CÁC THUẬT TOÁN TRÊN ĐỒ THỊ

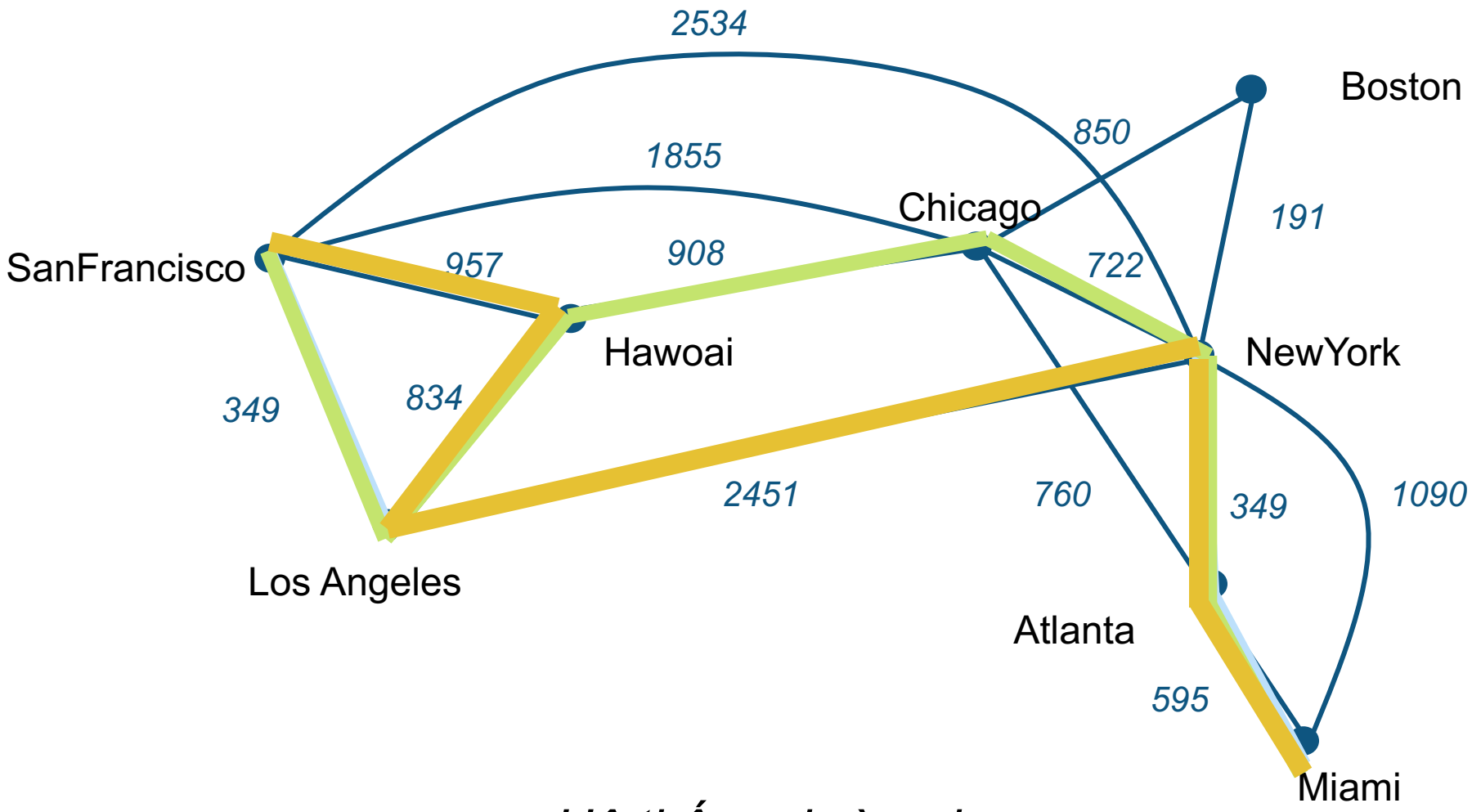
Giảng Viên: Ths Nguyễn Thị Quỳnh Như

Email: ntqnhu@hou.edu.vn

NỘI DUNG BÀI HỌC

- Đường đi ngắn nhất
- Cây và cây khung

I. Đường đi ngắn nhất



Hệ thống đường bay

Tìm đường đi ngắn nhất từ SanFrancisco đến Miami

1. Độ dài đường đi

- Độ dài đường đi từ đỉnh v_0 đến đỉnh v_p bằng tổng các trọng số của các cung(cạnh) trên đường đi đó và được biểu diễn bởi

$$d(v_0, v_p) = \sum_{i=1}^{p} w(v_0, v_i)$$

2. Phát biểu bài toán

- Cho đồ thị có trọng số G=(V,E), hãy tìm một đường đi ngắn nhất xuất phát từ đỉnh s thuộc V, đến đỉnh đích f thuộc V.
- Ký hiệu:
  - d(s,f), gọi là khoảng cách từ s đến f
  - d(s,f)=∞ nếu s không đến được f.
- Gọi c[u,v] là trọng số của cạnh [u,v].
  - c[v,v]=0 với mọi v thuộc V
  - c[u,v]= ∞ nếu u,v không thuộc E.

