CHIẾN LƯỢC THAM ĂN

- Các đặc trưng cơ bản
- Các ví dụ minh họa

CÁC ĐẶC TRƯNG CƠ BẢN

- Chỉ sử dụng để giải các bài toán tối ưu
- Giải thuật được thiết kế bằng chiến lược tham ăn (greedy) giải các bài toán con trước khi giải bài toán gốc

CÁC ĐẶC TRƯNG CƠ BẢN

- Quá trình tìm lời giải được thực hiện thông qua một dãy các bước để tìm lời giải (nghiệm) tối ưu cục bộ (lời giải các bài toán con) cho đến khi tìm thấy lời giải bài toán gốc

CÁC ĐẶC TRƯNG CƠ BẢN

- Mỗi bước tìm nghiệm cục bộ thỏa mãn ba tính chất sau
  - Phải thỏa mãn ràng buộc bài toán (feasible)
  - Tối ưu cục bộ (locally optimal)
  - Không thay đổi nghiệm cục bộ trong các bước kế tiếp

CÁC ĐẶC TRƯNG CƠ BẢN

- Chiến lược greedy luôn thực hiện một "lựa chọn" tốt nhất hiện tại khi tìm kiếm lời giải bài toán con mà không quan tâm đến bước tiếp theo
- Chiến lược gready không giải tất cả các bài toán con (tối ưu) như qui hoạch động mà mỗi bước chỉ tìm lời giải tối ưu trong một tập các bài toán con

CÁC VÍ DỤ MINH HỌA

- Giải thuật Kruskal
- Giải thuật Dijkstra
- Bài toán người du lịch
- Bài toán tô màu

GIẢI THUẬT KRUSKAL

- **Bài toán** Cho đồ thị vô hướng liên thông có trọng số G, tìm cây bao trùm nhỏ nhất (minimum spanning tree) của G

GIẢI THUẬT KRUSKAL

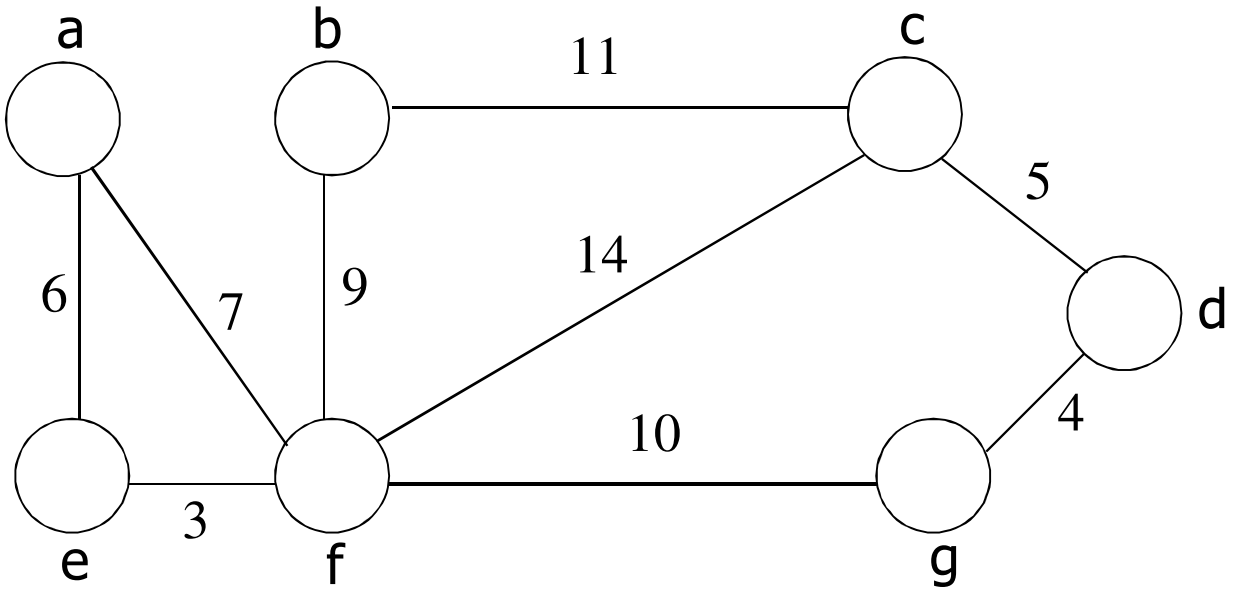
- Một bài toán con của bài toán tìm cây bao trùm nhỏ nhất là bài toán tìm một tập cạnh (ứng với số đỉnh xác định) có tổng trọng số nhỏ nhất (tối ưu cục bộ)
- Bài toán con nhỏ nhất ứng với tập cạnh F=∅
- Tại mỗi bước, mở rộng tập F thêm một cạnh mà F vẫn tối ưu (trọng số nhỏ nhất hiện tại)

