



THUẬT TOÁN ỨNG DỤNG - tài liệu môn thuật toán và ứng dụng

Thuật Toán Và Ứng Dụng (Trường Đại học Giao thông Vận tải)



Scan to open on Studeersnel

THUẬT TOÁN ÚNG DỤNG

I. Dijkstra , A*

• Dijkstra

Thuật toán Dijkstra hoạt động dựa trên việc tìm đường đi ngắn nhất đến một số đỉnh từ đỉnh xuất phát. Thuật toán này sử dụng một cấu trúc dữ liệu gọi là bảng định tuyến (routing table) để lưu trữ đường đi ngắn nhất hiện tại đến các đỉnh. Khi tìm đường đi ngắn nhất đến một đỉnh khác, thuật toán Dijkstra sẽ thay thế đường đi cũ bằng đường đi mới nếu đường đi mới có độ dài nhỏ hơn.

Thuật toán Dijkstra có độ phức tạp thời gian $O(V^2)$, trong đó V là số đỉnh trong đồ thị. Tuy nhiên, khi sử dụng cấu trúc dữ liệu heap để lưu trữ các đỉnh có thể được duyệt tới, thuật toán Dijkstra có thể được cải tiến để có độ phức tạp thời gian $O(E \log V)$, trong đó E là số cạnh trong đồ thị.

• Ý tưởng thuật toán Dijkstra:

+ Ý tưởng: Thực hiện một quá trình duyệt đồ thị xuất phát từ S trong đó với mỗi đỉnh N được xem xét ta tính số đo lô trình từ S đến N : $g(N)$

+ Các đỉnh được xem xét có thể được lưu ở 2 danh sách:

C = các đỉnh không xem xét lại

O = các đỉnh có thể được xem xét lại

+ Trong quá trình lặp để tìm đường đi ngắn nhất ta chọn đỉnh N thuộc O có $g(N)$ nhỏ nhất để tính các đỉnh kế tiếp.

Mỗi đỉnh tương ứng với 1 số $g(N)$. Trong đó $g(N)$ là chi phí thực tế đi từ đỉnh S ban đầu tới N

Close : tập đỉnh đóng

Open : tập đỉnh mở

Bước 1: Khởi tạo tập đỉnh mở với $\text{Open} = \{S\}$, $\text{Close} = \{\}$, $g(S) = 0$

Bước 2: Thực hiện vòng lặp while trong tập đỉnh mở

While ($O \neq \{\}$)

2.1 Chọn N thuộc O có $g(N)$ nhỏ nhất

2.2 Loại N ra khỏi O và thêm vào C

2.3 Nếu ($N == G$) \Rightarrow dừng và đưa ra kết luận

2.4. Xét các đỉnh S không thuộc C và kề với N

TH1: S là đỉnh mới (S không thuộc 0)

- $O = O \cup \{S\}$ (Thêm S vào O)
- $g(S) = g(N) + c(N,S)$ (khoảng cách từ So cho đến S sẽ bằng khoảng cách từ So tới N + khoảng cách từ N tới S)
- Ghi nhớ N là đỉnh trước S trên đường đi

TH2: S thuộc O

- Nếu $(g(S) > g(N) + c(N,S))$ (khoảng cách cũ lớn hơn khoảng cách mới)
- $g(S) = g(N) + c(N,S)$
- Ghi nhớ N là đỉnh trước S trên đường đi

TH3 : Nếu tất cả các đỉnh kề với N đều nằm trong C , bỏ N vào C rồi tiếp tục thực hiện vòng while

Bước 3: Kết luận bài toán.

Bài làm mẫu:

Cho khoảng cách theo đường bộ giữa các thành phố trong bảng sau:

* Lưu ý: Các giá trị trong bài toán chỉ mang tính giả thiết, không phải dữ liệu thật.

	Tp.HCM	Vũng Tàu	Tây Ninh	Thủ Dầu Một	Buôn Ma Thuột	Pleiku
Tp.HCM	0	116	0	30	345	0
Vũng Tàu	116	0	200	120	0	0
Tây Ninh	0	200	0	80	336	523
Thủ Dầu Một	30	120	80	0	0	497
Buôn Ma Thuột	345	0	336	0	0	187
Pleiku	0	0	523	497	187	0

a/ (2đ) Tìm đường đi ngắn nhất từ Tp.HCM đến Pleiku bằng thuật toán Dijkstra.

Dijkstra (Bài 1. a đề ôn)

BQ : Khởi tạo
~~open = {TP. HCM}~~
~~close = {}~~

Lần lượt đặt các thành phố TP. HCM, Vũng Tàu, Tây Ninh, Thủ Dầu Một, Buôn Ma Thuột, Pleiku là A, B, C, D, E, F

BQ. Khởi tạo : start = A, goal = F
~~open = {A}~~
~~close = {}~~
 $g(A) = 0$

B1. Chọn N = A

• Q = B : $g(B) = g(A) + \text{cost}(A, B) = 116$ prev(B) = A
• Q = D : $g(D) = g(A) + \text{cost}(A, D) = 30$ prev(D) = A
• Q = E : $g(E) = g(A) + \text{cost}(A, E) = 345$ prev(E) = A
open = {B, D, E}
close = {A}

B2. Chọn N = D

• Q = B : $g(B) = g(D) + \text{cost}(D, B)$
• Q = C : $g(C) = g(D) + \text{cost}(D, C) = 110$ prev(C) = D
• Q = F : $g(F) = g(D) + \text{cost}(D, F) = 527$ prev(F) = D
open = {B, F, C, F}
close = {A, D}

B3. Chọn N = C

Q = B $g(B) = g(C) + \text{cost}(C, B)$
Q = E $g(E) = g(C) + \text{cost}(C, E)$
Q = F $g(F) = g(C) + \text{cost}(C, F)$
open = {B, E, F}
close = {A, C, D}

B4. Chọn N = B

Q = B
open = {E, F}
close = {A, B, C, D}

B5. Chọn N = E

Q = F : $g(F) = g(E) + \text{cost}(E, F)$
open = {F}
close = {A, B, C, D, E}

B6. Chọn N = F

N = goal

Dùng
để dài đường đi ngắn nhất từ TP.HCM đến Pleiku
 $g(F) = 527$

Đường đi là: F

• A* (A – Star)

Thuật toán A* là một thuật toán tìm kiếm đường đi trong đồ thị hoặc lưới ô vuông, được sử dụng rộng rãi trong lĩnh vực trí tuệ nhân tạo và robot học. Thuật toán này kết hợp giữa thuật toán tìm kiếm theo chiều rộng (BFS) và thuật toán tìm kiếm theo chiều sâu (DFS), nhưng được cải tiến để tìm ra đường đi ngắn nhất từ điểm xuất phát đến điểm đích.