Phát biểu bài toán (2)

- Nhận xét: Để tìm đường đi từ s tới f, ta luôn có thể tìm được đỉnh f1khác f sao cho: d(s,f)=d(s,f1)+c[f1,f]
  - Với f1 là đỉnh liền trước f trong đường đi ngắn nhất.
  - Nếu s trùng với f1 thì từ s có thể đến f trực tiếp
  - Ngược lại: bài toán trở thành tìm đường đi ngắn nhất từ s-> f1
  - Lúc này ta lại tìm được đỉnh f2 khác mà: d(s,f1)=d(s,f2)+c[f2,f1]

3. Tìm đường đi ngắn nhất

- Trường hợp đồ thị không có chu trình âm, ta có 2 thuật toán tìm đường đi ngắn nhất sau:
  - Thuật toán Ford-BellMan
  - Thuật toán Dijkstra

4. Thuật toán Ford-BellMan

- Xuất phát từ đỉnh s.
  - Gọi d[v] là khoảng cách từ s->v
  - Khởi tạo: d[s]=0; d[v]= ∞ nếu v ≠ s
- Tối ưu hóa d[v]:
  - Xét mọi cặp đỉnh u,v của đồ thị, nếu có 1 cặp đỉnh u,v mà d[v]>d[u]+c[u,v] thì d[v]=d[u]+c[u,v]
  - Nghĩa là nếu độ dài đường đi từ s->v lớn hơn độ dài đường đi từ s->u cộng với độ dài đường đi từ u->v thì đường đi từ s-v sẽ là s->u->v.
  - Thuật toán kết thúc khi không thể tìm ra phương án nào tối ưu hơn nghĩa là d[v] không thay đổi được nữa
- Lưu ý: Thuật toán áp dụng với đồ thị không có chu trình âm

Cài đặt thuật toán

$for(\forall v \in V) \quad d[v] = +\infty;$

$d[s] = 0;$

do

$\{stop = true;$

$for(\forall u \in V)$

$for(\forall v \in V : (u,v) \in E)$

$if\ (d[v] > d[u] \quad + c[u,v] \quad)$

$\quad \{$

$d[v] = d[u] \quad + c[u,v];$

$truoc[v] = u; // luu \quad duong \quad di$

$stop = False;$

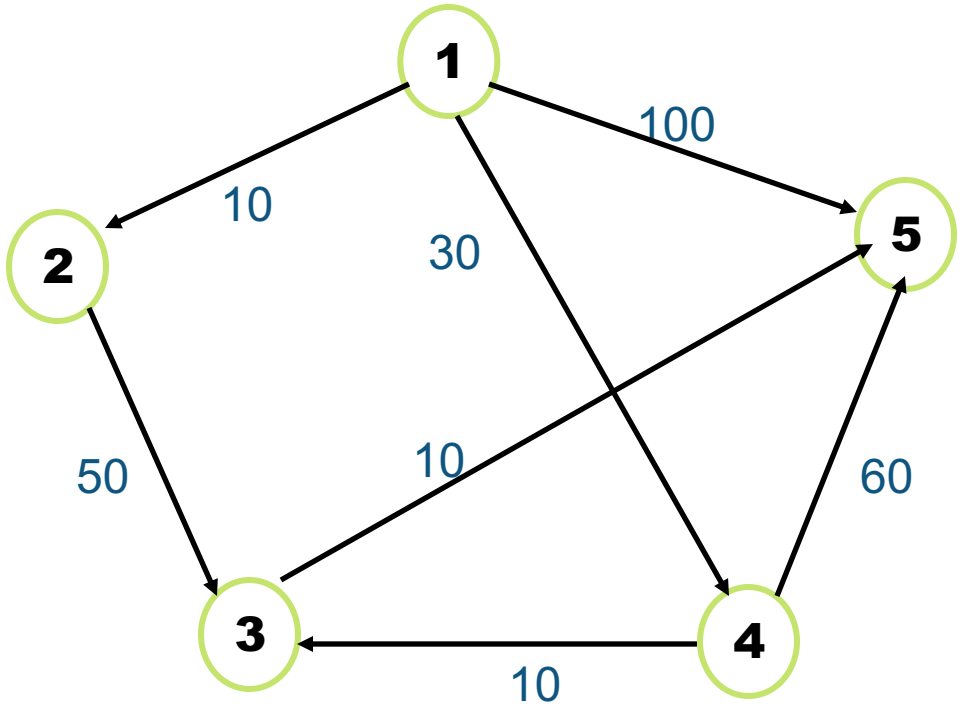
$\quad \}$

$\}while \quad (stop);$

Nhận xét 1:

- Đồ thị đang xét là đồ thị không có chu trình âm.
- Ở bước khởi tạo: d[v]= độ dài đường đi ngắn nhất từ s->v qua 0 cạnh
- Lần lặp thứ i:
  - d[v]= độ dài đường đi ngắn nhất từ s->v qua không quá i-1 cạnh.
- Vậy: đường đi ngắn nhất từ s-> v qua nhiều nhất là n-1 cạnh. Số lần lặp của thuật toán tối ưu là không quá n-1 lần

Mô tả thuật toán Ford-Bellman bằng bảng



u	v	C[u,v]
1	2	10
1	4	30
1	5	100
2	3	50
3	5	10
4	3	10
4	5	60

Tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 5.