GIẢI THUẬT KRUSKAL

- Thay vì khảo sát mọi tập con để tìm tập con có trọng số nhỏ nhất, giải thuật chọn một cạnh có trọng số nhỏ nhất (greedy) của đồ thị, thêm vào tập cạnh F của cây bao trùm sao cho không gây ra chu trình (sau đó loại cạnh vừa chọn ra khỏi đồ thị)
- Giải thuật dừng khi số cạnh trong F của cây bằng số đỉnh của đồ thị trừ 1

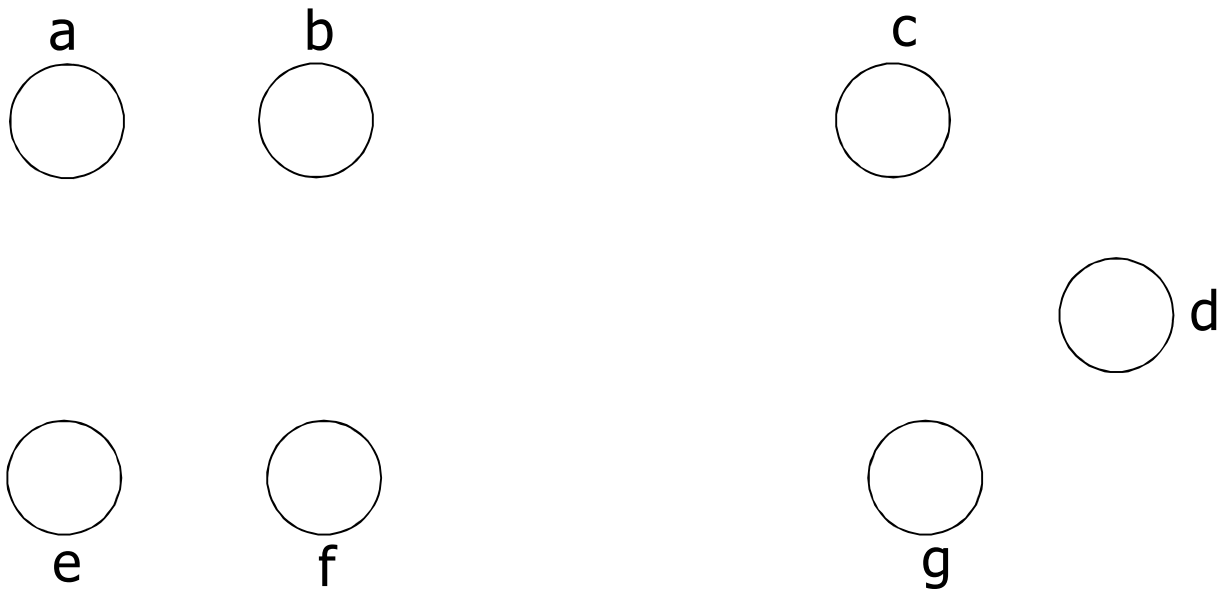
GIẢI THUẬT KRUSKAL

- Đồ thị G có trọng số



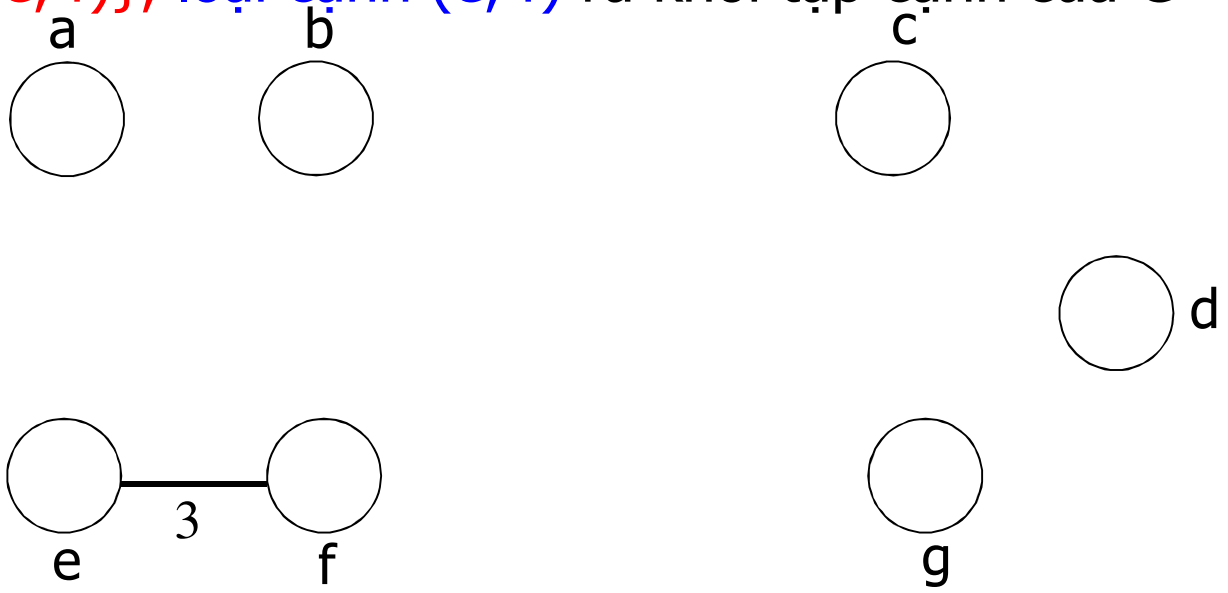
GIẢI THUẬT KRUSKAL

- Tập cạnh F=∅ (bài toán con nhỏ nhất)



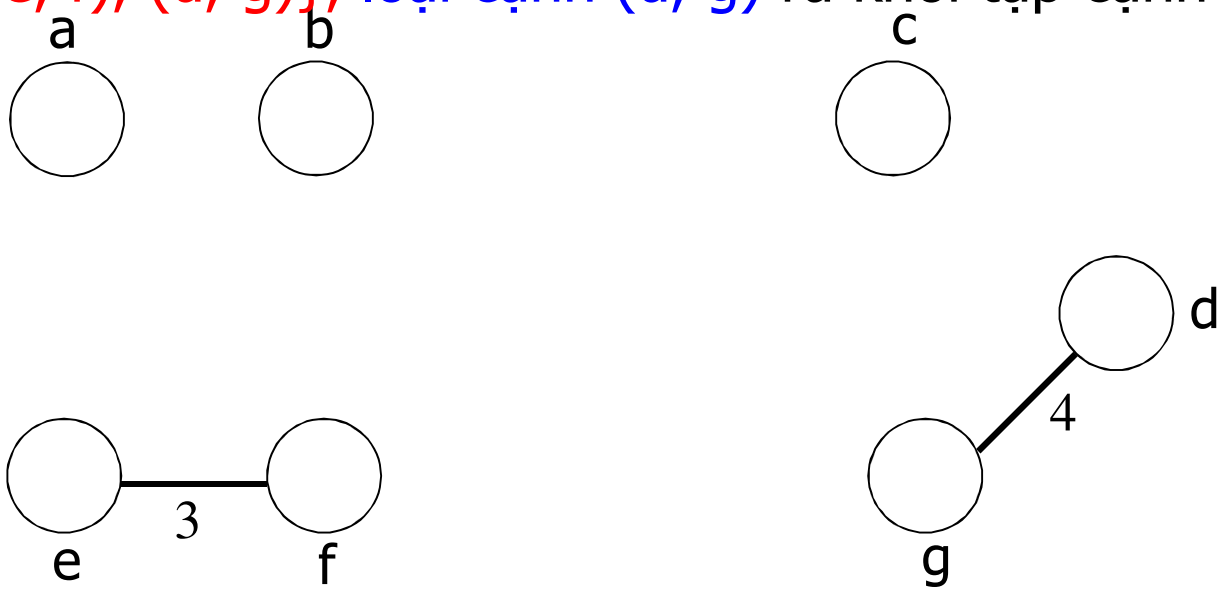
GIẢI THUẬT KRUSKAL

- Chọn cạnh (e, f) có trọng số bằng 3 (nhỏ nhất), tập cạnh mới F={(e, f)}, loại cạnh (e, f) ra khỏi tập cạnh của G