Ý tưởng :

Thuật toán A* sử dụng một hàm chi phí tổng cộng ($f(n)$) để ước lượng chi phí từ điểm xuất phát (start) đến điểm đích (goal) thông qua một đỉnh trung gian nào đó. Hàm $f(n)$ được định nghĩa như sau:

$$f(n) = g(n) + h(n)$$

Trong đó :

$g(n)$: chi phí thực tế từ điểm xuất phát đến đỉnh n.

$h(n)$: ước lượng chi phí từ đỉnh n đến đỉnh đích.

1. **Xác định chi phí thực tế $g(n)$** : Dùng để đo lường chi phí thực tế từ điểm xuất phát đến đỉnh n. Cập nhật giá trị $g(n)$ khi tìm thấy một đường đi tốt hơn.

2. **Ước lượng chi phí còn lại $h(n)$** : Đo lường ước lượng chi phí từ đỉnh n đến đỉnh đích. Điều này thường được thực hiện bằng cách sử dụng hàm heuristic, một hàm ước lượng không bao giờ lớn hơn chi phí thực tế. Heuristic giúp A* chọn đường đi tối ưu theo thời gian.
3. **Hàm tổng cộng $f(n)$** : Sử dụng hàm này để xác định đỉnh nào sẽ được mở tiếp theo. Đỉnh có giá trị $f(n)$ thấp nhất sẽ được chọn đầu tiên.
4. **Tìm kiếm theo chiều rộng và chiều sâu** : A* sử dụng một hàng đợi ưu tiên để mở rộng các đỉnh xung quanh, ưu tiên các đỉnh có $f(n)$ thấp nhất. Quá trình này tiếp tục cho đến khi đến được điểm đích hoặc không còn đỉnh nào để mở rộng.

Bước 1:

$$C = \{\};$$

$$O = \{S\};$$

$$g(S) = 0;$$

$$f(S) = h(S);$$

Bước 2:

while($O \neq \{\}$)

{

 2.1 Chọn $N \in O$ có $f(N)$ nhỏ nhất

 2.2 Lấy N từ O cho vào C

 2.3 if($N \equiv G$)

 Dừng. Kết luận: tìm được

 2.4 Xét các đỉnh kề Q của N

TH1: $Q \in O$

 if($g(Q) > g(N) + w(N, Q)$)

$g(Q) = g(N) + w(N, Q);$

$f(Q) = g(Q) + h(Q);$

$\text{prev}(Q) = N$

TH2: $Q \notin O$

$g(Q) = g(N) + w(N, Q);$

$f(Q) = g(Q) + h(Q);$

prev(Q)=N

}

Bước 3: Kết luận...

Bài ví dụ mẫu: (Đề giống của dijkstra nhưng thêm $h(x)$ được Phú Huy cho ngẫu nhiên)

A⁺, Dùng bài 1.a đề mục làm ví dụ, $h(i,j)$ là $h(i,j)$ và $h(A) = 137$, $h(B) = 229$, $h(C) = 103$, $h(D) = 114$, $h(E) = 171$.

B0: khởi tạo

start = A

goal = F

open = {A}

close = {}

$g(A) = 0$; ~~f(A)~~: $f(A) = 137$

B1: Chọn N = A.

$Q = B$: $g(B) = g(A) + \text{cost}(A,B) = 116$ prev(B) = A

$Q = D$: $g(D) = g(A) + \text{cost}(A,D) = 310$ prev(D) = A

$Q = E$: $g(E) = g(A) + \text{cost}(A,E) = 345$ prev(E) = A

$f(E) = g(E) + h(E) = 516$

Open = {B, D, E}

close = {A}

B2: Chọn N = D

$Q = B$: $g(B) = g(D) + \text{cost}(D,B) = 110$ prev(B) = D

$Q = C$: $g(C) = g(D) + \text{cost}(D,C) = 110$ prev(C) = D

$f(C) = g(C) + h(C) = 110 + 103 = 213$

$Q = F$: $g(F) = g(D) + \text{cost}(D,F) = 527$ prev(F) = D

$f(F) = g(F) + h(F) = 527 + 0 = 527$

Open = {B, E, C, F}

close = {A, D}

B3: Chọn N = C

$Q = B$: $g(B) < g(C) + \text{cost}(C,B)$

$Q = E$: $g(E) < g(C) + \text{cost}(C,E)$

$Q = F$: $g(F) < g(C) + \text{cost}(C,F)$

Open = {B, E, F}

close = {A, D, C}

B4: Chọn N = B

Open = {E, F}

close = {A, D, C, B}

B5: Chọn N = E

$Q = F$: $g(F) < g(E) + \text{cost}(E,F)$

Open = {F}

close = {A, D, C, B, E}

B6: Chọn N = F

N == goal

Dừng

đo dài đường đi ngắn nhất, là $g(F) = 527$
đường đi là: $F \leftarrow D \leftarrow A$

II. Prim , Kruskall

1. Prim

Cây khung (spanning tree) của một đồ thị là một đồ thị con liên thông không có chu trình đi qua tất cả các đỉnh. Một đồ thị sẽ có nhiều cây khung và bài toán của chúng ta là phải tìm ra cây khung nhỏ nhất.

Ý tưởng cơ bản của thuật toán Prim như sau:

- Bước 0: Khởi tạo tập T_0 , tập S_0 , tập $f(S_0)$. Trong đó:

$$T_0 = \emptyset$$

$S_0 = \{\text{đỉnh}\}$, có thể chọn đỉnh bất kì

$f(S_0)$: tập cạnh nối từ S_0 ra ngoài

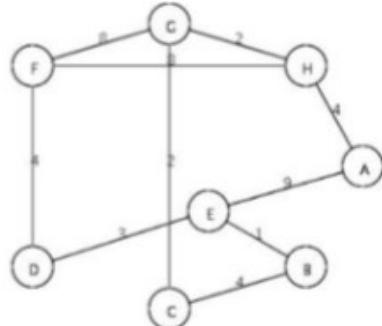
Tổng quát :

- Bước i:
 - $T_i = T_{i-1} + e_{i-1}$
với e_{i-1} có trọng số bé nhất trong $f(S_{i-1})$
 - $S_i = S_{i-1} + \{v'\}$ với $e_{i-1} = (v, v')$
- T_{n-1} là cây bao trùm tối thiểu

Bài Tập Làm Mẫu

Câu 2: (3 điểm)

- a/ (1đ) Trình bày thuật toán Prim.
b/ (2đ) Tìm cây khung tối thiểu của đồ thị bên
bằng thuật toán Prim:



Phim.