Cách xây dựng bảng

- Bảng bao gồm: N+1 cột, tối đa N+2 dòng
  - Cột 1: ghi thứ tự các bước thực hiện
  - Các cột còn lại ghi d[i] với i=1,n là các đỉnh của đồ thị
  - Dòng 1: ghi tiêu đề cột
  - Dòng 2: Ghi các giá trị khởi tạo
  - Các dòng tiếp theo dùng để lưu giá trị các bước thực hiện
- Dùng mảng "truoc" để lưu vết đường đi

Xây dựng bảng tìm đường đi ngắn nhất

Số lần lặp	d[1]	d[2]	d[3]	d[4]	d[5]
Khởi tạo	**0**	∞	∞	∞	∞
1	0	10	40	30	70
2	0	10	40	30	50
3	0	10	40	30	50
Dừng vì không còn nhãn nào có thể tối ưu thêm					

u	v	C[u,v]
1	2	10
1	4	30
1	5	100
2	3	50
3	5	10
4	3	10
4	5	60

- Mỗi lần tối ưu nhãn cho đỉnh u nào đó phải thay đổi giá trị lưu vết của u trong mảng Truoc

Nhận xét 2

- Tư tưởng thuật toán FordBellman:
  - Với đỉnh v thuộc V, d[v]: độ dài đường đi ngắn nhất từ s->v
  - Khởi tạo d[s]=0, d[v]=∞ với mọi v ≠s.
  - Tiến hành tối ưu dần các nhãn d[v] theo công thức d[v]=min(d[v], d[u]+c[u,v]), với mọi u,v thuộc V
- Với tư tưởng trên, nhãn đỉnh v được sửa nhờ vào đỉnh u, nhưng u cũng lại được tối ưu nên phải quay lại sửa nhãn đỉnh v. Việc này có thể lặp lại nhiều lần.
- Để khắc phục điều này, ta không xét mọi cặp đỉnh (u,v) để dùng u sửa nhãn cho v mà chỉ chọn đỉnh u nào mà nhãn của d[u] không còn tối ưu được nữa. Đây chính là ý tưởng của thuật toán Dijkstra.

5. Thuật toán Dijkstra

- **Bước 1:** Khởi tạo
  - Gọi d[v]: đường đi ngắn nhất từ s->v
  - d[s]=0; d[v]=+∞ với mọi v≠s
  - Free[v]: nhãn của đỉnh v.
    - Free[v]=true: nhãn tự do → có thể tối ưu thêm
    - Free[v]=False: nhãn cố định → không thể tối ưu thêm
- **Bước 2:** Lặp
  - Bước 2.1. Cố định nhãn → chọn trong các đỉnh có nhãn tự do, lấy ra đỉnh u có d[u] nhỏ nhất, cố định nhãn đỉnh u.
  - Bước 2.2. Sửa nhãn → Dùng đỉnh u, xét tất cả các đỉnh v và sửa lại d[v] theo công thức: d[v]=min(d[v],d[u]+c[u,v]);
  - Lưu vết đường đi.
  - Bước lặp kết thúc khi đỉnh đích f đã được cố định nhãn (tìm được đường đi) hoặc tại thao tác cố định nhãn, tất cả các đỉnh tự do đều có nhãn là +∞ (không tìm được đường đi)
- **Bước 3**: Dựa vào mảng lưu vết, thông báo kết quả.

