VNU-HUS MAT3500: Toán rời rạc

Lý thuyết đồ thị III

Hoàng Anh Đức

Bộ môn Tin học, Khoa Toán-Cơ-Tin học
Đại học KHTN, ĐHQG Hà Nội
hoanganhduc@hus.edu.vn

Nội dung

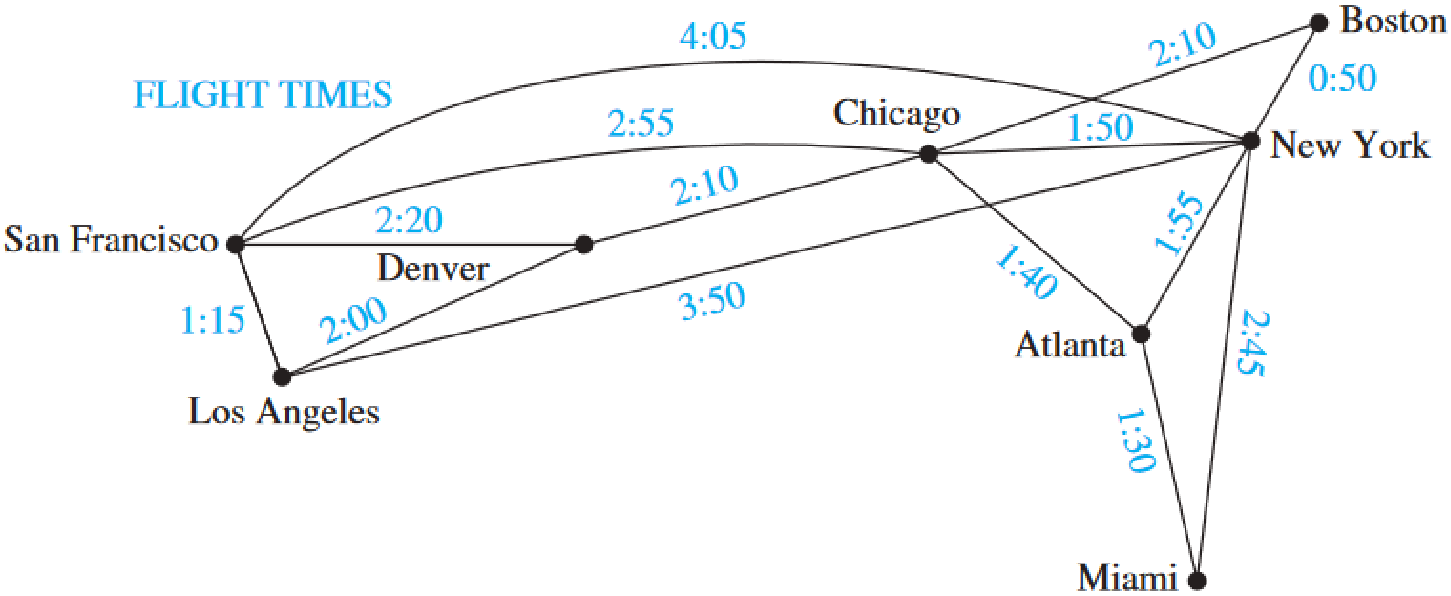
Bài toán đường đi ngắn nhất
- Đồ thị có trọng số
- Thuật toán Dijkstra

Đồ thị phẳng
- Định nghĩa và khái niệm
- Công thức Euler
- Định lý Kuratowski

Tô màu đồ thị

Cây
- Giới thiệu
- Một số tính chất của cây
- Cây có gốc

- Một *đồ thị có trọng số (weighted graph)* $G = (V, E, w)$ gồm tập đỉnh V, tập cạnh E, và một hàm $w : E \to \mathbb{R}$ gán mỗi cạnh (cung) $e \in E$ bởi một số thực $w(e)$ gọi là *trọng số (weight)* của cạnh (cung) e
- Trong bài giảng, chúng ta chỉ xét các đồ thị có *trọng số dương*, nghĩa là, $w : E \to \mathbb{R}^+$



Hình: Đồ thị có trọng số mô tả thời gian của các chuyến bay giữa một số thành phố ở Mỹ (từ [Rosen 2012], Chương 10)

Bài toán đường đi ngắn nhất

Đồ thị có trọng số

Cho $G = (V, E, w)$ là đơn đồ thị có trọng số

- Một đường đi từ u đến v qua các cạnh (cung) $e_1, e_2, \dots, e_n$ có *chiều dài (length)* $c(u, v) = \sum_{i=1}^{n} w(e_i)$
- *Khoảng cách (distance)* giữa hai đỉnh u, v, ký hiệu $d_G(u, v)$, là chiều dài ngắn nhất (nghĩa là, $c(u, v)$ nhỏ nhất) của một đường đi từ u đến v

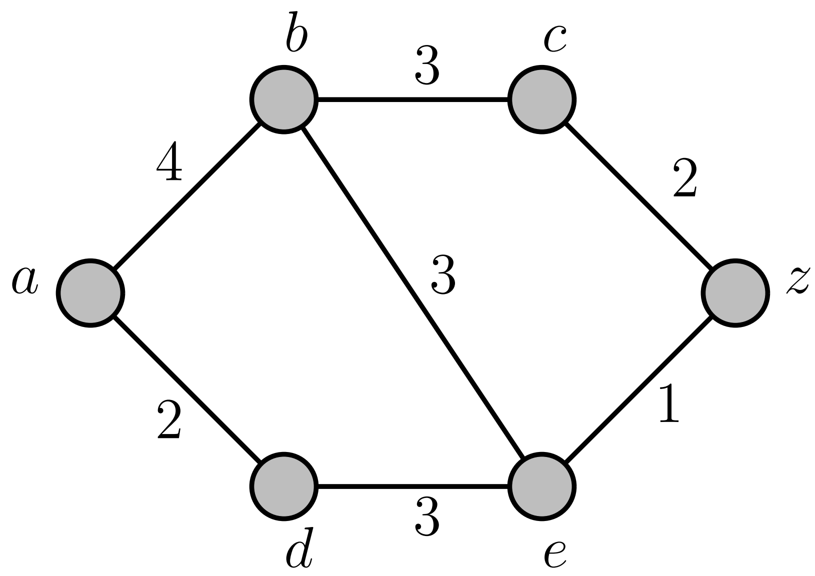
Bài toán đường đi ngắn nhất

- **Input:** Đồ thị $G = (V, E, w)$ trong đó $w : E \to \mathbb{R}^+$, và hai đỉnh $a, z \in V$
- **Output:** Tính khoảng cách $d_G(a, z)$

- Ta tìm hiểu thuật toán Dijkstra giải bài toán đường đi ngắn nhất trong trường hợp *G là đơn đồ thị vô hướng liên thông có trọng số*