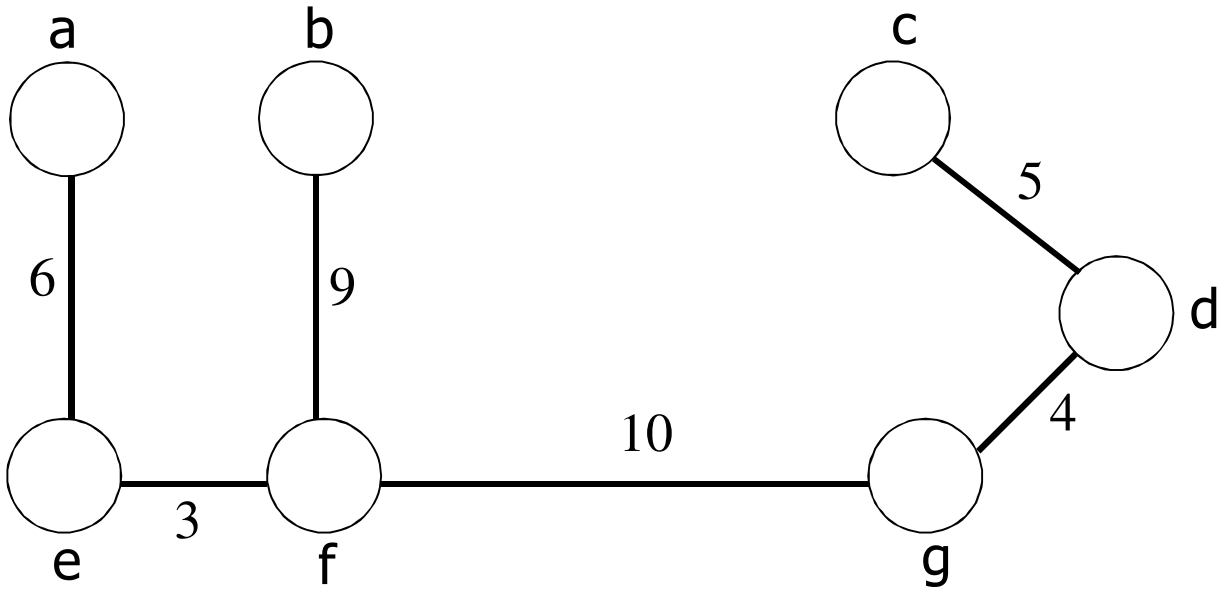
GIẢI THUẬT KRUSKAL

- Chọn cạnh (d, g) có trọng số bằng 4 (nhỏ nhất), tập cạnh mới F={(e, f), (d, g)}, loại cạnh (d, g) ra khỏi tập cạnh của G



GIẢI THUẬT KRUSKAL

- Sau 6 lần giải bài toán con (chọn cạnh), giải thuật kết thúc với cây khung nhỏ nhất T=(V, F) của G



GIẢI THUẬT KRUSKAL

Kruskal(G, w)

1. $F \leftarrow \varnothing$; $Q \leftarrow E[G]$; $N \leftarrow V[G]$
3. **while** $|F| < |N| - 1$ and $Q \neq \varnothing$
4. **do** $e \leftarrow$ Extractmin(Q) ▷ e có trọng số bé nhất
5. **if** $F \cup \{e\}$ not contain cycle **then** $F \leftarrow F \cup \{e\}$
6. **if** $|F| < |N| - 1$
7. **then** G is not connected
8. **else return** T ▷ T= (V, F)

GIẢI THUẬT KRUSKAL

- Q và N là các tập cạnh và đỉnh của G=(V, E)
- Thời gian thực hiện lệnh e ← Extractmin(Q) ở dòng 4 (lệnh cơ bản) không vượt quá $O(\log_2 E)$
- Chi phí cho tất cả các lần lặp trong vòng lặp **while** 3-5 không quá $O(V\log_2 E)$
- Do đó, tổng chi phí là $O(V\log_2 E)$

GIẢI THUẬT DIJKSTRA

- **Bài toán** Tìm các đường đi ngắn nhất từ một đỉnh s đến mọi đỉnh khác trong một đồ thị có trọng số không âm

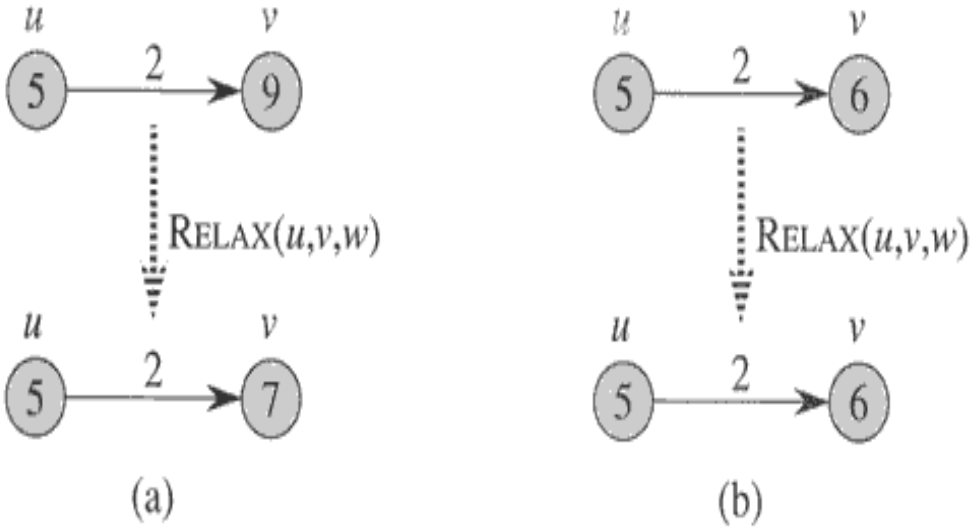
GIẢI THUẬT DIJKSTRA

- Mỗi bài toán con là một bài toán xác định một tập con các đỉnh và đường đi ngắn nhất từ s đến các đỉnh đó
- Bài toán con ban đầu ứng với tập đỉnh S= ∅ (hoặc S={s})
- Mỗi lần dùng chiến lược greedy để mở rộng S thêm một đỉnh
- Khi đã chọn một đỉnh v để thêm vào S thì sẽ không được thay đổi ở bước sau