Cài đặt thuật toán

$for(\forall v \in V) \quad d[v] = +\infty;$

$d[s] = 0;$

do

$\{u = \min(d[v] \quad | \forall v \in V, Free[v] = true);$

$if\ (u == f) \,||\, (u == 0) \quad break;$

$// thuat\_toan\_kthuc\_khi\_u\_la\_f\_hoac\_tat\_ca\_cac\_nhan\_la + \infty$

$Free[u] = false; // co\_dinh\_nhan\_dinh\_u$

$for(\forall v \in V : (u,v) \in E)$

$if\ (Free[v] \&\& (d[v] > d[u] \quad + c[u,v] \quad))$

$\{$

$d[v] = d[u] \quad + c[u,v];$

$truoc[v] = u; // luu \quad duong \quad di$

$\}$

$\}while \quad (false);$

Cách xây dựng bảng

- Bảng bao gồm: N+3 cột, tối đa N+2 dòng
  - Cột 1: ghi thứ tự các bước thực hiện
  - Cột 2: Ghi tập S là tập các đỉnh được cố định nhãn
  - Cột 3: Ghi đỉnh u với u là đỉnh thuộc V-S sao cho d[u] nhỏ nhất
  - Các cột còn lại ghi d[i] với i=1,n là các đỉnh của đồ thị
  - Dòng 1: ghi tiêu đề cột
  - Dòng 2: Ghi các giá trị khởi tạo
  - Các dòng tiếp theo dùng để lưu giá trị các bước thực hiện
  - Đỉnh nào cố định được nhãn thì cho vào tập S.
- Dùng mảng "truoc" để lưu vết đường đi

Xây dựng bảng tìm đường đi ngắn nhất

Số lần lặp	S	u	d[1]	d[2]	d[3]	d[4]	d[5]
Khởi tạo	{}	-	**0**	∞	∞	∞	∞
1	{1}	1	0	**10**	∞	30	100
2	{1,2}	2	0	10	60	**30**	100
3	{1,2,4}	4	0	10	**40**	30	90
4	{1,2,3,4}	3	0	10	40	30	**50**
5	{1,2,3,4,5}	5	Cố định được nhãn f=5, kết thúc				

- Mảng Truoc có giá trị như sau:
- Truoc 1 2 3 4 5

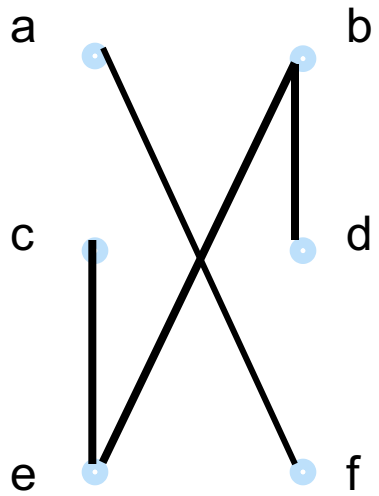
  -1 1 4 1 3

II. Cây và cây khung

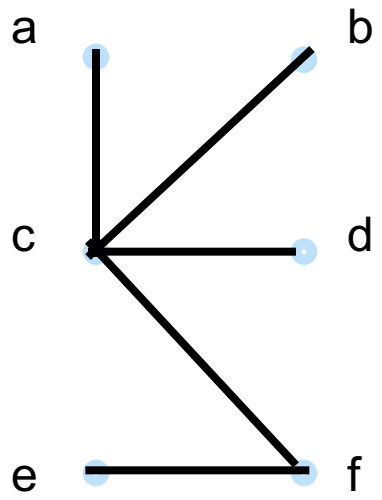
- Định nghĩa cây
- Định nghĩa cây khung
- Cây khung cực tiểu
- Các thuật toán tìm cây khung cực tiểu
  - Thuật toán Prim
  - Thuật toán Kruskal

1. Định nghĩa cây

- Định nghĩa 1: Cây là đồ thị vô hướng liên thông không có chu trình.
- Rừng là đồ thị gồm nhiều thành phần liên thông tách rời nhau và mỗi thành phần liên thông của nó lại là một cây.
- Ví dụ:



G1



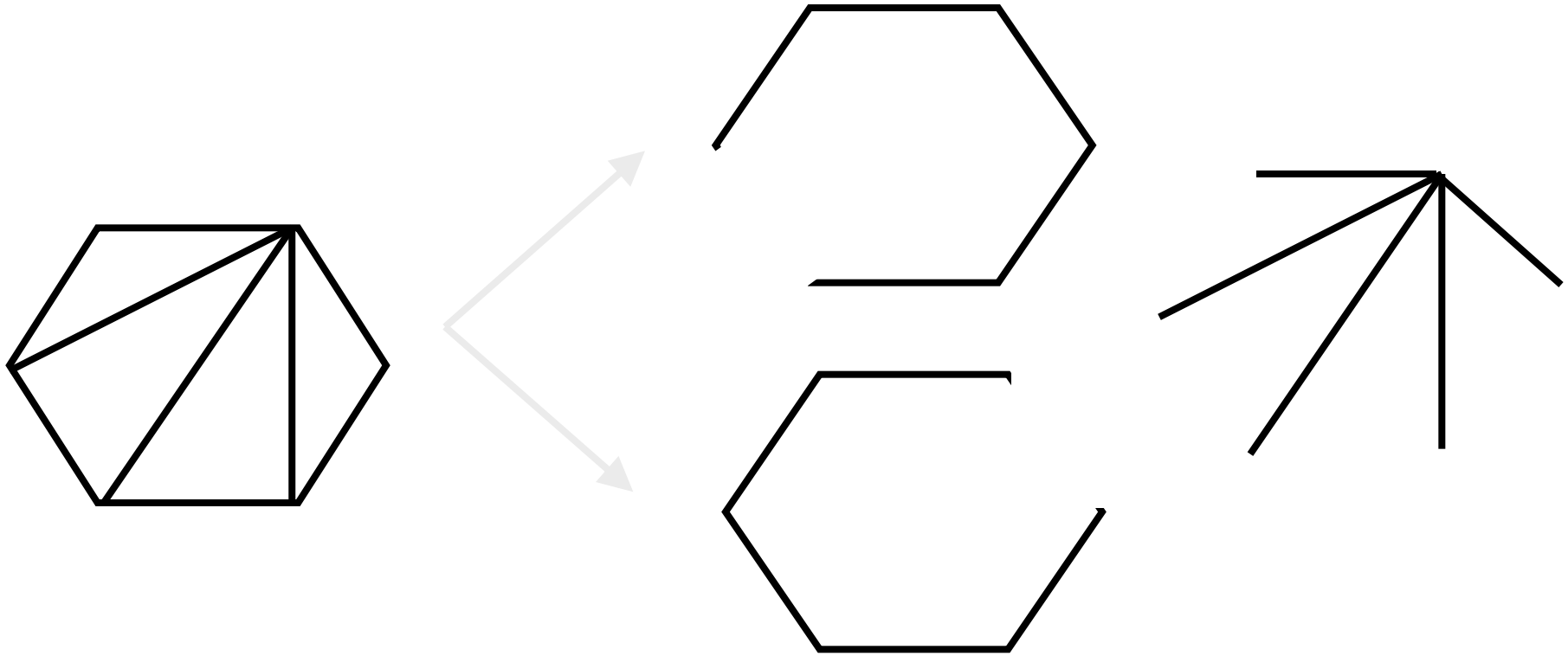
G2

2. Định lý:

- Định lý 1: Giả sử G=(V,E) là đồ thị vô hướng n đỉnh. Khi đó các mệnh đề sau là tương đương
- T là cây
- T không có chu trình và có n-1 cạnh
- T liên thông và mỗi cạnh đều là cầu (huỷ bất kỳ cạnh nào cũng làm mất tính liên thông)
- Giữa hai đỉnh bất kỳ của T luôn tồn tại đường đi sơ cấp(đường đi đơn) duy nhất nối hai đỉnh này
- T không có chu trình và nếu thêm một cạnh mới nối 2 đỉnh bất kỳ của T thì T sẽ tạo thành chu trình
- T liên thông và có n-1 cạnh

3. Định nghĩa cây khung

- Định nghĩa 2: Cho G là một đơn đồ thị. Một cây được gọi là cây khung của G nếu nó là một đồ thị con của G và chứa tất cả các đỉnh của G
- Ví dụ:



4. Cây khung nhỏ nhất/cực tiểu

- Định nghĩa: Cây khung cực tiểu trong đồ thị liên thông có trọng số là một cây khung có tổng trọng số trên các cạnh của cây khung đó là nhỏ nhất
- Các thuật toán tìm cây khung cực tiểu
  - Prim
  - Kruskal