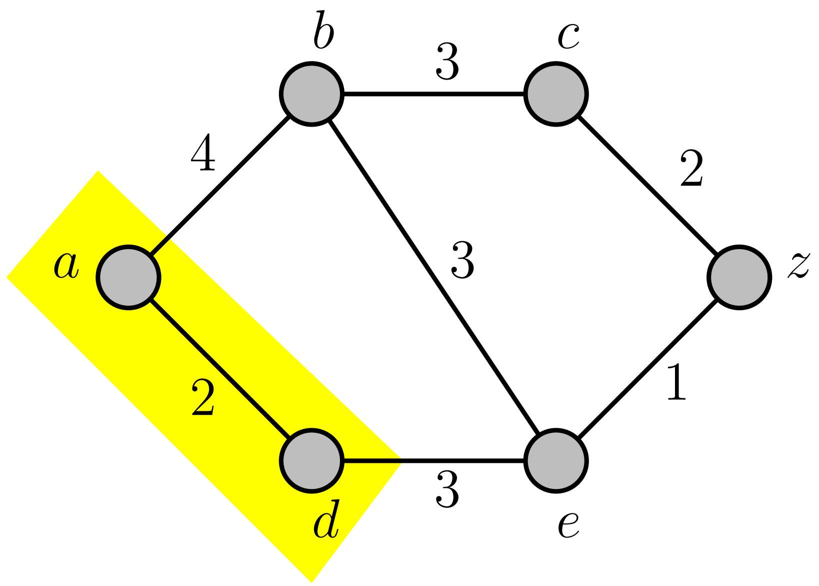
Ví dụ 1

Tìm khoảng cách giữa hai đỉnh a và z trong đồ thị sau

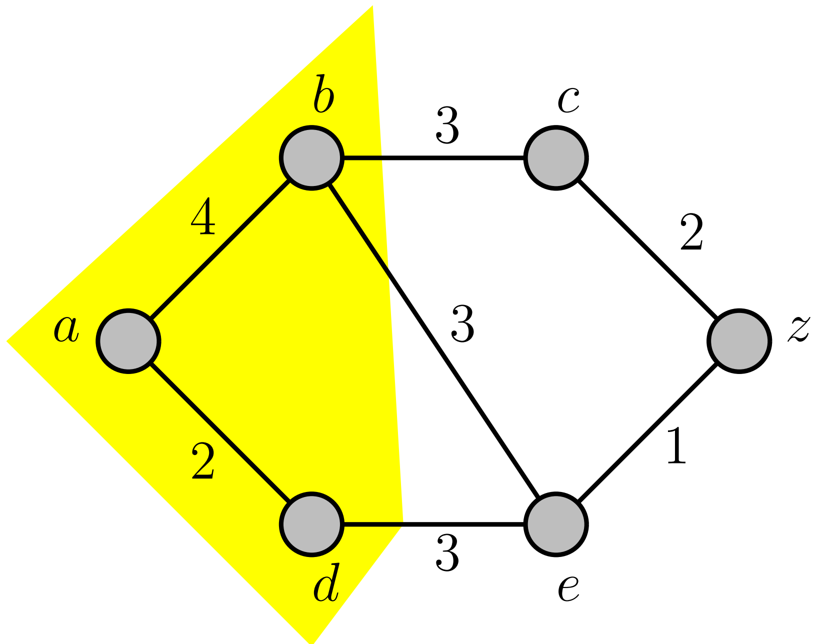


Ý tưởng: Tìm đường đi ngắn nhất từ a tới các đỉnh kế tiếp cho đến khi đạt đến z. Chú ý rằng với các đỉnh u, v, z bất kỳ, *độ dài đường đi ngắn nhất từ u đến v **đi qua đỉnh** z kề với v bằng khoảng cách giữa u và z cộng với trọng số cạnh nối v và z*

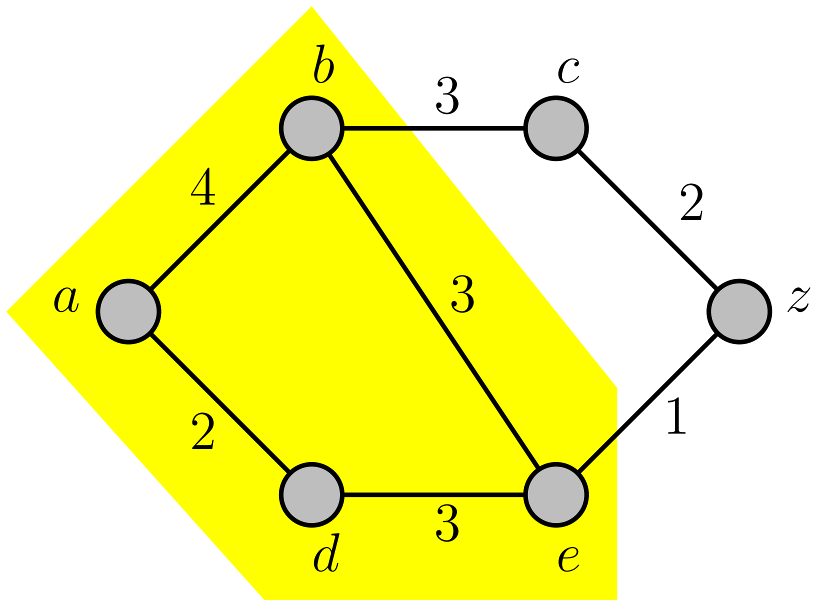
- Đỉnh gần a nhất
  - Xét các cạnh xuất phát từ a
  - $d_G(a, a) + w(a, b) = 4$, $d_G(a, a) + w(a, d) = 2$
  - Chọn d. Ta có $d_G(a, d) = 2$, ứng với đường đi a, d



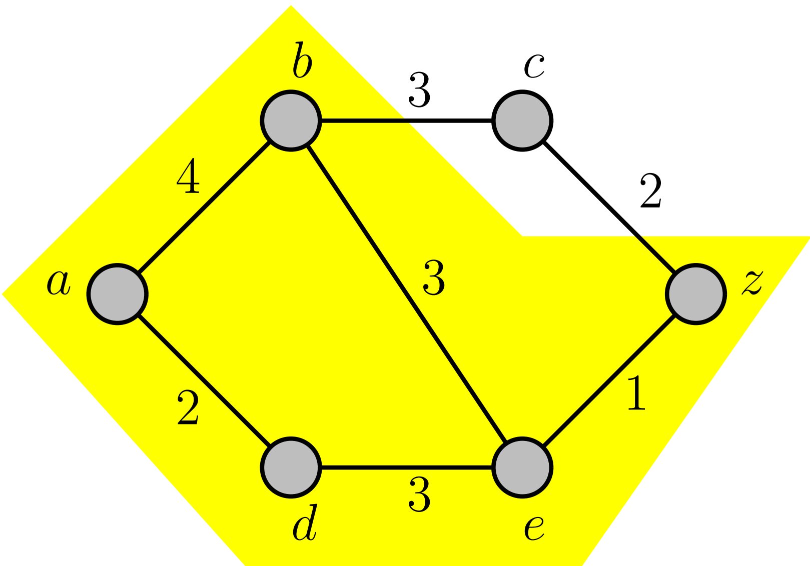
- Đỉnh gần a thứ hai
  - Xét các cạnh xuất phát từ a hoặc d
  - $d_G(a,a) + w(a,b) = 4$, $d_G(a,d) + w(d,e) = 5$
  - Chọn b. Ta có $d_G(a,b) = 4$, ứng với đường đi a, b



- Đỉnh gần a thứ ba
  - Xét các cạnh xuất phát từ b hoặc d
  - $d_G(a,b) + w(b,c) = 7$, $d_G(a,b) + w(b,e) = 7$, $d_G(a,d) + w(d,e) = 5$
  - Chọn e. Ta có $d_G(a,e) = 5$, ứng với đường đi a, d, e (đường đi ngắn nhất từ a đến d hợp với cạnh de)



- Đỉnh gần a thứ tư
  - Xét các cạnh xuất phát từ b hoặc e
  - $d_G(a, b) + w(b, c) = 7$, $d_G(a, e) + c(e, z) = 6$
  - Chọn z. Ta có $d_G(a, z) = 6$, ứng với đường đi a, d, e, z (đường đi ngắn nhất từ a đến e hợp với cạnh ez)