GIẢI THUẬT DIJKSTRA

- Ký hiệu d[v] là một cận trên của độ dài đường đi ngắn nhất d(s,v) từ s đến v, giải thuật kiểm tra và giảm d[v] cho đến khi d[v]=d(s, v)
  - Nếu d[v]>d[u]+w(u,v) thì làm tốt cận trên d[v] bằng cách gán d[v]= d[u]+w(u,v) (gọi là relaxation)
  - Nếu d[v] đã tốt nhất thì đưa v vào trong tập S = {v $\in$V | d[v] =d(s, v)}, lúc này d[v] là độ dài đường đi ngắn nhất từ s đến v (việc chọn v khi d[v] tốt nhất là dùng greedy, bước đầu tiên đỉnh s có d[s] =0 được chọn)

GIẢI THUẬT DIJKSTRA



Nếu d[v]>d[u]+w(u,v)
thì gán d[v]= d[u]+w(u,v)

Figure 24.3 Relaxation of an edge (u, v) with weight $w(u, v) = 2$. The shortest-path estimate of each vertex is shown within the vertex. **(a)** Because $d[v] > d[u] + w(u, v)$ prior to relaxation, the value of $d[v]$ decreases. **(b)** Here, $d[v] \leq d[u] + w(u, v)$ before the relaxation step, and so $d[v]$ is unchanged by relaxation.

GIẢI THUẬT DIJKSTRA

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       then d[v] ← d[u] + w(u, v)
3            π[v] ← u
```

GIẢI THUẬT DIJKSTRA