$$\text{Bước 0: } T_0 = \{G\}$$

$$S_0 = \{G\}$$

$$f(S_0) = \{GF(8), GH(2), GI(2)\}$$

$$\text{Bước 1: } T_1 = T_0 + G(2) = \{GC\}$$

$$S_1 = \{G, C\}$$

$$f(S_1) = \{GF(8), GH(2), CB(4)\}$$

$$\text{Bước 2: } T_2 = T_1 + GI(2) = \{GC, GH\}$$

$$S_2 = \{G, C, H\}$$

$$f(S_2) = \{GF(8), CB(4), HF(8), HA(4)\}$$

$$\text{Bước 3: } T_3 = T_2 + CB = \{GC, GH, CB\}$$

$$S_3 = \{G, C, H, B\}$$

$$f(S_3) = \{GF(8), HF(8), HA(4), EB(1)\}$$

$$\text{Bước 4: } T_4 = T_3 + EB = \{GC, GH, CB, EB\}$$

$$S_4 = \{G, C, H, B, E\}$$

$$f(S_4) = \{GF(8), HF(8), HA(4), ED(3), EA(9)\}$$

$$\text{Bước 5: } T_5 = T_4 + ED = \{GC, GH, CB, EB, ED\}$$

$$S_5 = \{G, C, H, B, E, D\}$$

$$f(S_5) = \{GF(8), HF(8), HA(4), \boxed{ED}, EA(9), DF\}$$

$$\text{Bước 6: } T_6 = T_5 + HA = \{GC, GH, CB, EB, ED, HA\}$$

$$S_6 = \{G, C, H, B, E, D, A\}$$

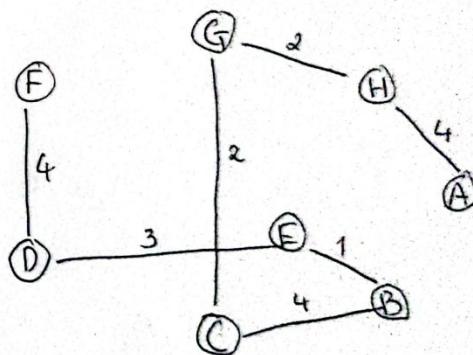
$$f(S_6) = \{GF(8), HF(8), EA, DF(4)\}$$

$$\text{Bước 7: } T_7 = T_6 + DF(4) = \{GC, GH, CB, EB, ED, HA, DF\}$$

$$S_7 = \{G, C, H, B, E, D, A, F\}$$

$$f(S_7) = \{GF(8), HF(8), EA(9)\}$$

Cây $T_7 = \{GC, GH, CB, EB, ED, HA, DF\}$ là rãy bao
nhóm tối thiểu với $w(T_7) = 20$



Đến

2. Kruskall

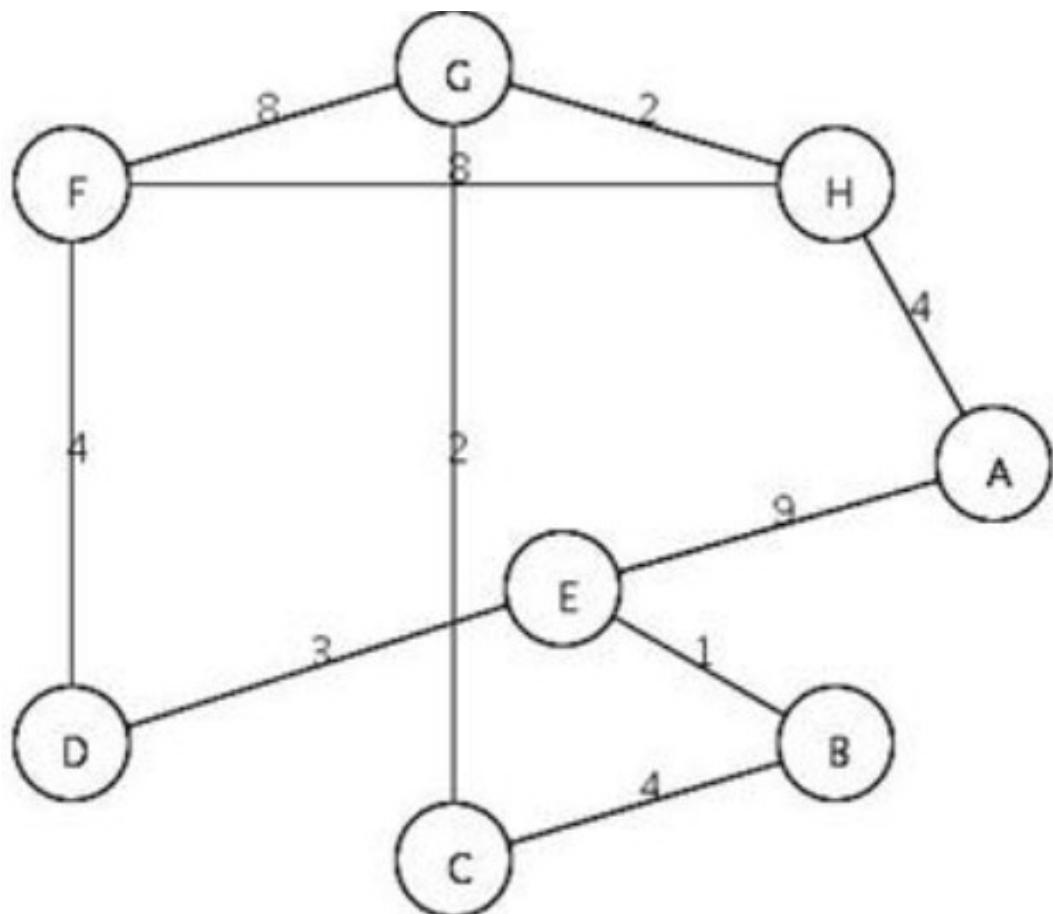
- Thuật toán Kruskal là một thuật toán tham lam được sử dụng để tìm cây khung nhỏ nhất trong một đồ thị vô hướng, liên thông và có trọng số. Mục tiêu của thuật toán là tìm một tập hợp các cạnh không tạo thành chu trình và có trọng số tổng nhỏ nhất.