Bài toán đường đi ngắn nhất
Thuật toán Dijkstra

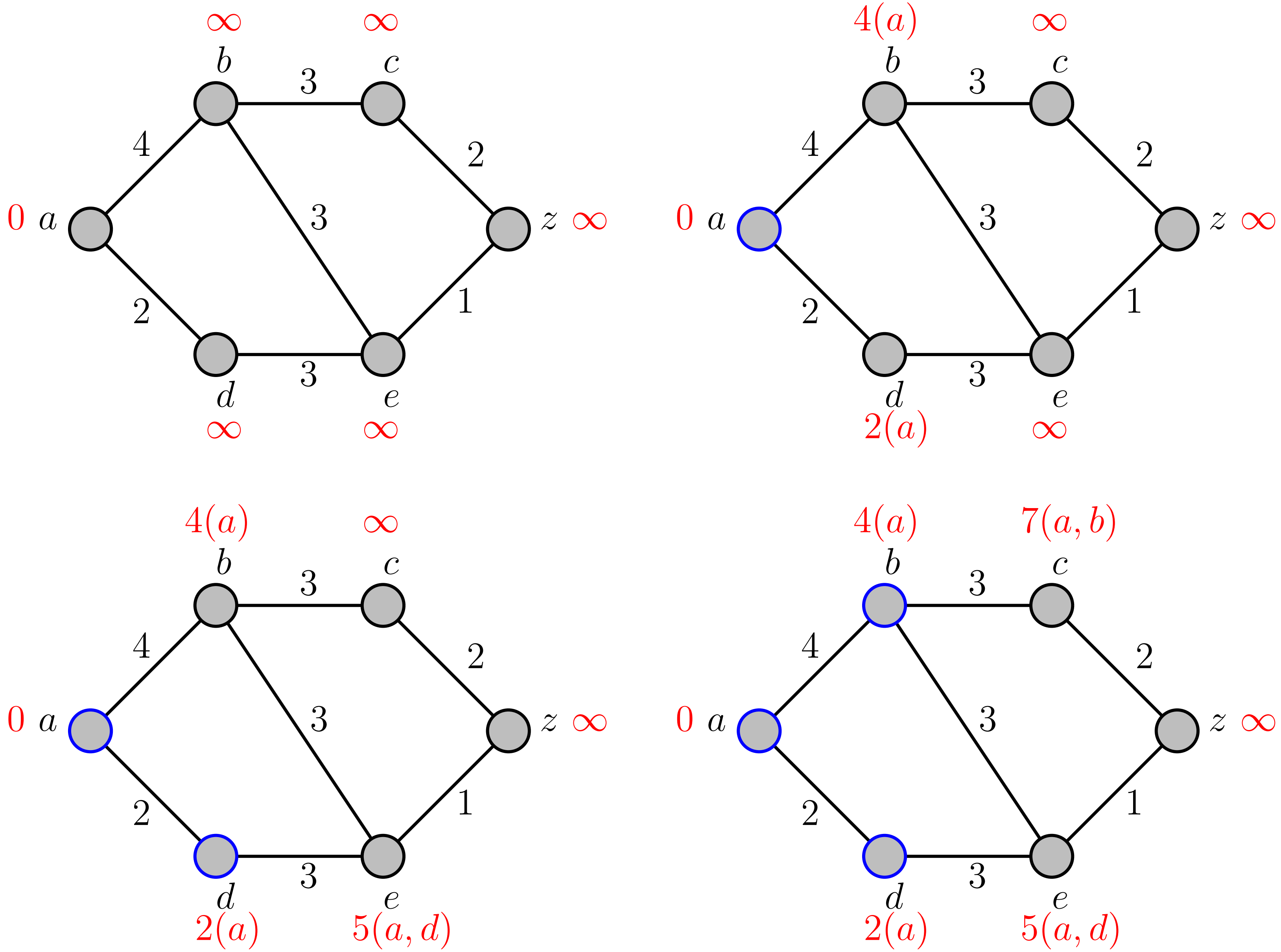
Bài toán đường đi ngắn nhất

- **Input:** Đơn đồ thị vô hướng $G = (V, E, w)$ trong đó $V = \{v_0 = a, v_1, \dots, v_n = z\}$, $w : [V]^2 \to \mathbb{R}^+ \cup \{\infty\}$ với $w(v_i, v_j) = \infty$ nếu $v_iv_j \notin E$
- **Output:** Khoảng cách $d_G(a, z)$

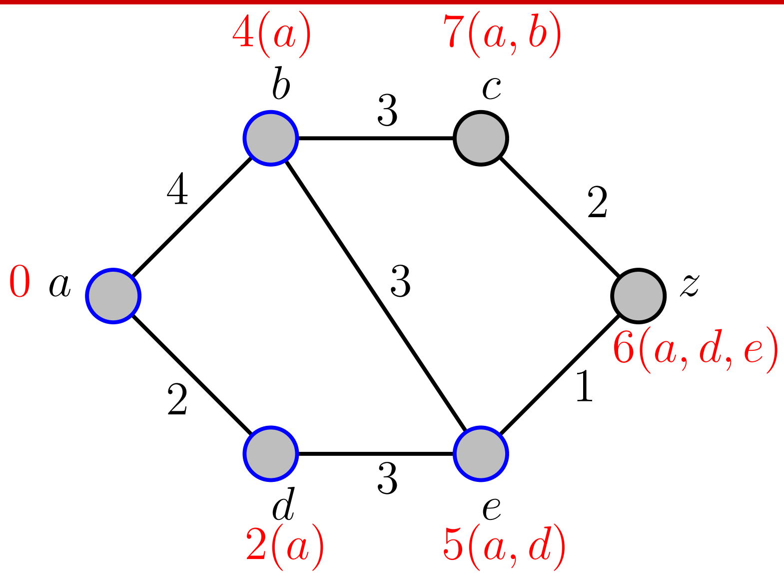
- Khởi tạo: Gán nhãn $L(a) := 0$, $L(v_i) := \infty$ với mọi $v_i \neq a$, và lấy một tập $S := \emptyset$.
- Trong khi $z \notin S$, lặp lại các bước sau:
  - Gọi u là đỉnh không thuộc S với $L(u)$ nhỏ nhất. Thêm u vào S.
  - Với mọi đỉnh v không thuộc S
    - Nếu $L(u) + w(u, v) < L(v)$ thì gán $L(v) := L(u) + w(u, v)$ (Sửa đổi nhãn của các đỉnh không thuộc S)
- Cuối cùng, $L(z)$ là độ dài đường đi ngắn nhất (khoảng cách) từ a đến z.

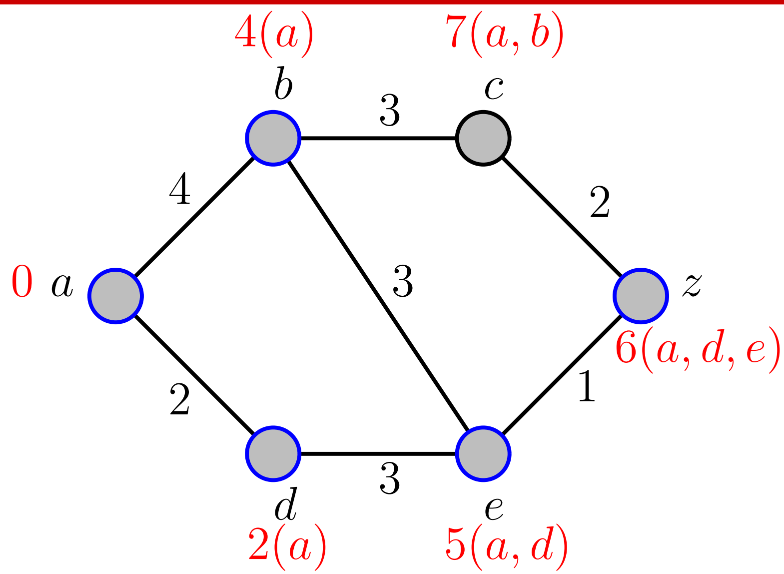
Bài toán đường đi ngắn nhất
Thuật toán Dijkstra

- Tính đúng đắn của thuật toán Dijkstra có thể được chứng minh bằng cách *sử dụng bất biến vòng lặp*. Cụ thể, trước mỗi lần lặp, hàm L và tập S thỏa mãn các tính chất sau:
  (1) Với mọi $v \in S$, $L(v)$ là khoảng cách từ a đến v
  (2) Với mọi $v \in V - S$, $L(v)$ là độ dài đường đi ngắn nhất từ a đến v **chỉ qua các đỉnh thuộc** $S \cup \{v\}$
  (3) Với mọi $u \in S$ và $v \in V - S$, $L(u) \leq L(v)$
- Thông thường, thuật toán Dijkstra chạy trong thời gian $O(n^2)$, với $n = |V|$
- Với cấu trúc dữ liệu "đống Fibonacci" (Fibonacci Heap), thuật toán Dijkstra có thể được lập trình để chạy trong thời gian $O(m + n \log n)$, với $n = |V|$ và $m = |E|$. Hiệu quả của cách lập trình này được thể hiện khi chạy với các "đồ thị thưa" (sparse graph) cực lớn (các đồ thị có m rất nhỏ so với n^2)
- Thuật toán Dijkstra cũng có thể được lập trình để xuất ra một đường đi ngắn nhất từ a đến mỗi đỉnh khác trong đồ thị


∞ ∞
b 3 c
4 2
0 a 3 z ∞
2 1
d 3 e
∞ ∞
4(a) ∞
b 3 c
4 2
0 a 3 z ∞
2 1
d 3 e
2(a) ∞
4(a) ∞
b 3 c
4 2
0 a 3 z ∞
2 1
d 3 e
2(a) 5(a, d)
4(a) 7(a, b)
b 3 c
4 2
0 a 3 z ∞
2 1
d 3 e
2(a) 5(a, d)

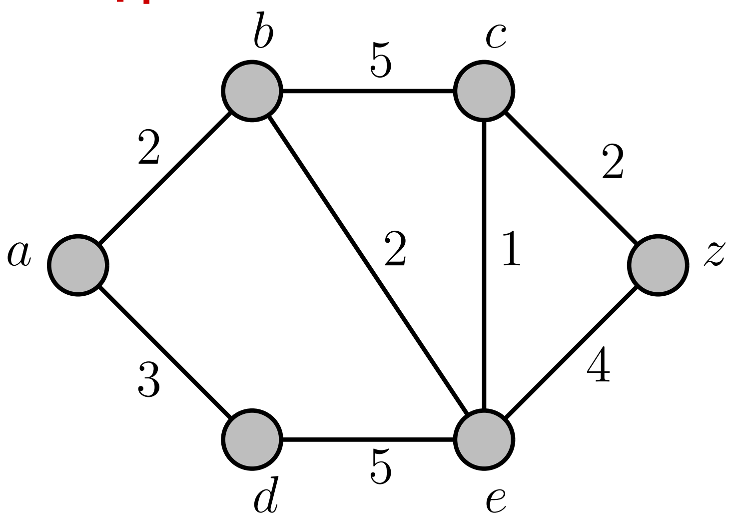
Bài toán đường đi ngắn nhất
Thuật toán Dijkstra