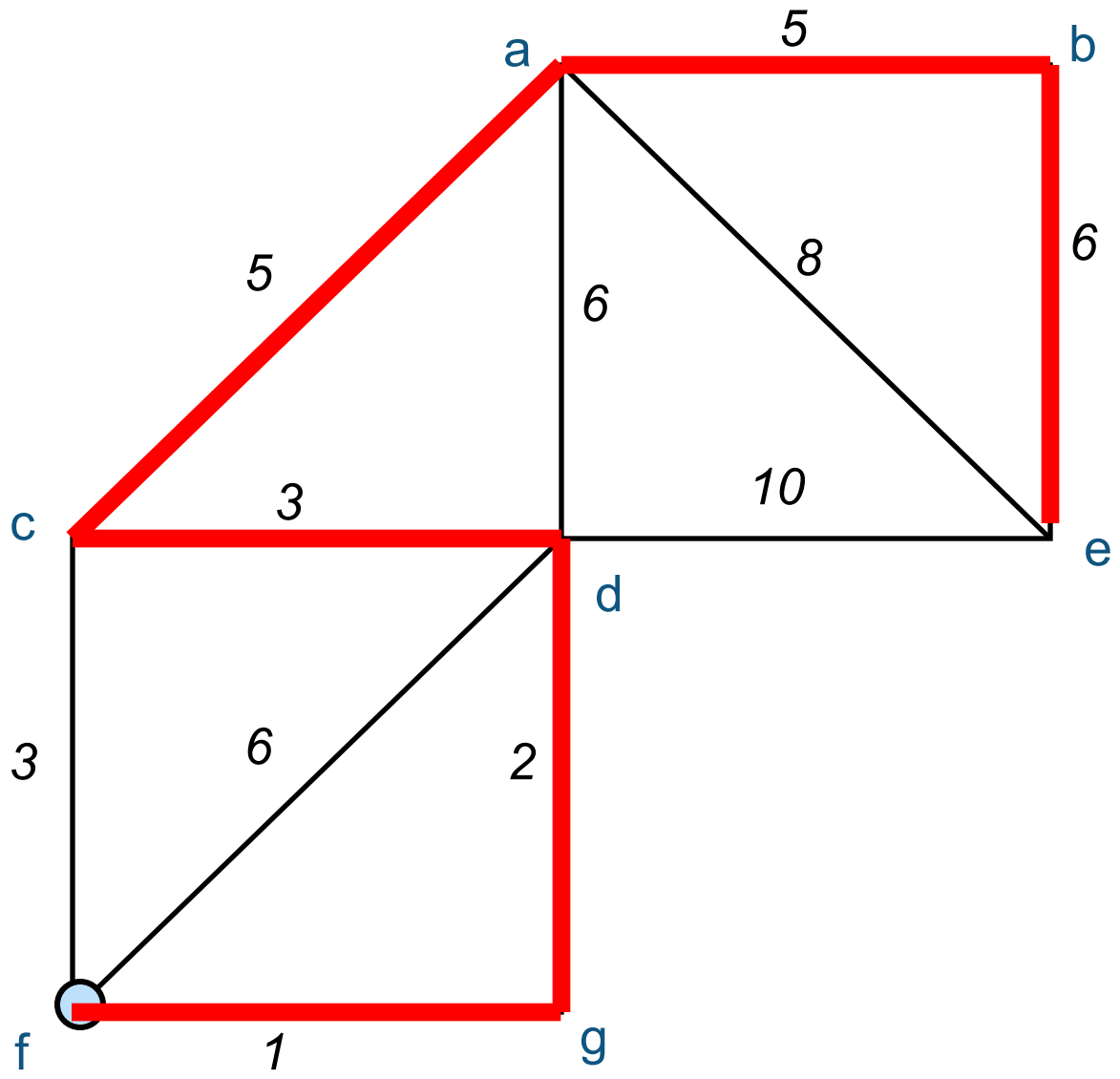
a. Thuật toán Prim:

- Ý tưởng:
  - Chọn một đỉnh bất kỳ đặt vào cây khung
  - Sau đó lần lượt ghép vào cây các cạnh có trọng số nhỏ nhất chưa xét mà liên thuộc với đỉnh của cây đã xét sao cho không tạo thành chu trình.

Thuật toán Prim

- Gọi Tk :tập chứa các cạnh tạo thành cây khung ở bước lặp thứ k
- Gọi Sk :tập các đỉnh của cây khung tại bước lặp thứ k
- Thuật toán thực hiện các bước lặp sau:
  - bước 0: Chọn đỉnh bất kỳ chưa xét bổ sung vào tập đỉnh S0 của cây khung
  - bước 1: chọn cạnh có trọng số nhỏ nhất liên thuộc với đỉnh đã xét đưa vào tập T1 (tập chứa cây khung)
  - bước k: xây dựng tập Tk dựa trên tập Tk-1 như sau:
    - Chọn cạnh (u,v) có giá trị trọng số nhỏ nhất chưa được xét sao cho (u,v) liên thuộc với đỉnh bất kỳ trong Sk-1 mà không tạo thành chu trình => bổ sung cạnh (u,v) vào tập Tk và đỉnh u,v vào tập Sk.
    - Quá trình lặp liên tiếp khi k = n –1 ⇔ Chọn được n-1 cạnh vào cây khung
- Chú ý: Trong một đồ thị có thể có nhiều cây khung nhỏ nhất khác nhau

▪Ví dụ: Tìm cây khung nhỏ nhất của đồ thị sau



b. Thuật toán Kruskal

- Ý tưởng:
  - Chọn cạnh có trọng số nhỏ nhất của đồ thị
  - Lần lượt ghép thêm vào các cạnh có trọng số tối thiểu trong số các cạnh chưa xét sao cho không tạo thành chu trình với các cạnh đã chọn

Thuật toán Kruskal

- Thuật toán
  - Bước 0: sắp xếp các cạnh theo thứ tự tăng dần của trọng số
  - Bước 1: chọn cạnh đầu tiên có trọng số nhỏ nhất trong số các cạnh chưa xét và bổ sung cạnh đó vào cây khung
  - Bước k: xây dựng tập Tk bằng cách chọn ra cạnh chưa xét có trọng số nhỏ nhất theo thứ tự mà không tạo thành chu trình => bổ sung cạnh vào tập Tk chứa cây khung
  - Quá trình thực hiện lặp liên tiếp cho đến khi bổ sung n-1 cạnh vào cây khung