Ý tưởng cơ bản:

Thuật giải Krusal

- i) Sắp xếp các cạnh của G theo thứ tự tăng dần về trọng số
- ii) Xây dựng cây bao trùm tối thiểu:
Bước 0: $T_0 = \emptyset$
Bước i: $T_i = T_{i-1} + e_j$
 $e_j : \begin{cases} \text{không tạo thành chu trình cho } T_i \\ j: \text{chỉ số nhỏ nhất} \end{cases}$
- iii) T_{n-1} là cây bao trùm tối thiểu

Bài Tập Làm Mẫu



Kruskal (Đề xài đề thi nong dề màu)

EB	GC	GH	DE	FD	HA	CB	GF	FH	EA
1	2	2	3	4	4	4	8	8	9

Bước 0: $T_0 = \{ \}$

Bước 1: $T_1 = T_0 + EB = \{ EB \}$

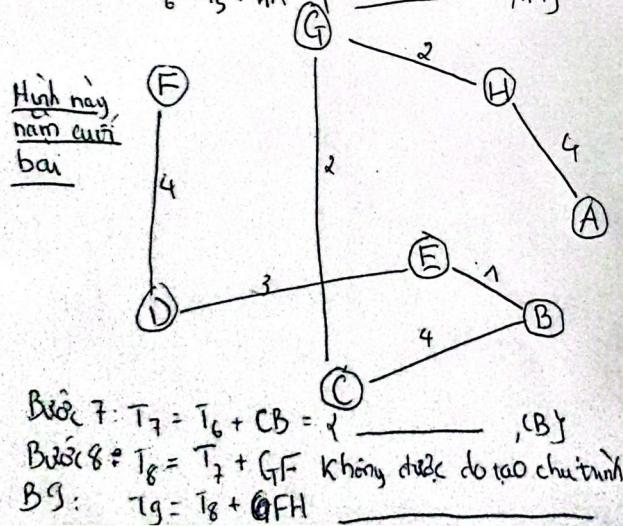
Bước 2: $T_2 = T_1 + GC = \{ EB, GC \}$

Bước 3: $T_3 = T_2 + GH = \{ EB, GC, GH \}$

Bước 4: $T_4 = T_3 + DE = \{ EB, GC, GH, DE \}$

Bước 5: $T_5 = T_4 + FD = \{ EB, GC, GH, DE, FD \}$

Bước 6: $T_6 = T_5 + HA = \{ EB, GC, GH, DE, FD, HA \}$



B.10: $T_{10} = T_9 + EA$ K^o được tạo chu trình

III. Ford-Fulkerson

- Thuật toán Ford-Fulkerson

Thuật toán Ford-Fulkerson là một thuật toán tìm kiếm theo chiều sâu (DFS) và không được đảm bảo hoạt động chính xác trên đồ thị có trọng số âm. Điều này là do khi tìm đường tăng luồng trên đồ thị có trọng số âm, thuật toán có thể bị mắc kẹt trong vòng lặp vô hạn, dẫn đến không tìm được kết quả đúng.

Dưới đây là mô tả sơ lược về cách thức giải quyết bài toán bằng thuật toán Ford-Fulkerson:

- Tìm luồng cực đại

1. Khởi tạo luồng ban đầu bằng 0.
2. Tìm một đường đi từ đỉnh nguồn tới đỉnh đích trong đồ thị.
3. Tìm luồng nhỏ nhất trên đường đi vừa tìm được.

4. Tăng luồng trên các cạnh trên đường đi đó bằng lượng luồng nhỏ nhất vừa tìm được.
5. Lặp lại bước 2 đến khi không còn đường đi nào từ đỉnh nguồn tới đỉnh đích trong đồ thị.

- **Tìm lát cắt cực tiểu**

Sau khi thực hiện thuật toán Ford-Fulkerson để tìm luồng cực đại, ta có thể sử dụng một số phương pháp khác để tìm lát cắt cực tiểu.

Một phương pháp đơn giản là sử dụng thuật toán DFS để tìm các đỉnh nằm trong lát cắt. Cụ thể:

1. Khởi tạo tập S bằng đỉnh source và tập T bằng toàn bộ các đỉnh còn lại trong đồ thị.
2. Khởi tạo mảng `visited` với giá trị `False` tương ứng với tất cả các đỉnh.
3. Thực hiện DFS trên đỉnh nguồn, đánh dấu `visited` cho các đỉnh được truy cập.
4. Với mỗi đỉnh u đã được truy cập và có cạnh nối đến một đỉnh v chưa được truy cập và có trọng số dương, ta đưa v vào tập S .
5. Lặp lại bước 3 và 4 cho đến khi không còn đỉnh nào được thăm.
6. Lát cắt cực tiểu là cạnh giữa tập S và tập T .

Ford-Fulkerson theo slide của thầy, chỉnh sửa xíu chỗ nâng luồng

Để tìm luồng cực đại của mạng vận tải G , ta xuất phát từ luồng tuỳ ý ϕ của G , rồi nâng luồng lên đầy, sau đó áp dụng thuật toán Ford-Fulkerson.

Thuật toán gồm 3 bước:

Bước 1 (đánh dấu ở đỉnh của mạng):

Chúng ta lần lượt đánh dấu các đỉnh v_i được chọn bằng cặp chỉ số $\pm v_i, d_{v_i}$. Trong đó v_i là lượng tăng luồng nhỏ nhất từ đỉnh xuất phát đến đỉnh hiện tại.

Lối vào v_0 được đánh dấu bằng v_0, ∞ . Nếu đỉnh v_i đã được đánh dấu thì ta dùng cặp chỉ số chỉ số $+v_i, \min(d, c_{v_i,y} - f_{v_i,y})$ để đánh dấu cho mọi đỉnh y chưa được đánh dấu mà $(v_i, y) \in E$ và cung này chưa bão hòa ($\phi(v_i, y) < m(v_i, y)$). Nếu đỉnh v_i đã được đánh dấu thì ta dùng chỉ số $-v_i, \min(dx, f_{z,v_i})$ để đánh dấu cho mọi đỉnh z chưa được đánh dấu mà $(z, v_i) \in E$ và luồng của cung này dương ($\phi(z, v_i) > 0$).

Nếu với phương pháp này ta đánh dấu được tới lối ra v_n thì trong G tồn tại giữa v_0 và v_n một xích α , mọi đỉnh đều khác nhau và được đánh dấu theo chỉ số của đỉnh liền trước nó (chỉ sai khác nhau về dấu). Khi đó chắc chắn ta nâng được giá trị của luồng.

Bước 2 (nâng giá trị của luồng): Để nâng giá trị của luồng φ , ta đặt:

Đặt b là d_{v_n}

$$\varphi'(e) = \varphi(e), \text{ nếu } e \notin \alpha$$

$$\varphi'(e) = \varphi(e)+b, \text{ nếu } e \in \alpha \text{ được định hướng theo chiều của xích } \alpha \text{ đi từ } v_0 \text{ đến } v_n$$

$$\varphi'(e) = \varphi(e)-b, \text{ nếu } e \in \alpha \text{ được định hướng ngược với chiều của xích } \alpha \text{ đi từ } v_0 \text{ đến } v_n$$

φ' thoả mãn các điều kiện về luồng, nên φ' là một luồng và ta có: $\varphi' n = \varphi n + b$.