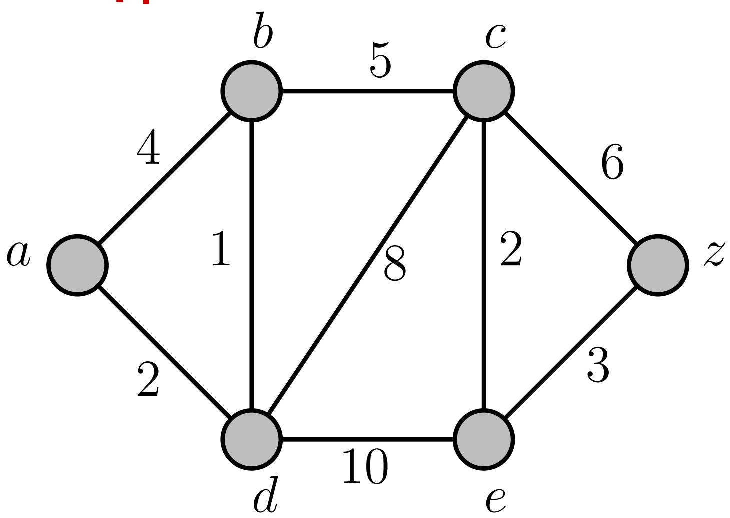


Áp dụng thuật toán Dijkstra để tìm khoảng cách từ a đến z trong mỗi đồ thị sau

Bài tập 1



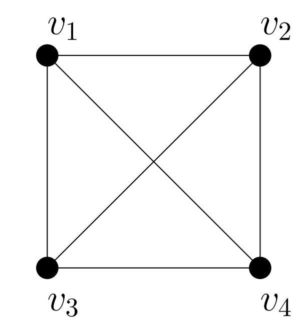
Bài tập 2

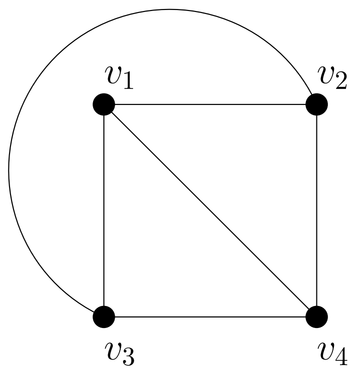


- Một đồ thị được gọi là *đồ thị phẳng (planar graph)* nếu nó có thể được vẽ trên mặt phẳng sao cho không có hai cạnh nào cắt nhau (ở một điểm không phải là đầu mút của cạnh).
- Hình vẽ như thế được gọi là một *biểu diễn phẳng (planar representation)* của đồ thị.

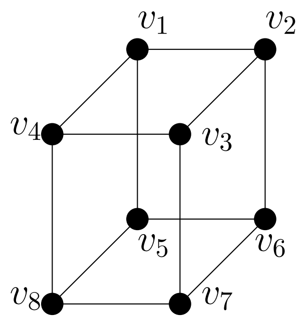
Ví dụ 2

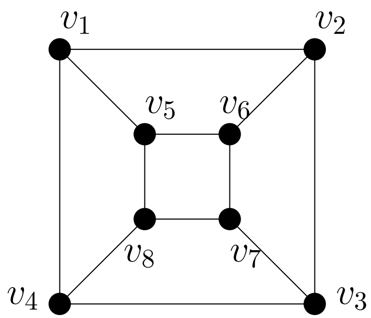
K_4





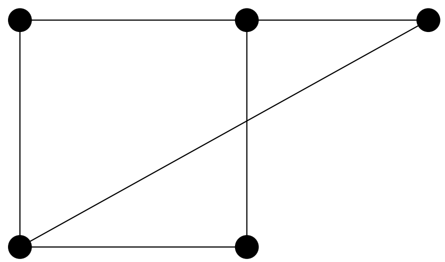
Q_3



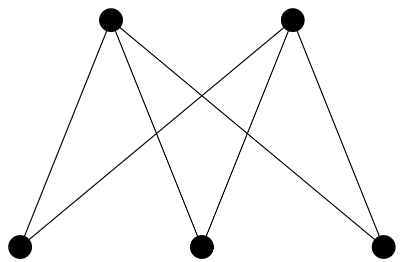


Bài tập 3

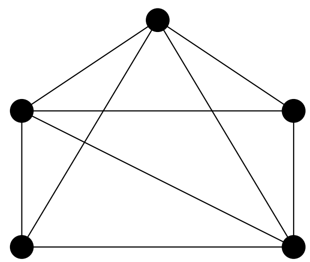
Tìm một biểu diễn phẳng của các đồ thị phẳng sau



G_1



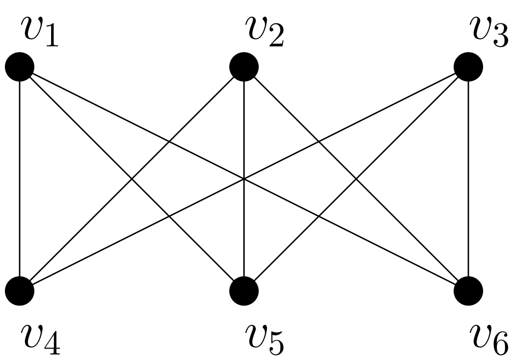
G_2



G_3

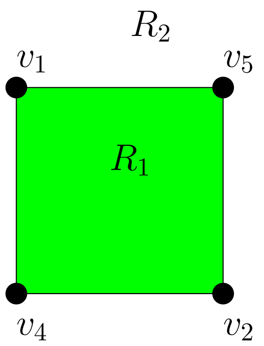
Ví dụ 3

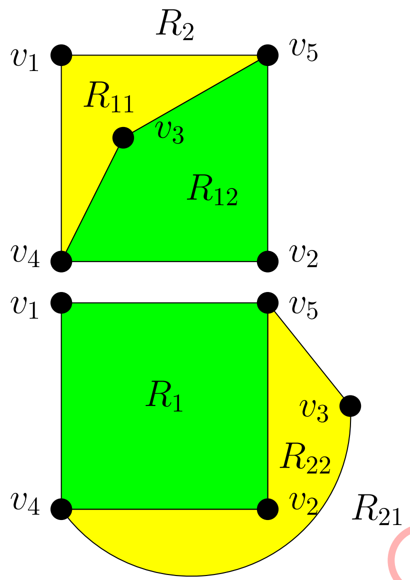
$K_{3,3}$ không là đồ thị phẳng



Ta chứng minh khẳng định trên bằng phản chứng. Giả sử $K_{3,3}$ là đồ thị phẳng

- Trong bất kỳ biểu diễn phẳng nào của $K_{3,3}$ ta có v_1 và v_2 đều phải luôn nối với v_4 và v_5 Các đỉnh này chia mặt phẳng thành hai miền R_1 và R_2.
- Đỉnh v_3 thuộc R_1 hoặc R_2
- Vị trí của v_6?



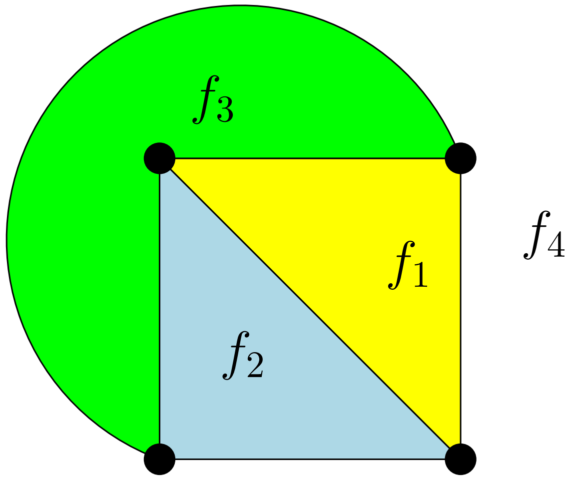


- Biểu diễn phẳng của một đồ thị phẳng $G = (V, E)$ chia mặt phẳng thành các *miền (region)*, kể cả *miền vô hạn (unbounded region)*
- Hai điểm bất kỳ trong cùng một miền có thể được nối với nhau bằng một nét liền mà không cắt bất kỳ cạnh nào
- *Bậc (degree)* của một miền f, ký hiệu $\deg(f)$, là số cạnh của G trên biên của f

Ví dụ 4

Biểu diễn phẳng của K_4

- chia mặt phẳng thành 4 miền f_1, f_2, f_3, và f_4; và
- $\deg(f_1) = \deg(f_2) = \deg(f_3) = \deg(f_4) = 3$



- **Chú ý:** $\sum_{\text{miền } f \text{ của } G} \deg(f) \leq 2|E|$ (Mỗi cạnh thuộc tối đa hai miền)

Định lý 1: Công thức Euler

Giả sử G là một đơn đồ thị phẳng và liên thông gồm m cạnh, n đỉnh, và r miền. Ta có $n - m + r = 2$

Chứng minh.