5. Đồ thị Euler và đồ thị Hamilton

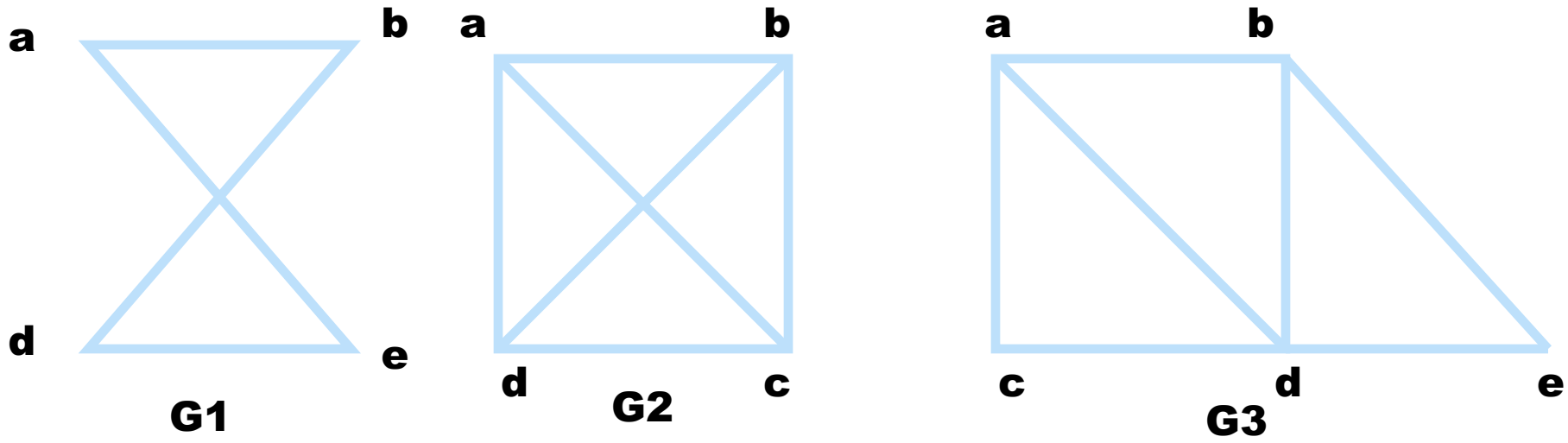
- Định nghĩa đồ thị Euler
- Định nghĩa đồ thị Hamilton
- Tìm chu trình Euler và Hamilton trong đồ thị

a. Đồ thị Euler:

- Định nghĩa 1:
  - Chu trình đơn trong đồ thị G đi qua tất cả các cạnh của đồ thị, mỗi cạnh đúng một lần được gọi là chu trình Euler.
  - Đường đi đơn đi qua tất cả các cạnh của đồ thị G, mỗi cạnh đúng một lần được gọi là đường đi Euler.
  - Đồ thị được gọi là đồ thị Euler nếu có chu trình Euler và được gọi là đồ thị nửa Euler nếu có đường đi Euler

Ví dụ

- Ví dụ: Các đồ thị sau thuộc loại nào?



Định lý

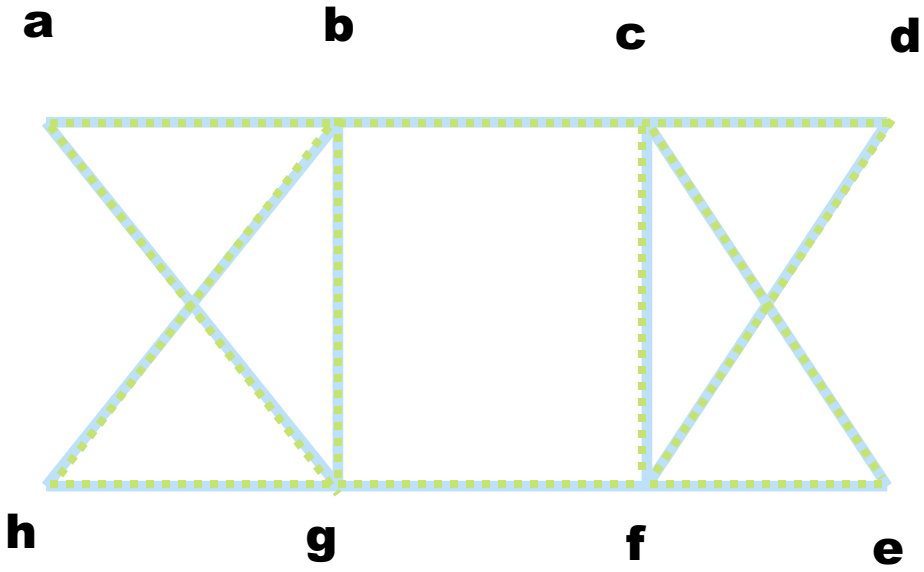
- Định lý 1 (Euler): Đồ thị vô hướng liên thông G là đồ thị Euler khi và chỉ khi mọi đỉnh của đồ thị G đều có bậc chẵn ⇔ điều kiện cần và đủ để đồ thị là đồ thị Euler
  - Bổ đề 1: Nếu bậc của mỗi đỉnh của đồ thị G không nhỏ hơn 2 (>=2) thì đồ thị G chứa chu trình
- Định lý 2: Đồ thị vô hướng liên thông G là nửa Euler khi và chỉ khi nó có đúng 2 đỉnh bậc lẻ.