Như vậy, ta đã nâng được luồng lên b vị. Sau đó lặp lại một vòng mới. Vì khả năng thông qua của các cung đều hữu hạn, nên quá trình phải dừng lại sau một số hữu hạn bước.

Bước 3:

Nếu với luồng φ_0 bằng phương pháp trên ta không thể nâng giá trị của luồng lên nữa, nghĩa là ta không thể đánh dấu được đỉnh v_n , thì ta nói rằng quá trình nâng luồng kết thúc và φ_0 đã đạt giá trị cực đại, đồng thời gọi φ_0 là luồng kết thúc.

Tăng luồng f dọc theo đường tăng P

Augment(f, P)

```
b ← cf(P) = min{cf(u, v) : (u, v) ∈ p}
FOR e ∈ P DO
    IF (e ∈ E) THEN // cạnh thuận
        f(e) ← f(e) + b
    ELSE // cạnh nghịch
        f(eR) ← f(e) - b
RETURN f
```

Ví dụ

Thuật toán Ford-Fulkerson

```
Ford_Fulkerson(G, c, s, t);
FOR e ∈ E DO // Khởi tạo luồng 0
    f(e) ← 0
Gf ← đồ thị tăng luồng f

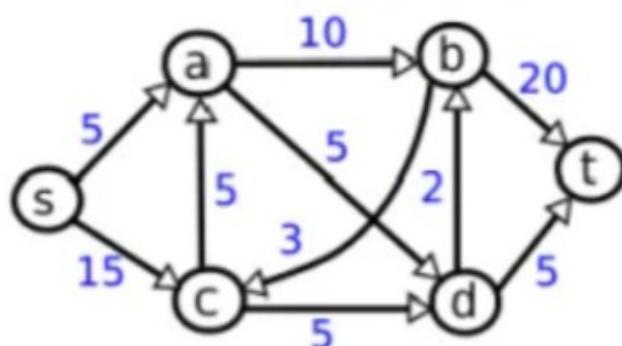
WHILE (tìm được đường tăng luồng P) DO
    f ← augment(f, P)
    Sửa lại Gf
RETURN f
```

Code:
<http://www.giaithuatlaptrinh.com/?p=1483>

Bài Tập Làm Mẫu

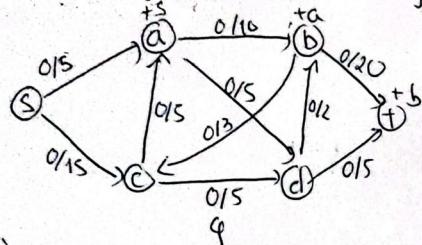
Câu 3: (3.5 điểm)

- a/ (1đ) Trình bày thuật toán Ford —Fulkerson cho bài toán tìm luồng cực đại.
b/ (2.5đ) Giải bài toán mạng vận tải sau với luồng vận tải khởi đầu bằng 0:

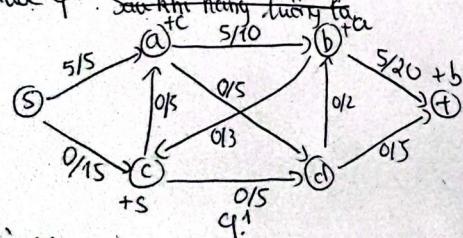


Ford-Fulkerson

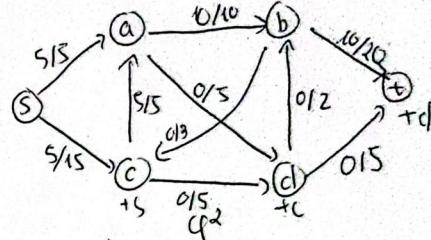
Mạng vận tải G có đỉnh phát là S và đỉnh thu là T
Khởi tạo luồng ban đầu bằng 0, ta có ϕ^0



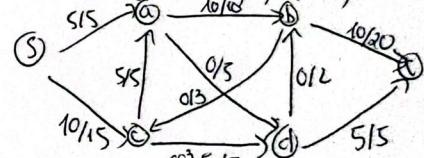
Luồng ϕ^0 có đường đi $(S, A), (A, B), (B, T)$ gồm các cung chưa bão hòa. Ta có thể nâng luồng của các cung này lên một khoảng bằng min($c_f(S, A), c_f(A, B), c_f(B, T)$) = 5 để được ϕ^1 . Sau khi nâng luồng ta



Luồng ϕ^1 có đường đi $(S, C), (C, A), (A, B), (B, T)$ gồm các cung chưa bão hòa. Ta có thể nâng luồng của các cung này lên một khoảng bằng min($c_f(S, C), c_f(C, A), c_f(A, B), c_f(B, T)$) = 5 để được ϕ^2 .



Luồng này có đường đi $(S, C), (C, D), (D, T)$ gồm các cung chưa bão hòa. Ta có thể nâng luồng của các cung này lên một khoảng bằng min($c_f(S, C), c_f(C, D), c_f(D, T)$) = 5, để được ϕ^3



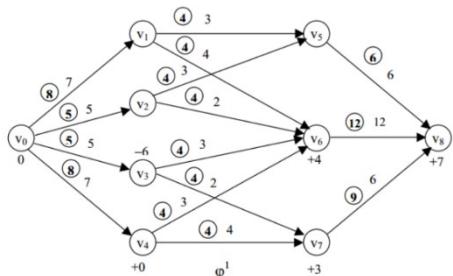
Tiếp theo ta không tìm được đường đi nào nữa nên luồng này đầy và ta chỉ có thể đánh dấu cung định S nên ta chia luồng ra làm hai

$$C_{\text{t}}^2 = 10 + 5 = 15$$

* Nếu k^o tìm dc đường đi tiếp nhung vẫn đánh dấu đc rồi ta lì luận như hình (1) bên dưới

* Lì luận chia luồng sau khi lý luận như hình (2) bên dưới

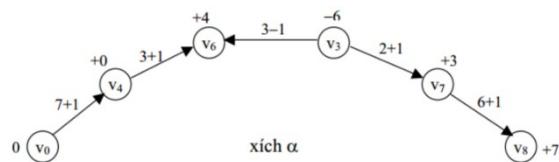
- Sau khi nâng luồng ta có ϕ^1



- Do mỗi đường xuất phát từ v_0 đến v_8 đều chứa ít nhất một cung bão hòa, nên luồng ϕ^1 là luồng đầy. Song nó chưa phải là luồng cực đại.