- Xây dựng dãy đồ thị $G_1, G_2, \dots, G_m = G$
  - Chọn một cạnh bất kỳ của G làm G_1
  - G_i được tạo thành từ G_{i-1} bằng cách thêm một cạnh bất kỳ liên thuộc với một đỉnh của G_{i-1} $(i \in \{2, 3, \dots, m\})$
  - Gọi n_i, m_i, r_i lần lượt là số đỉnh, cạnh, và miền của một biểu diễn phẳng của G_i
- Công thức Euler đúng với mọi G_i
  - Ta có $n_1 - m_1 + r_1 = 2 - 1 + 1 = 2$
  - Giả sử công thức Euler đúng với G_i, tức là $n_i - m_i + r_i = 2$ Gọi $a_{i+1}b_{i+1}$ là cạnh thêm vào G_i để tạo thành G_{i+1}. Có hai khả năng:
    - một trong hai đỉnh a_{i+1}, b_{i+1} không thuộc G_{i-1}
    - cả a_{i+1} và b_{i+1} thuộc G_{i-1}

Hệ quả 2

Giả sử G là một đồ thị phẳng liên thông gồm m cạnh và n đỉnh ($n \geq 3$). Khi đó, $m \leq 3n - 6$. Thêm vào đó, nếu $m = 3n - 6$ thì mỗi miền của G có chính xác 3 cạnh trên biên.

Chứng minh.

- Nhận xét rằng mỗi miền của G có ít nhất 3 cạnh trên biên, do đó $\sum_{\text{miền } f \text{ của } G} \deg(f) \geq 3r$. Mặt khác, ta cũng có $\sum_{\text{miền } f \text{ của } G} \deg(f) \leq 2m$. Suy ra $3r \leq 2m$.
- Áp dụng công thức Euler, ta có $2 = n - m + r \leq n - m + 2m/3$, suy ra $m \leq 3n - 6$.

□

Bài tập 4

Giả sử G là một đồ thị đơn phẳng và liên thông gồm 20 đỉnh, mỗi đỉnh có bậc 3. Một biểu diễn phẳng của G chia mặt phẳng thành bao nhiêu miền?

Bài tập 5

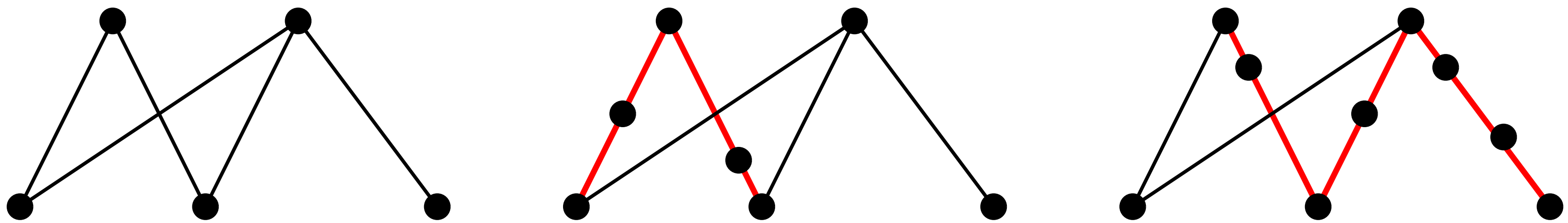
Chứng minh K_5 không là đồ thị phẳng

Bài tập 6

Chứng minh rằng nếu G là một đơn đồ thị phẳng và liên thông thì G có một đỉnh có bậc nhỏ hơn hoặc bằng 5

- Cho đồ thị G. Một *phép phân chia (subdivision)* một cạnh e của G được thực hiện bằng cách thay thế e bằng một đường đi đơn
- Hai đồ thị G_1 và G_2 được gọi là *đồng phôi (homeomorphic)* nếu chúng được xây dựng từ cùng một đồ thị thông qua một dãy các phép phân chia

Ví dụ 5



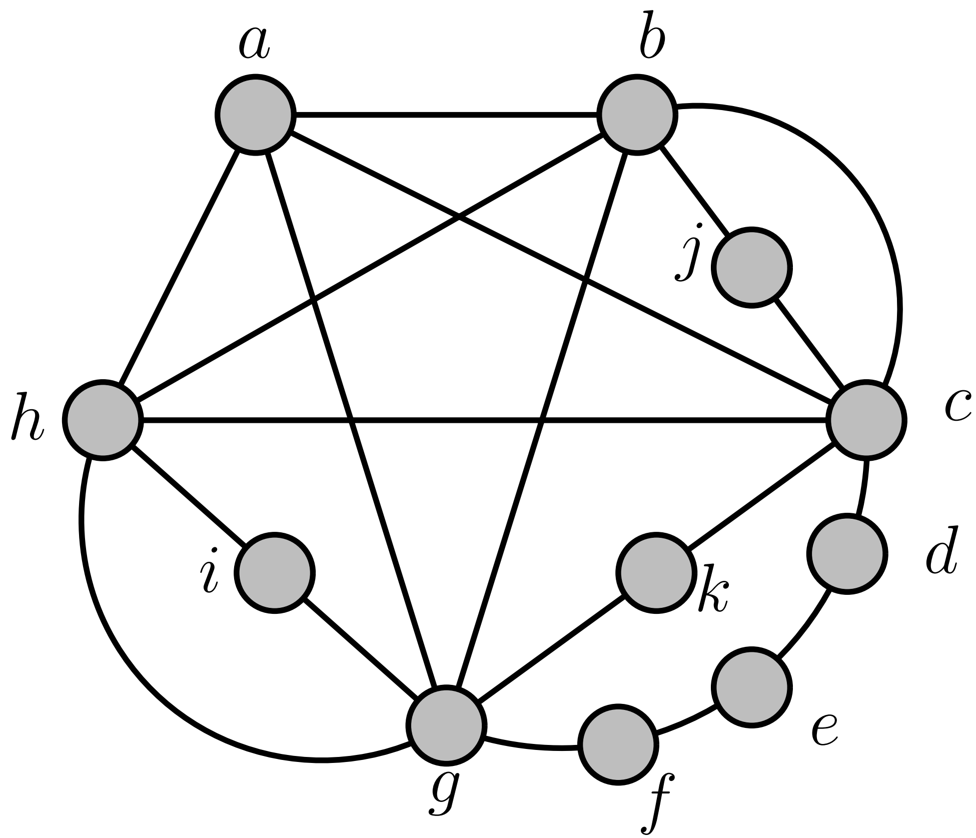
G G_1 G_2

Định lý 3: Định lý Kuratowski

G là đồ thị phẳng khi và chỉ khi nó không chứa bất kỳ đồ thị nào đồng phôi với K_5 hoặc $K_{3,3}$

Bài tập 7

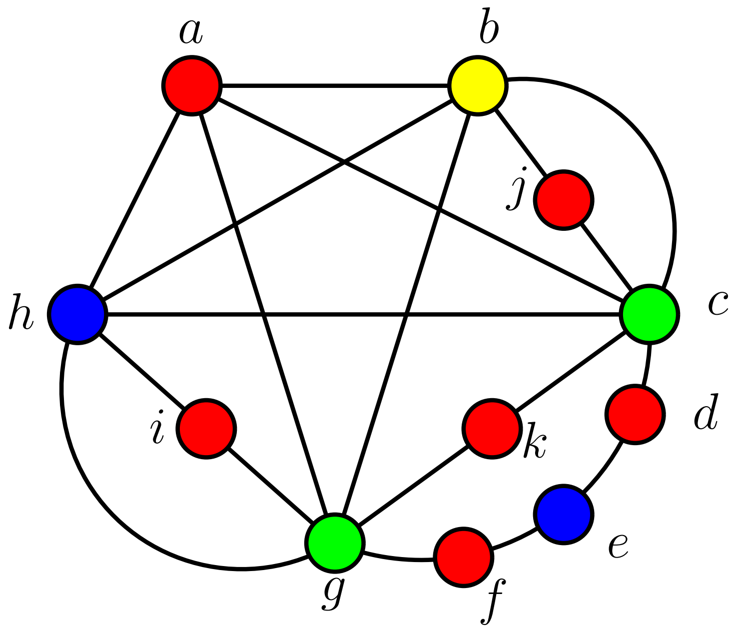
Sử dụng Định lý Kuratowski, hãy chứng minh đồ thị sau không là đồ thị phẳng



Tô màu đồ thị

Cho đơn đồ thị vô hướng $G = (V, E)$

- *Tô màu* một đồ thị đơn là sự gán màu cho các đỉnh của đồ thị sao cho không có hai đỉnh liền kề được gán cùng một màu. Cụ thể, với các "màu" $1, 2, \dots, k$, một *cách tô màu các đỉnh (vertex k-coloring)* của G là một hàm $f : V \to \{1, 2, \dots, k\}$ thỏa mãn $f(u) \neq f(v)$ với mọi $u, v \in V$ với $uv \in E$



