhận được sau khi áp dụng thuật toán Hungarian vào đồ thị lưỡng phân G là tối đại.

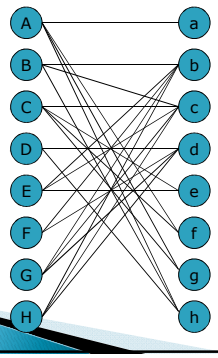
41

Định lý 6.6 (định lý Hall)

- Một bộ ghép M của đồ thị lưỡng phân G=(X,Y,E) được gọi là X-đầy đủ (*X-complete matching*) nếu M chứa mọi đỉnh của X. Với $A \subset X$, đặt $\Gamma(A)$ là tập hợp các đỉnh $y \in Y$ kề với một đỉnh $x \in A$. Khi này, G có 1 bộ ghép X-đầy đủ nếu và chỉ nếu $\forall A \subset X$, $|\Gamma(A)| \geq |A|$.

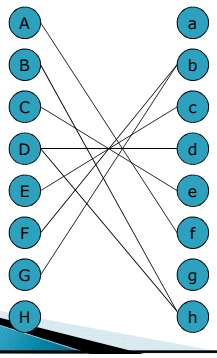
42

Thí dụ



43

Thí dụ (tt)



44