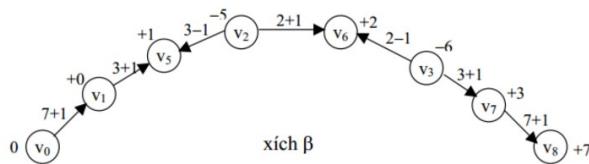
NHUYỄN HỮU NGHĨA

- Áp dụng thuật toán Ford-Fulkerson để nâng luồng ϕ^1
- Xét xích $\alpha = (v_0, v_4, v_6, v_3, v_7, v_8)$



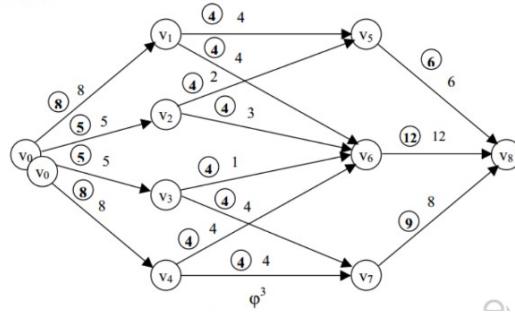
- Quá trình đánh dấu từ v^0 đến v^8 để có thể nâng luồng ϕ^1 lên một đơn vị bằng cách biến đổi luồng tại các cung thuộc xích α được đánh dấu. Sau đó ta có luồng ϕ^2

- ④ Xét xích $\beta = (v_0, v_1, v_5, v_2, v_6, v_3, v_7, v_8)$



- ⑤ Quá trình đánh dấu từ v_0 đến v_8 để có thể nâng luồng φ^2 lên một đơn vị bằng cách biến đổi luồng tại các cung thuộc xích β được đánh dấu. Sau đó ta có luồng φ^3 .

Luồng φ^3



Tiếp theo ta chỉ có thể đánh dấu được đỉnh v_0 nên quá trình nâng luồng kết thúc và ta được giá trị của luồng cực đại là: $\varphi_{v_8}^3 = 6+12+8 = 26$.

IV. Heuristic

1. Định nghĩa :

- Là giải pháp được viết dưới dạng thủ tục tương tự như thuật toán nhưng không đòi hỏi các tiêu chuẩn như thuật toán.
- Tính đúng: chấp nhận các thuật giải đơn giản có thể cho kết quả đúng hay gần đúng nhưng có khả năng thành công cao hơn.
- Để có thể được chấp nhận, thuật giải phải thể hiện một giải pháp hợp lý nhất có thể trong tình huống hiện tại bằng cách:
 - Tận dụng mọi thông tin hữu ích
 - Sử dụng tri thức, kinh nghiệm trực giác của con người
 - Tự nhiên đơn giản nhưng cho kết quả chấp nhận được

➤ Bài toán Phân công công việc

Dựa theo kinh nghiệm của con người: Ta sẽ lấy công việc tốn nhiều thời gian nhất phân công cho máy làm việc ít nhất.

=> Nguyên lý thứ tự

```
for(i = 0; i < n; i++){
    • Chọn công việc J chưa được phân mà tốn nhiều
    • thời gian thực hiện nhất
    • Chọn máy M làm việc ít nhất
    • Phân công J cho M
}
```

Ví dụ : Giả sử có n công việc và phân cho m máy trong 1 xí nghiệp

vd: $n = 10, m = 3$
tj: 4 9 5 2 7 6 10 8 7 5

-> tj sắp xếp: 10 9 8 7 7 6 5 5 4 2

M1	10	6	5	
M2	9	7	4	2
M3	8	7	5	

Thuật giải heuristic cho bài toán phân công công việc có độ phức tạp là $O(n \lg n + m^2)$ thấp hơn $O(m^n)$

➤ Bài toán Tô màu đồ thị

- Sử dụng nguyên lí thứ tự

Thuật toán: Ta có tập V là tập các đỉnh, V' là tập các đỉnh được sắp xếp giảm dần theo bậc của đỉnh.

while($V' \neq$ rỗng) {

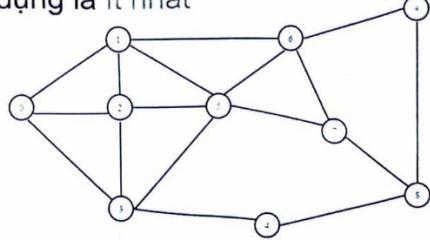
- Chọn đỉnh 5 đầu danh sách trong V'
- Chọn màu i
- Tô màu i cho đỉnh S
- Duyệt các đỉnh còn lại trong V' không kề với S và tô màu i, mỗi lần tô màu cho một đỉnh ta đánh dấu đỉnh đó để không tô màu i cho các đỉnh lân cận
- Loại tất cả đỉnh vừa tô ra khỏi V'

}

Ví dụ:

vd:

U sú dụng là ít nhất



V:

Đỉnh:	0	1	2	3	4	5	6	7	8	9
Bậc:	3	4	4	4	2	5	4	3	3	2

V':

Đỉnh:	5	1	2	3	6	0	7	8	4	9
Bậc:	5	4	4	4	4	3	3	3	2	2

- B1: Tô màu 1 (xanh lam) cho đỉnh 5, lần lượt duyệt qua các đỉnh còn lại trong V', đỉnh 1 2 3 6 7 kề với 5 nên ta sẽ không xét

+ Tô màu 1 cho đỉnh 0

+ Tô màu 1 cho đỉnh 8

+ Đỉnh 4 kề với đỉnh 8 đã tô -> không tô

+ Đỉnh 9 kề với đỉnh 8 đã tô -> không tô

Sau bước 1 ta có 5 0 8 tô màu 1 => loại 5 0 8 ra khỏi V' ta được V':

Đỉnh:	1	2	3	6	7	4	9
Bậc:	4	4	4	3	2	2	

- B2: Tô màu 2 (cam) cho đỉnh 1, lần lượt duyệt qua các đỉnh con lại trong V', đỉnh 2 6 kề với 1 nên ta sẽ không xét:

+ Tô màu 2 cho đỉnh 3

+ Tô màu 2 cho đỉnh 7

+ Đỉnh 4 kề với đỉnh 3 đã tô -> không tô

+ Tô màu 2 cho đỉnh 9

Sau bước 2 ta có 1 3 7 9 được tô màu 2 => loại 1 3 7 9 ra khỏi V', cập nhật lại V':

Đỉnh:	2		6
Bậc:	4		4

- B3: Tô màu 3 (tím) cho đỉnh 2, lần lượt duyệt qua các đỉnh còn lại trong V':

+ Tô màu 2 cho đỉnh 6

+ Tô màu 2 cho đỉnh 4

Sau B3 ta có 2 6 4 được tô màu 3 => loại 2 6 4 ra khỏi V', cập nhật lại V':

V' == rỗng => Dừng thuật toán

Ta được:

Màu 1: 5 0 8

Màu 2: 1 3 7 9

Màu 3: 2 4 6

➤ Bài toán Người du lịch

- Dựa theo kinh nghiệm của con người: Khi đi trên những đoạn đường ngắn nhất thì cuối cùng ta sẽ có một hành trình ngắn nhất.