- *Sắc số (chromatic number)* của G, ký hiệu $\chi(G)$, là số tối thiểu các màu cần thiết để tô màu G

Tô màu đồ thị

- Một số nhận xét
  (1) Mọi đồ thị G gồm n đỉnh có thể được tô màu bằng n màu
  (2) $\chi(K_n) = n$
  (3) Ta ký hiệu $\omega(G)$ là số nguyên dương lớn nhất $r \geq 1$ thỏa mãn K_r là đồ thị con của G. Với mọi đồ thị G, ta có $\omega(G) \leq \chi(G)$. Thông thường, $\omega(G) \neq \chi(G)$
  (4) $\chi(C_n) = 2$ nếu $n \geq 4$ chẵn và $\chi(C_n) = 3$ nếu $n \geq 3$ lẻ
  (5) G là đồ thị hai phần khi và chỉ khi $\chi(G) = 2$
- *Chưa biết* có tồn tại hay không một thuật toán chạy trong thời gian đa thức để xác định xem một đồ thị G có thể được tô màu bằng 3 màu hay không
- Để chỉ ra $\chi(G) = k$ với đồ thị G nào đó, ta cần:
  - Chỉ ra một cách tô màu các đỉnh của G bằng k màu.
  - Chỉ ra rằng không thể dùng ít hơn k màu để tô màu các đỉnh của G.

Bài tập 8

Tính $\chi(W_n)$, $\chi(K_{m,n})$, *và* $\chi(Q_n)$

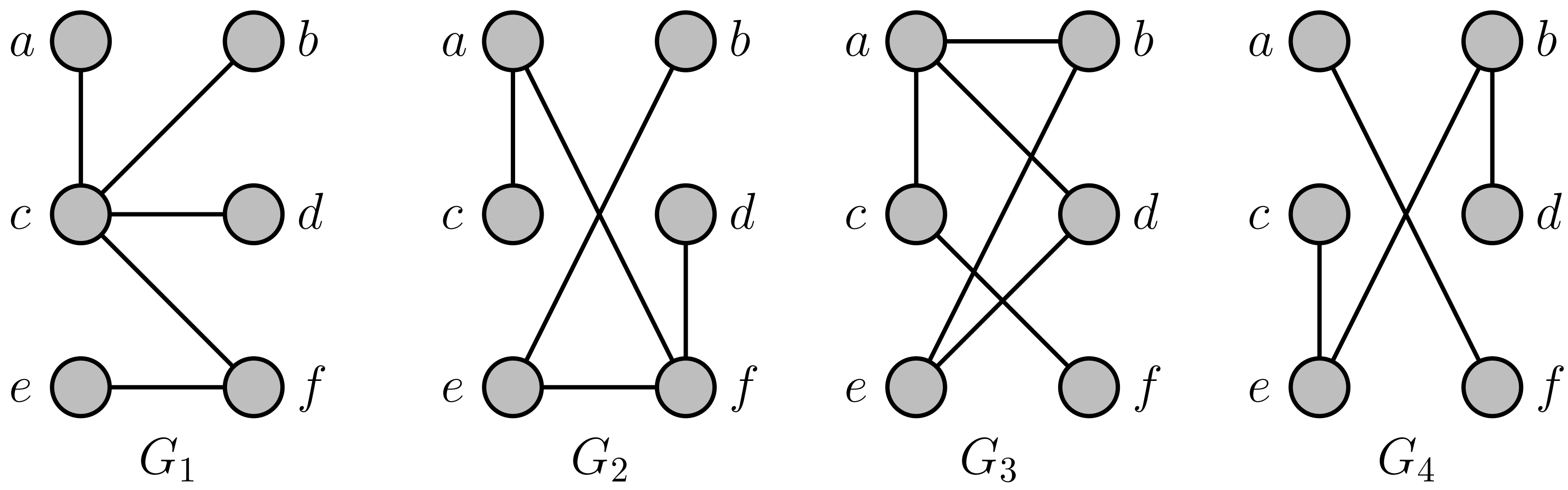
Tô màu đồ thị

Định lý 4: Định lý bốn màu

Với đồ thị phẳng G, ta luôn có $\chi(G) \leq 4$

- Định lý bốn màu lần đầu tiên được phát biểu dưới dạng một phỏng đoán vào năm 1850
- Năm 1976, Kenneth Appel and Wolfgang Haken (Đại học Illinois) chứng minh định lý bốn màu bằng cách giả sử nếu Định lý bốn màu là sai thì sẽ có một phản ví dụ thuộc một trong gần 2000 loại khác nhau, và chỉ ra rằng không có loại nào dẫn đến phản ví dụ. Các trường hợp này được phân tích cẩn thận nhờ máy tính

- Đơn đồ thị vô hướng $T = (V, E)$ được gọi là một *cây (tree)* nếu T là đồ thị liên thông và không có chu trình
- Đơn đồ thị vô hướng $T = (V, E)$ được gọi là một *rừng (forest)* nếu T là đồ thị không có chu trình



Hình: G_1, G_2 là cây. G_3, G_4 không là cây. G_4 là rừng. G_3 không là rừng

Định lý 5

Một đồ thị vô hướng G là một cây khi và chỉ khi giữa hai đỉnh bất kỳ của G tồn tại một đường đi đơn duy nhất

Chứng minh.

($\Rightarrow$) Giả sử G là cây

- Do G liên thông, với hai đỉnh bất kỳ $u, v \in V$, *tồn tại* một đường đi đơn giữa u và v
- Ta chứng minh đường đi này là *duy nhất*. Giả sử phản chứng rằng có ít nhất hai đường đi đơn $P = v_0, v_1, \ldots, v_k$ và $Q = w_0, w_1, \ldots, w_\ell$ khác nhau giữa hai đỉnh u và v, với $v_0 = w_0 = u$ và $v_k = w_\ell = v$
  - Do $v_0 = w_0$, tồn tại i thỏa mãn $v_j = w_j$ với mọi $0 \leq j \leq i$ và $v_{i+1} \neq w_{i+1}$
  - Do $v_k = w_\ell$, tồn tại $p \geq i + 1$ nhỏ nhất thỏa mãn $v_p \in \{w_i, \ldots, w_\ell\}$ hoặc $w_p \in \{v_i, \ldots, v_k\}$. Giả sử $v_p = w_q$ với $q \geq i$ nhỏ nhất có thể
  - $v_i, v_{i+1}, \ldots, v_p$ và $w_q, \ldots, w_{i+1}, w_i$ tạo thành một chu trình trong G, mâu thuẫn với giả thiết G là cây

Chứng minh (tiếp).

($\Leftarrow$) Giả sử với mọi cặp đỉnh $u, v \in V$, tồn tại một đường đi đơn duy nhất giữa u và v trong G

- Theo định nghĩa, G là *liên thông*
- Ta chứng minh G *không có chu trình*. Giả sử phản chứng rằng tồn tại một chu trình C trong G. Do đó, với hai đỉnh u, v bất kỳ thuộc C, có hai đường đi đơn khác nhau nối u và v, mâu thuẫn với giả thiết tồn tại duy nhất một đường đi đơn giữa hai đỉnh bất kỳ trong G

$\square$

Định lý 6

Mọi cây $T = (V, E)$ với $|V| \geq 2$ có ít nhất hai đỉnh bậc 1

Chứng minh.

Lấy một đường đi đơn *dài nhất* $v_0, v_1, \ldots, v_m$ trong T. Bậc của v_0 và v_m đều bằng 1, nếu không ta có thể tìm được một đường đi dài hơn □

Định lý 7

Mọi cây gồm n đỉnh có chính xác $n - 1$ cạnh

Chứng minh.