```
DIJKSTRA(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   S ← ∅
3   Q ← V[G]
4   while Q ≠ ∅
5       do u ← EXTRACT-MIN(Q)
6          S ← S ∪ {u}
7          for each vertex v ∈ Adj[u]
8              do RELAX(u, v, w)
```

GIẢI THUẬT DIJKSTRA

INITIALIZE-SINGLE-SOURCE(G, s)

1 **for** each vertex $v \in V[G]$
2 **do** $d[v] \leftarrow \infty$
3 $\pi[v] \leftarrow$ NIL
4 $d[s] \leftarrow 0$

GIẢI THUẬT DIJKSTRA

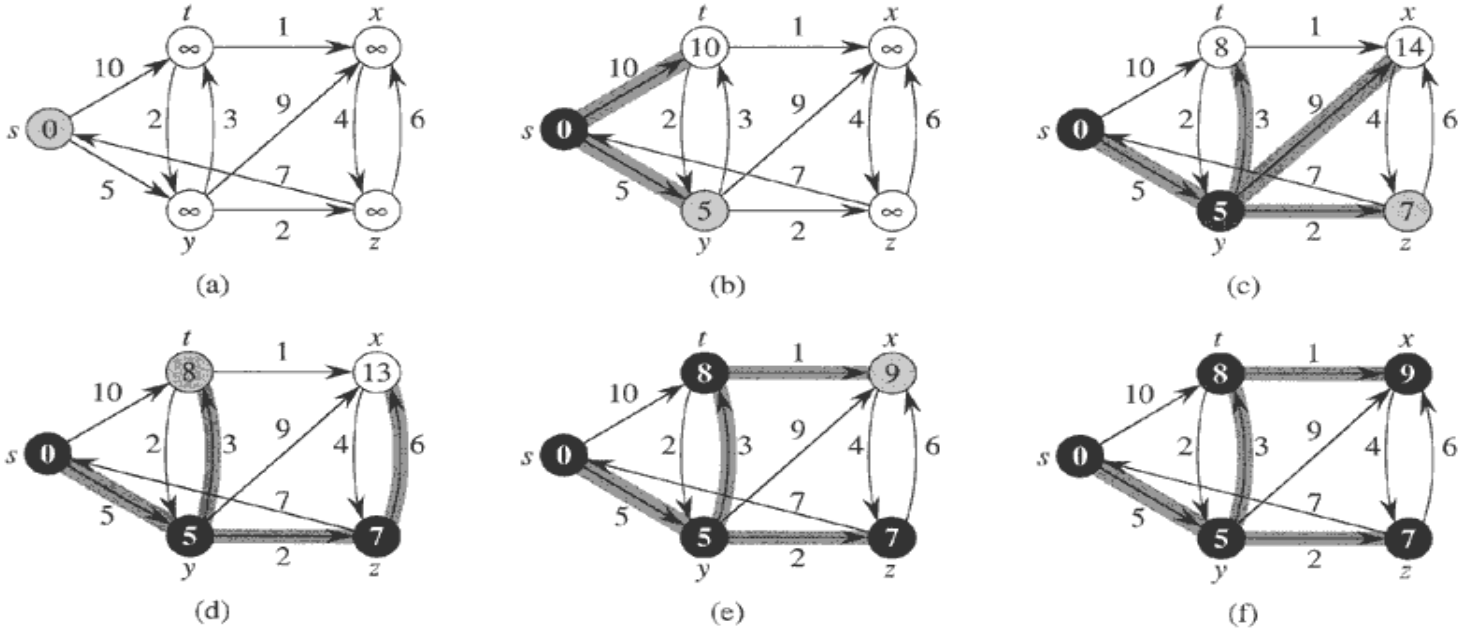


Figure 24.6 The execution of Dijkstra's algorithm. The source s is the leftmost vertex. The shortest-path estimates are shown within the vertices, and shaded edges indicate predecessor values. Black vertices are in the set S, and white vertices are in the min-priority queue $Q = V - S$. **(a)** The situation just before the first iteration of the **while** loop of lines 4–8. The shaded vertex has the minimum d value and is chosen as vertex u in line 5. **(b)–(f)** The situation after each successive iteration of the **while** loop. The shaded vertex in each part is chosen as vertex u in line 5 of the next iteration. The d and π values shown in part (f) are the final values.

THUẬT TOÁN DIJKSTRA

- Tổng chi phí EXTRACT-MIN là O(VlgV)
- Mỗi u được đưa vào S đúng một lần, mỗi cạnh kề trong Adj[u] được kiểm tra đúng một lần để giảm cận trên, số lần kiểm tra tất cả các cạnh kề như vậy là |E|
- Thời gian xử lý của RELAX là O(1)
- Vậy tổng chi phí là O(VlgV) +O(E) (nếu |E|>VlgV thì coi lệnh 8 là cơ bản, ngược lại là lệnh 5)

BÀI TOÁN NGƯỜI DU LỊCH

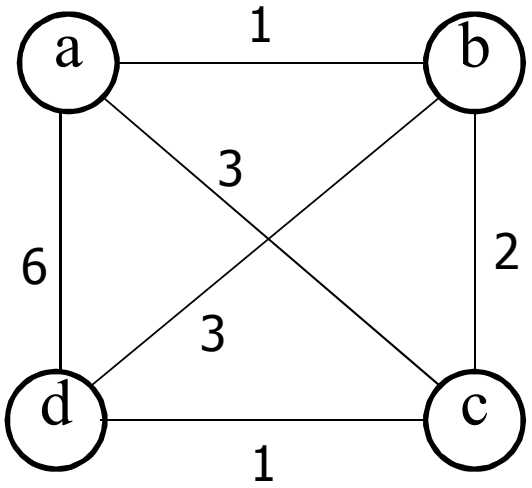