Thuật toán xác định chu trình Euler

- Xuất phát từ một đỉnh bất kỳ nào đó của đồ thị G, đi theo các cạnh của đồ thị một cách tuỳ ý dựa vào nguyên tắc sau:
  - Nguyên tắc 1: Mỗi khi đi qua một cạnh nào đó thì xoá cạnh đó đi, sau đó xoá đi các đỉnh cô lập nếu có của đồ thị
  - Nguyên tắc 2: Tại mỗi bước đi không bao giờ đi qua một cầu trừ khi không còn cách nào khác để di chuyển

Ví dụ

- Ví dụ: Tìm chu trình Euler trong đồ thị sau



Chu trình Euler của đồ thị:

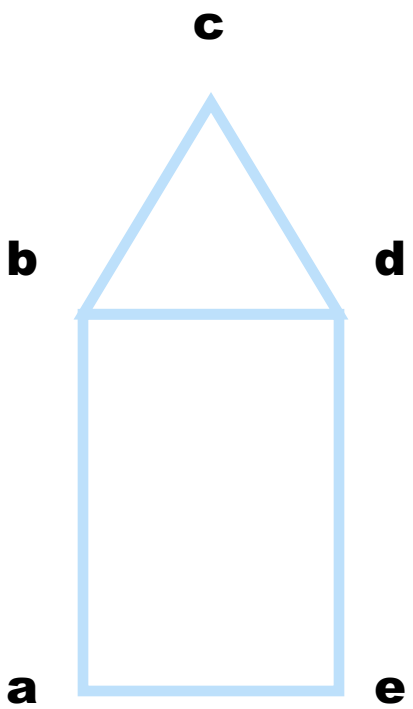
a->b->c->d->f->e->c->f->g->h->b->g->a

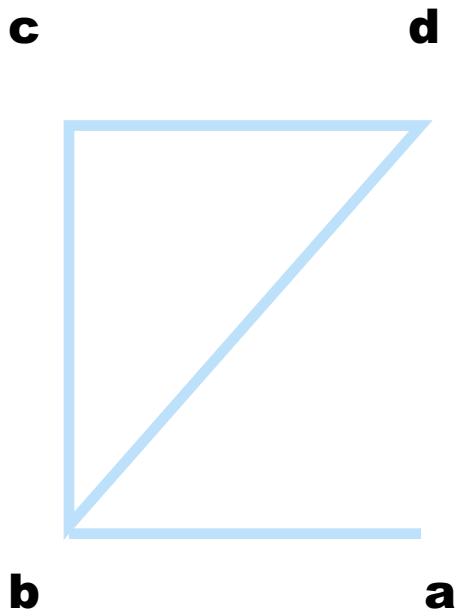
b. Đồ thị Hamilton

- Định nghĩa 2:
  - Đường đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần được gọi là đường đi Hamilton.
  - Chu trình bắt đầu từ một đỉnh v nào đó qua tất cả các đỉnh còn lại mỗi đỉnh đúng một lần rồi quay lại đỉnh v được gọi là chu trình Hamilton.
  - Đồ thị G gọi là đồ thị Hamilton nếu nó chứa chu trình Hamilton, và đồ thị G gọi là nửa Hamilton nếu nó chứa đường đi Hamilton.

Ví dụ

- Ví dụ: đồ thị sau có chu trình hoặc đường đi Hamilton?





Thuật toán tìm chu trình Hamilton

- Cho đồ thị G=(V,E), để tìm chu trình Hamilton cho đồ thị, thực hiện theo 4 nguyên tắc sau:
  - Quy tắc 1: Nếu tồn tại một đỉnh v của đồ thị G có deg(v)≤1 thì đồ thị G không có chu trình Halminton
  - Quy tắc 2: Nếu đỉnh v có bậc là 2, deg(v)=2, thì cả hai cạnh tới v đều phải thuộc chu trình Halminton.
  - Quy tắc 3: Chu trình Hamilton không chứa bất kỳ chu trình con thực sự nào.
  - Quy tắc 4: Trong quá trình xây dựng chu trình Hamilton, sau khi đã lấy 2 cạnh tới một đỉnh v đặt vào chu trình Hamilton rồi thì không thể lấy thêm cạnh nào tới v nữa ⇔ xoá mọi cạnh còn lại tới v.

Bài tập áp dụng

- Viết chương trình thực hiện các công việc sau:
  - Tìm đường đi ngắn nhất từ đỉnh a đến đỉnh z
  - Tìm cây khung nhỏ nhất của đồ thị sau, theo 2 cách (áp dụng thuật toán Kruskal và Prim).
  - Tìm chu trình Hamilton của đồ thị sau