Quy nạp theo n

- **Bước cơ sở:** Với $n = 1$, cây gồm 1 đỉnh có 0 cạnh
- **Bước quy nạp:** Giả sử mọi cây gồm k đỉnh có chính xác $k - 1$ cạnh, với $k \geq 1$. Ta chứng minh mọi cây T gồm $k + 1$ đỉnh có chính xác k cạnh. Thật vậy, do T là một cây, T có một đỉnh u bậc 1 nào đó. $T - u$ là một cây gồm k đỉnh và theo giả thiết quy nạp có $k - 1$ cạnh. Do đó, T có $(k - 1) + 1 = k$ cạnh

□

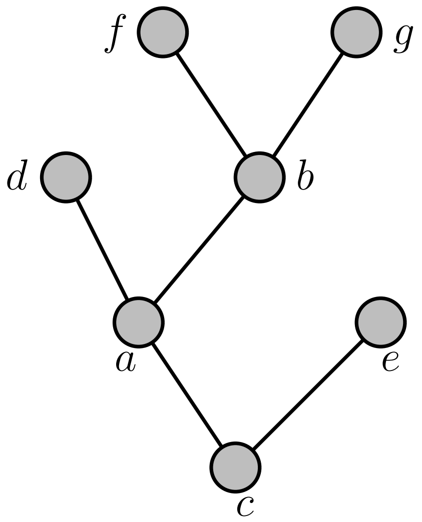
Bài tập 9
Chứng minh rằng một đồ thị $G = (V, E)$ là một cây khi và chỉ khi G không có chu trình và với mọi cặp đỉnh $u, v \in V$ thỏa mãn $uv \notin E$, đồ thị $G + uv$ có chính xác một chu trình

Bài tập 10
Giả sử một cây $T = (V, E)$ có số các cạnh là một số chẵn. Chứng minh rằng tồn tại một đỉnh trong T có bậc chẵn

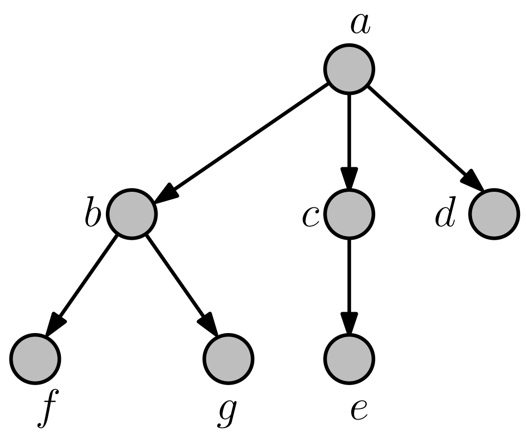
Bài tập 11
Cho $T = (V, E)$ là một cây. Gọi Δ là bậc lớn nhất của một đỉnh trong T, nghĩa là, $\Delta = \max_{v \in V} \deg_T(v)$. Chứng minh rằng T có ít nhất Δ đỉnh bậc 1

Bài tập 12
Cho $T = (V, E)$ là một cây với $|V| > 1$. Giả sử nếu v là một đỉnh liền kề với một đỉnh bậc 1 trong T thì $\deg_T(v) \geq 3$. Chứng minh rằng tồn tại hai đỉnh bậc 1 trong T có chung một đỉnh liền kề với hai đỉnh đó

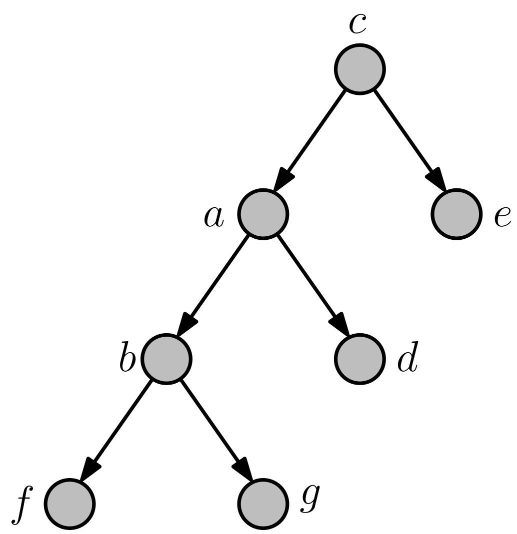
Một *cây có gốc (rooted tree)* là một cây trong đó một đỉnh được coi là *đỉnh gốc (root)* và mọi cạnh được định hướng xa khỏi đỉnh gốc. Ta ký hiệu một cây T với gốc r bằng cặp (T, r)



Cây T



Cây T với gốc a



Cây T với gốc c

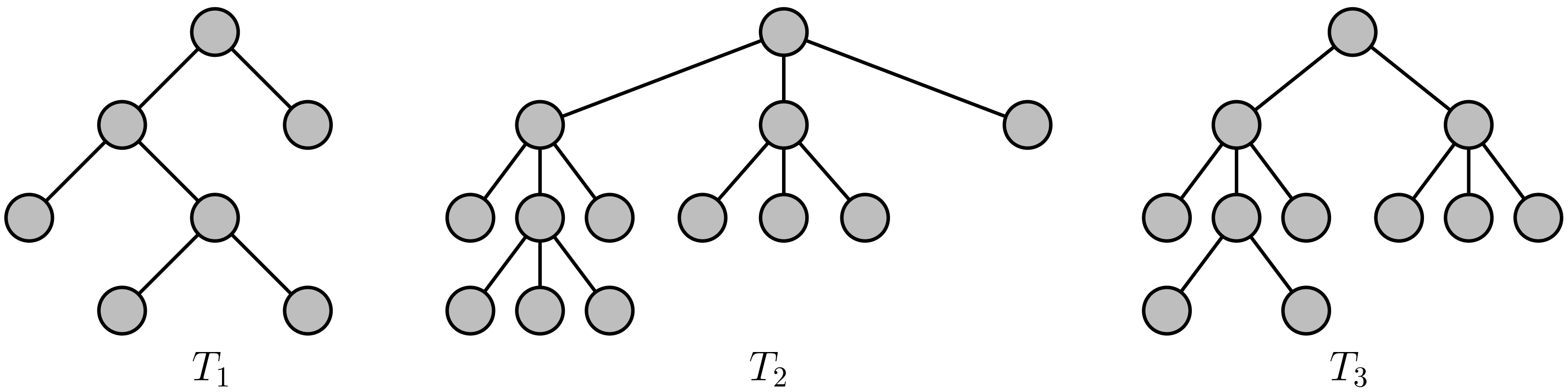
Thông thường, một cây có gốc (T, r) được vẽ với *đỉnh gốc ở trên đỉnh của đồ thị*. Khi đó, ta có thể *bỏ qua các hướng của các cạnh*, do việc lựa chọn đỉnh gốc đã xác định các hướng này

- Với một đỉnh v khác đỉnh gốc của một cây có gốc T
  - *Đỉnh cha (parent)* của v là đỉnh u duy nhất sao cho có một cạnh có hướng từ u đến v. Ta cũng gọi v là *đỉnh con (child)* của u.
  - Đỉnh v được gọi là *đỉnh lá (leaf)* nếu v không có đỉnh con. Các đỉnh có đỉnh con được gọi là *các đỉnh trong (internal vertices)*
  - Các đỉnh có chung một đỉnh cha được gọi là *các đỉnh anh em (siblings)*
  - *Các đỉnh tổ tiên (ancestors)* của v là các đỉnh trên đường đi từ gốc tới đỉnh đó. *Các đỉnh con cháu (descendants)* của v là các đỉnh có v là một đỉnh tổ tiên
  - *Cây con (subtree) với gốc v* là đồ thị con của T cảm sinh bởi v và các đỉnh con cháu của nó
- Đỉnh gốc là đỉnh duy nhất trong cây *không có đỉnh cha* và là *tổ tiên của mọi đỉnh còn lại*. Đỉnh gốc luôn là một đỉnh trong, trừ trường hợp cây chỉ có một đỉnh duy nhất là đỉnh gốc thì đỉnh gốc là một đỉnh lá

- Một cây có gốc T được gọi là một *cây m-phân (m-ary tree)* nếu tất cả các đỉnh trong của T có tối đa là m đỉnh con. Nếu như mỗi đỉnh trong của T có chính xác m đỉnh con, ta gọi T là một *cây m-phân đầy đủ (full m-ary tree)*
- Cây m-phân với $m = 2$ được gọi là *cây nhị phân (binary tree)*

Bài tập 13

Trong số các cây sau, với m là số nguyên dương nào đó, cây nào là cây m-phân? Cây nào là cây m-phân đầy đủ?