- Cho n thành phố, mỗi thành phố đều có đường đi đến tất cả các thành phố còn lại với độ dài biết trước. Một người đi du lịch (bán hàng) xuất phát từ một thành phố đã định, đi qua mọi thành phố, mỗi thành phố đúng một lần, rồi trở về thành phố xuất phát. Tìm đường đi ngắn nhất của người du lịch (bán hàng)

BÀI TOÁN NGƯỜI DU LỊCH

- Giải thuật sử dụng chiến lược tham ăn cho bài toán như sau
  - Chọn một thành phố kỳ làm thành phố xuất phát
  - Lặp lại các công việc (hoạt động) sau cho đến khi tất cả các thành phố đã được đi qua: đi đến thành phố gần nhất chưa đi qua (greedy) bắt đầu từ thành phố mà người du lịch đang có mặt
  - Trở về thành phố xuất phát

BÀI TOÁN NGƯỜI DU LỊCH

- Lộ trình người du lịch a→b →c →d →a



BÀI TOÁN NGƯỜI DU LỊCH

Traveler(T, w, s) //s là thành phố xuất phát

1. P $\leftarrow$ (s); Q $\leftarrow$ {s}; u $\leftarrow$s
3. **while** length[P]<|T|
4. **do** e $\leftarrow$ (u, v) with w(e) = min{w(u, x)| x $\in$Adj[u] and x$\notin$Q}
5. P $\leftarrow$ P$\otimes$v ▷ Thêm v vào đường đi P
7. Q $\leftarrow$Q$\cup${v}
6 u $\leftarrow$v
7. P $\leftarrow$ P$\otimes$s ▷ Trở về đỉnh xuất phát
8. **return** P

BÀI TOÁN NGƯỜI DU LỊCH

- Kích thước đầu vào là số đỉnh n của đồ thị
- Chi phí của lệnh 4 (cơ bản) không quá O(n)
- Vòng lặp while thực hiện n lần
- Vì vậy thời gian $T(n)=O(n^2)$
- **Lưu ý**: thuật toán có thể chỉ tìm được lộ trình xấp xỉ với lộ trình ngắn nhất

BÀI TOÁN TÔ MÀU

- **Bài toán** Cho một đồ thị vô hướng G, hãy tô màu các đỉnh của G sao cho sử dụng ít màu nhất và 2 đỉnh kề nhau không cùng màu

BÀI TOÁN TÔ MÀU

- Mỗi lần tô một số đỉnh nhiều nhất (greedy) không kề nhau cùng một màu
- Khởi đầu tập các đỉnh được tô và tập các màu dùng để tô là rỗng

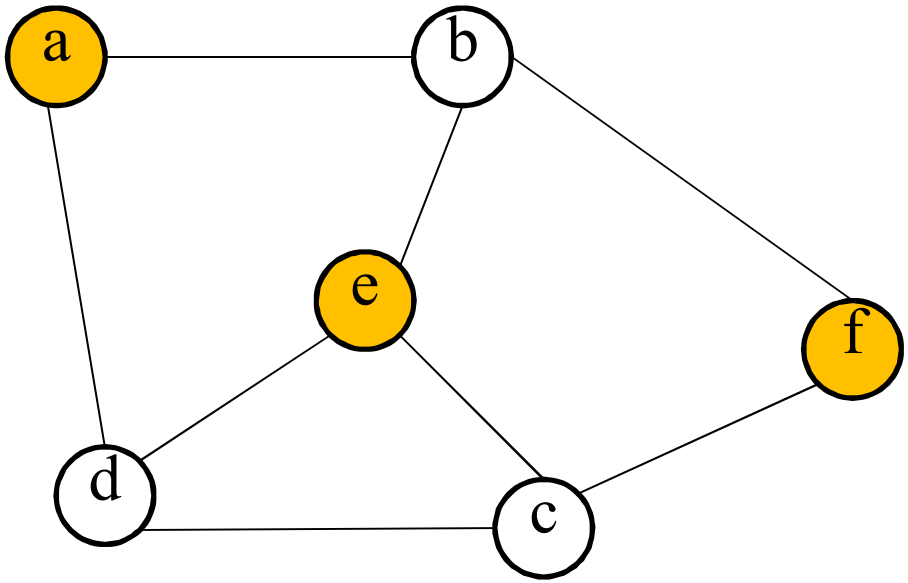
BÀI TOÁN TÔ MÀU

Greedy(G) //tìm một tập đỉnh nhiều nhất không có 2 đỉnh kề nhau