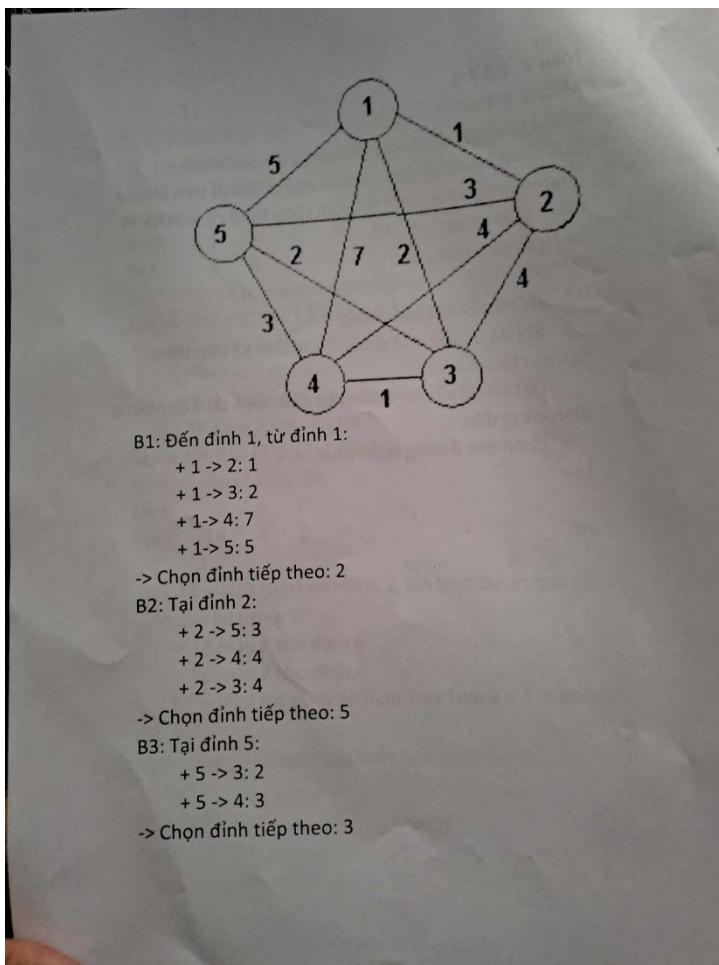
=> Nguyên lý tham lam

While(Chưa đi hết các đỉnh){

- Khi đã đi đến một đỉnh, chọn đỉnh kế tiếp theo nguyên tắc:
- Liệt kê tất cả con đường từ đỉnh hiện tại đến những đỉnh chưa đến.
- Chọn con đường ngắn nhất.

}

Ví dụ:



B4: Tại đỉnh 3:

+ 3 -> 4: 1

-> Chọn đỉnh tiếp theo: 4

B5: Đã qua ta các đỉnh -> Dừng thuật toán

Ta được đường đi: 1 -> 5 -> 3 -> 4 -> 1 với độ dài:

$$1+3+2+1+7 = 14$$

Độ phức tạp: $O(n^2)$ thấp hơn so với $O(n!)$

d. Bài toán cái túi:

➤ Bài toán Balo

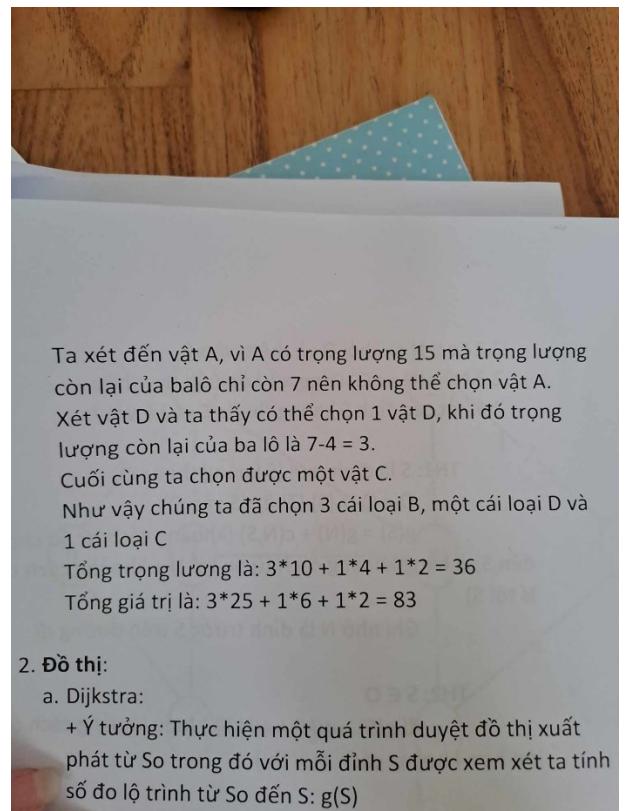
- Theo tri thức của con người: từ giá trị và khối lượng ta có thể tính ra được tỉ lệ giá trị/ khối lượng mỗi đồ vật và tạm gọi tỉ lệ này là đơn giá, sau đó sắp xếp đồ vật theo đơn giá giảm dần ta sẽ có thể áp dụng phương pháp tham lam để chọn đồ vật cho vào túi.

Thuật giải áp dụng:

- + Tính đơn giá cho các loại đồ vật.
- + Xét các loại đồ vật theo thứ tự đơn giá từ lớn đến nhỏ.
- + Với mỗi đồ vật được xét sẽ lấy một số lượng tối đa mà trọng lượng còn lại của ba lô cho phép.
- + Xác định trọng lượng còn lại của ba lô và quay lại

Ví dụ:

nào nưa.
Loại đồ vật : A - B - C - D
Trọng lượng : 15 - 10 - 2 - 4
Giá trị : 30 - 25 - 2 - 6
Từ bảng đã cho ta tính đơn giá cho các loại đồ vật và sắp xếp các loại đồ vật này theo thứ tự đơn giá giảm dần ta có bảng sau.
Loại đồ vật : B - A - D - C
Trọng lượng : 10 - 15 - 4 - 2
Giá trị : 25 - 30 - 6 - 2
Đơn giá : 2.5 - 2.0 - 1.5 - 1.0
Theo đó thì thứ tự ưu tiên để chọn đồ vật là B, A, D và cuối cùng là C.
Vật B được xét đầu tiên và ta chọn tối đa 3 cái vì mỗi cái vì trọng lượng mỗi cái là 10 và ba lô có trọng lượng 37. Sau khi đã chọn 3 vật loại B, trọng lượng còn lại trong ba lô là $37 - 3 \times 10 = 7$