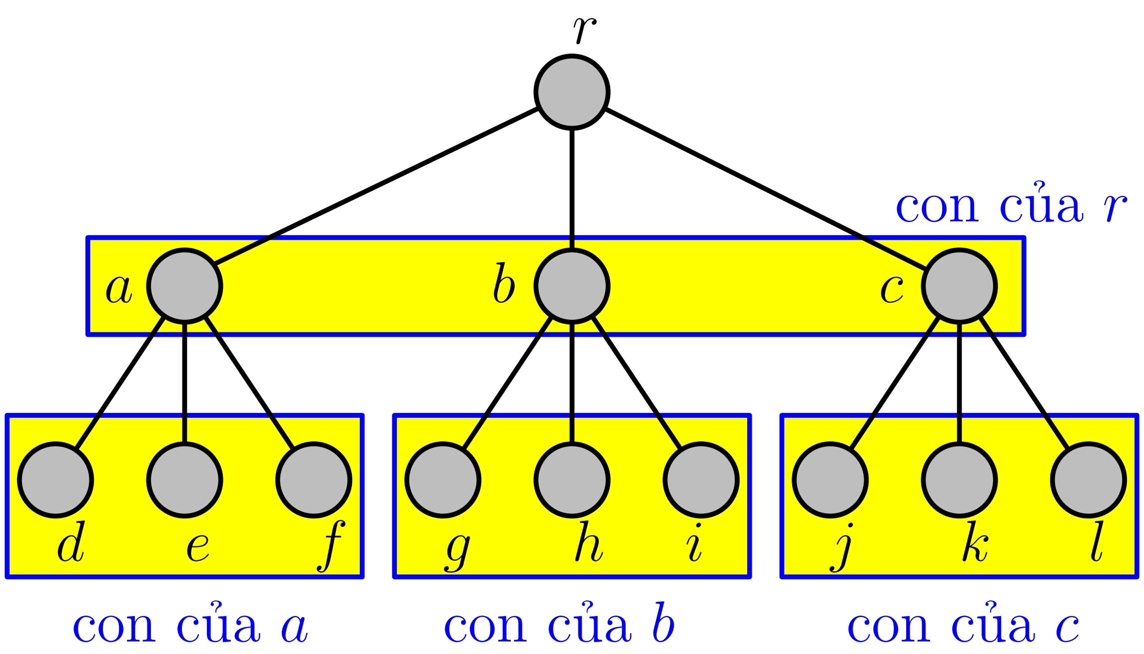
- Một *cây có gốc được sắp thứ tự (ordered rooted tree)* là một cây có gốc trong đó các đỉnh con của mỗi đỉnh được sắp xếp theo một thứ tự nào đó
- Ta thường vẽ cây có gốc được sắp thứ tự theo cách thông thường với hiểu ngầm rằng *các đỉnh con của mỗi đỉnh được sắp xếp theo thứ tự từ trái sang phải*
- Với một cây nhị phân được sắp thứ tự, nếu một đỉnh trong u có hai đỉnh con thì đỉnh thứ nhất được gọi là *đỉnh con trái (left child)*, đỉnh thứ hai được gọi là *đỉnh con phải (right child)*. Tương tự, cây con có gốc là đỉnh con trái được gọi là *cây con trái (left subtree)* và cây có gốc là đỉnh con phải được gọi là *cây con phải (right subtree)* của u
- Các cây có gốc sắp thứ tự được ứng dụng rộng rãi trong khoa học máy tính, ví dụ như các cây phân tích cú pháp (parse trees), cấu trúc cây trong các tài liệu XML (XML documents), cấu trúc cây mô tả các thư mục và tệp (directories and file system), v.v...

Định lý 8

Với mọi $m \geq 1$, mọi cây m-phân đầy đủ với i đỉnh trong có chính xác $n = m \cdot i + 1$ đỉnh

Chứng minh.

Mọi đỉnh khác đỉnh gốc là con của một đỉnh trong nào đó. Do đó có $m \cdot i$ đỉnh con (của các đỉnh trong) và 1 đỉnh gốc



- *Mức (level)* của một đỉnh trong một cây có gốc là độ dài của đường đi duy nhất từ gốc đến đỉnh đó. Mức của đỉnh gốc là 0
- *Độ cao (height)* h của một cây có gốc là độ dài lớn nhất của một đường đi từ gốc đến đỉnh lá. Nói cách khác, độ cao của một cây là mức lớn nhất của một đỉnh trong cây đó
- Một cây m-phân có độ cao h được gọi là *cân đối (balanced)* nếu tất cả các đỉnh lá có mức $h-1$ hoặc h

Định lý 9

Với mọi $m \geq 1$, một cây m-phân đầy đủ với

(i) n *đỉnh có* $i = (n-1)/m$ *đỉnh trong và* $l = ((m-1)n+1)/m$ *đỉnh lá;*

(ii) i *đỉnh trong có* $n = mi + 1$ *đỉnh và* $l = (m-1)i + 1$ *đỉnh lá;*

(iii) l *đỉnh lá có* $n = (ml-1)/(m-1)$ *đỉnh và* $i = (l-1)/(m-1)$ *đỉnh trong*

Bài tập 14

Sử dụng Định lý 8, hãy chứng minh Định lý 9

Bài tập 15

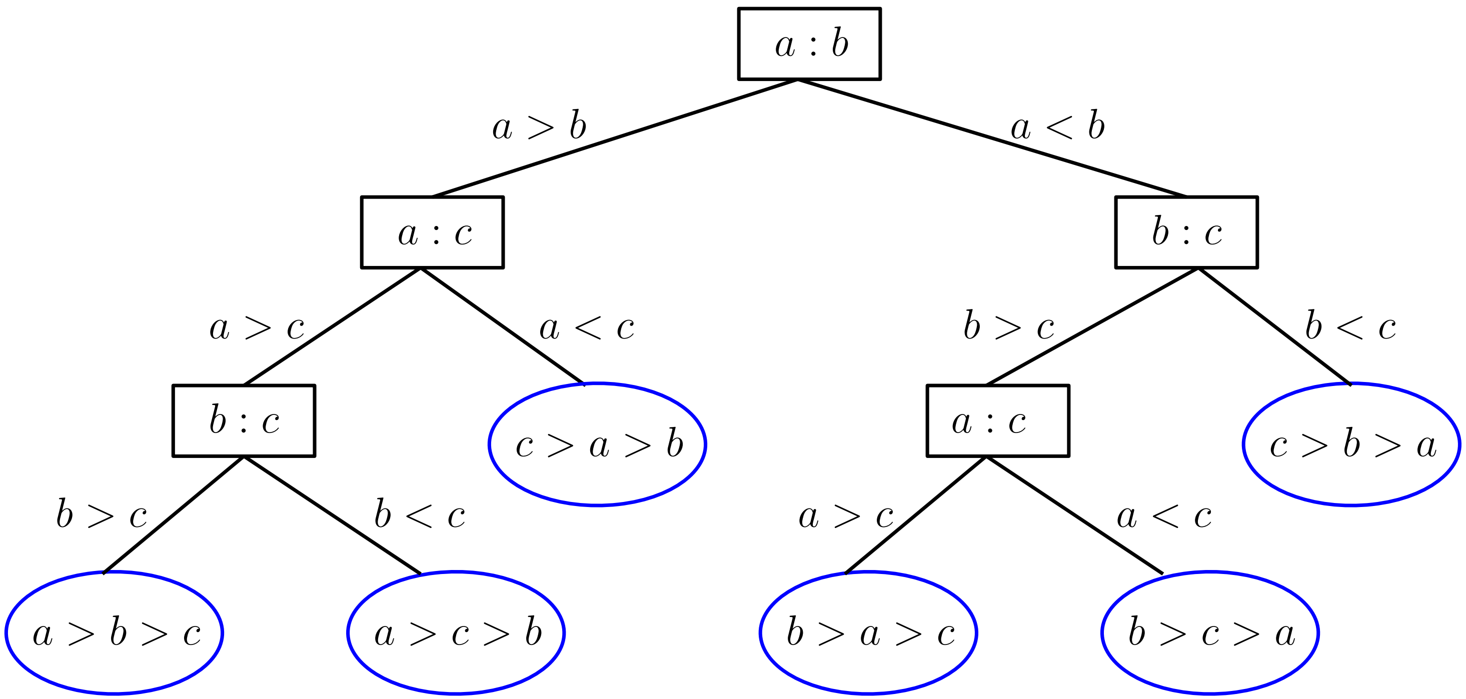
*Chứng minh rằng với mọi $m \geq 1$, có nhiều nhất m^h đỉnh lá trong một cây m-phân có độ cao h. (**Gợi ý:** Quy nạp theo h)*

Một ứng dụng của cây nhị phân được thể hiện trong định lý sau

Định lý 10

Mọi thuật toán sắp xếp n phần tử nào đó dựa trên các phép so sánh hai phần tử bất kỳ cần sử dụng ít nhất $\lceil \log_2(n!) \rceil$ phép so sánh trong trường hợp xấu nhất

- Do $\log_2(n!) = \Theta(n \log n)$, *một thuật toán sắp xếp n phần tử trong trường hợp xấu nhất cần $\Theta(n \log n)$ phép so sánh là một thuật toán tối ưu*, theo nghĩa là không tồn tại thuật toán sắp xếp nào khác có thời gian chạy trong thời gian xấu nhất tốt hơn
- **Ý tưởng:** Mô hình các thuật toán bằng cây nhị phân đầy đủ với *các đỉnh trong là các phép so sánh* và *các đỉnh lá là các kết quả thu được* khi đi từ gốc (gọi là *cây quyết định (decision tree)*). *Số phép so sánh* sử dụng để ra một kết quả chính là *độ dài đường đi từ gốc đến nút lá tương ứng*



Hình: Mô hình thuật toán sắp xếp dãy ba phần tử a, b, c bằng cây quyết định

Bài tập 16

Với mọi $m \geq 1$, chứng minh rằng trong một cây m-phân có l đỉnh lá và có độ cao h, ta có $h \geq \lceil \log_m l \rceil$