```
1   Newclr ← Ø
2   for each uncolored vertex v of G do
3       if  v is not adjacent to any vertex in Newclr
4           then  Newclr ← Newclr∪{v}  //greedy
5   return Newclr
```

BÀI TOÁN TÔ MÀU

- Sau khi thực hiện Greedy(G), Newclr={a, e, f}



BÀI TOÁN TÔ MÀU

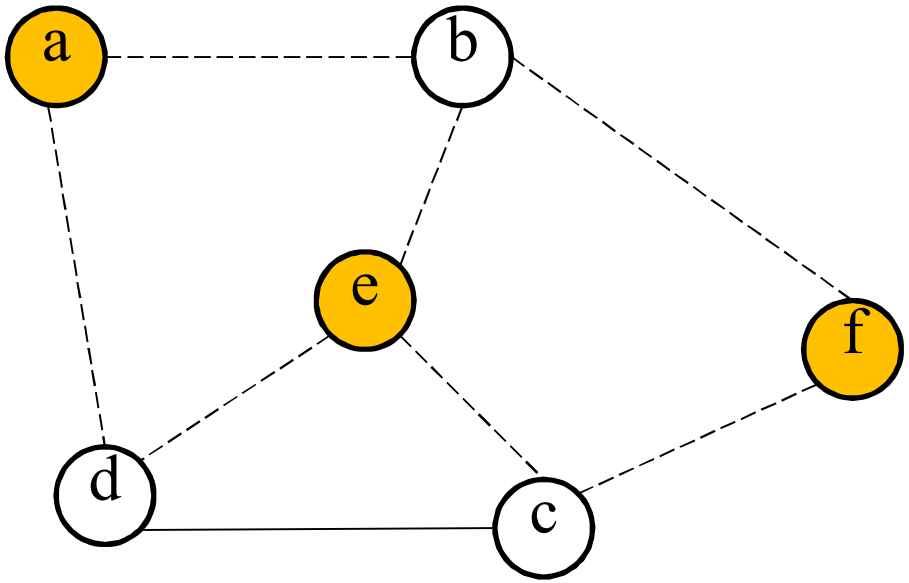
```
ColoringGraph(G)
1    C ← Ø; N ← Ø
2    while V[G] ≠ Ø  do
3            C ← Greedy(G)
4            Coloring every v in C the same color k∉N
5            V[G] ← V[G]-C
6            N ← N∪{k}
7    return N   // tập màu ít nhất có thể tô
```

BÀI TOÁN TÔ MÀU

- Kích thước đầu vào là số đỉnh n trong V[G]
- Thời gian chạy của Greedy(G) là O(n) (thao tác cơ bản)
- Thời gian các lệnh 4 và 5 không quá O(n)
- Vậy $T(n)=O(n^2)$

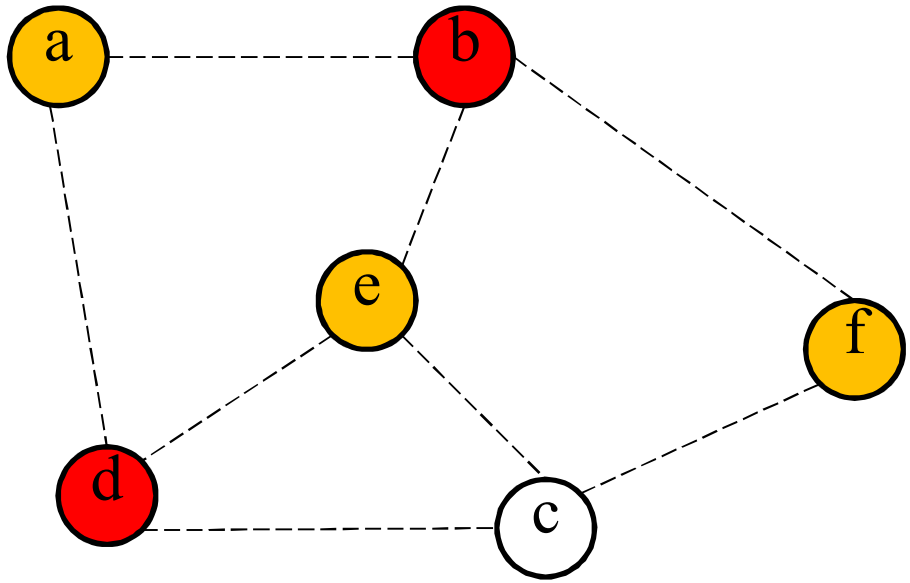
BÀI TOÁN TÔ MÀU

- Sau khi thực hiện Greedy(G) lần đầu (k là màu vàng)



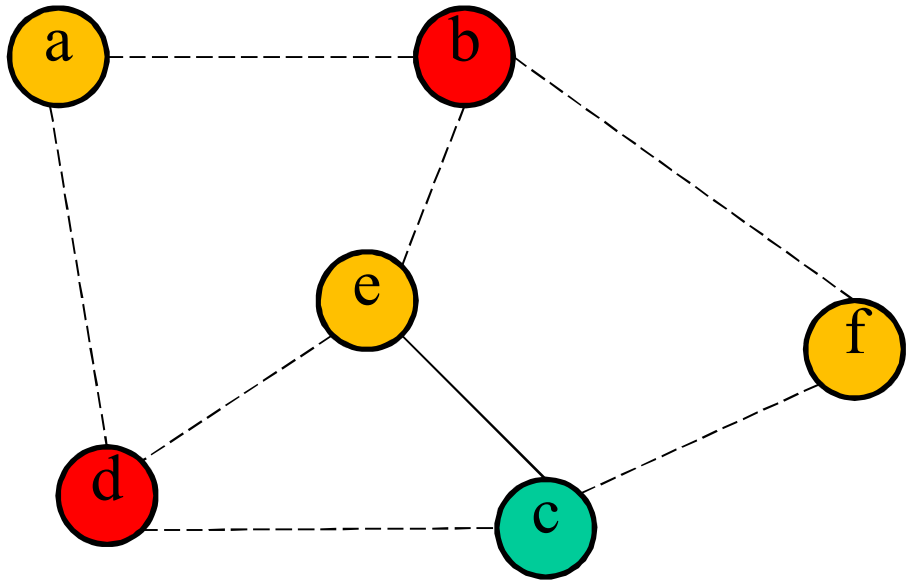
BÀI TOÁN TÔ MÀU

- Sau khi thực hiện Greedy(G) lần 2 (k là màu đỏ)



BÀI TOÁN TÔ MÀU

- Sau khi thực hiện Greedy(G) lần 3 (k là màu xanh)



BÀI TẬP VỀ NHÀ

- Đọc chương 9 Greedy Technique sách Levitin
- Làm bài tập về nhà đã cho trong DS bài tập
- Bài tập thực hành: Hiện thực giải thuật Dijkstra và giải thuật tô màu đồ thị