V. Tìm kiếm

1. Tìm kiếm theo chiều rộng (BFS)

- $O = [S]; C=\{\}$;
- While O khác rỗng
- {
 - Lấy p từ đầu O
 - Duyệt p
 - Bỏ p vào C
 - Với mỗi q kề p, q không thuộc C: bỏ q vào cuối O
}



• **Định nghĩa:**

- Thuật toán duyệt đồ thị ưu tiên chiều rộng (Breadth-first search - BFS) là một trong những thuật toán tìm kiếm cơ bản và thiết yếu trên đồ thị. Mà trong đó, những đỉnh nào gần đỉnh xuất phát hơn sẽ được duyệt trước.
- Tìm kiếm theo chiều rộng (BFS) là một thuật toán để duyệt đồ thị hoặc cây. BFS áp dụng cho cây và đồ thị gần như giống nhau. Sự khác biệt duy nhất là đồ thị có thể chứa các chu trình, vì vậy chúng ta có thể duyệt lại cùng một nút. Để tránh xử lý lại cùng một nút, chúng ta sử dụng mảng boolean đã truy cập, mảng này sẽ đánh dấu các đỉnh đã truy cập. BFS sử dụng cấu trúc dữ liệu hàng đợi (queue) để tìm đường đi ngắn nhất trong biểu đồ.

• **Ý tưởng :**

Bước 1 : Khởi tạo tập Open = {S} (giả sử S là đỉnh ban đầu), Close = \emptyset

Bước 2 : Thực hiện vòng lặp while trong tập Open :

2.1 : Chọn p ở đầu tập Open

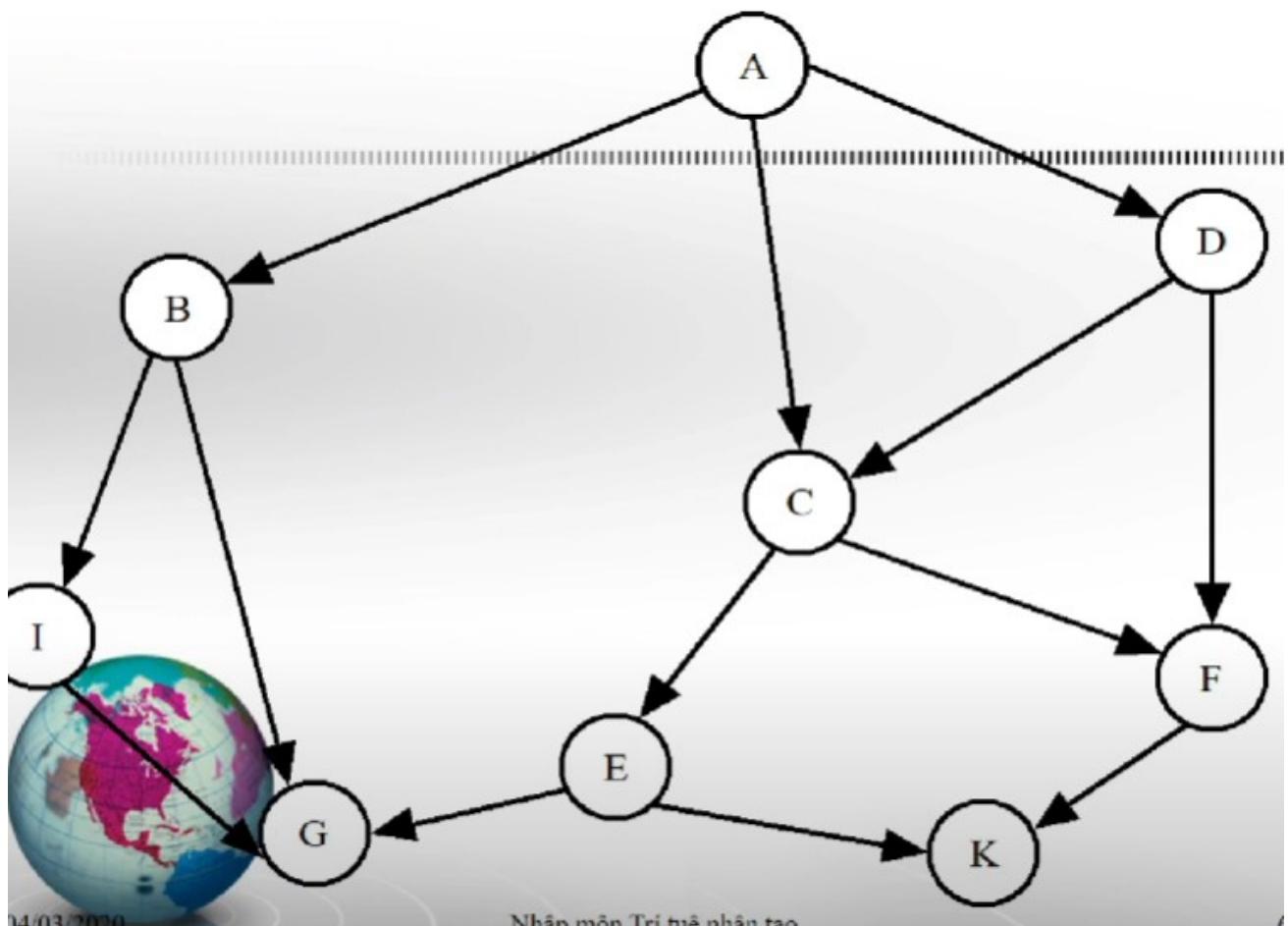
2.2 : Duyệt p

2.3 : Bỏ p vào tập Close

2.4 : Với mỗi q kề p , với q không thuộc Close thì bỏ q vào cuối tập Open

Bước 3 : Cập nhật lại tập Open và Close

Bài Tập Mẫu.



BFS

B khởi tạo $O = \{A\}$

$$C = \{\}$$

B1: $p = A$

$$C = \{A\}$$

$$O = \{B, C, D\}$$

B2: $p = B$

$$C = \{A, B\}$$

$$O = \{C, D, I, G\}$$

B3: $p = C$

$$C = \{A, B, C\}$$

$$O = \{D, I, G, E, F\}$$

B4: $p = D$

$$C = \{A, B, C, D\}$$

$$O = \{I, G, E, F\}$$

B5: $p = I$

$$C = \{A, B, C, D, I\}$$

$$O = \{G, E, F\}$$

B6: $p = G$

$$C = \{A, B, C, D, I, G\}$$

$$O = \{E, F\}$$

B7. $p = E$

$$C = \{A, B, C, D, I, G, E\}$$

$$O = \{F, K\}$$

B8: $p = F$

$$C = \{A, B, C, D, I, G, E\}$$

$$O = \{K\}$$

B9: $p = K$

$$C = \{A, B, C, D, I, G, E\}$$

$$O = \{\}$$

O rỗng nên chung

2. Tìm kiếm theo chiều sâu (DFS)

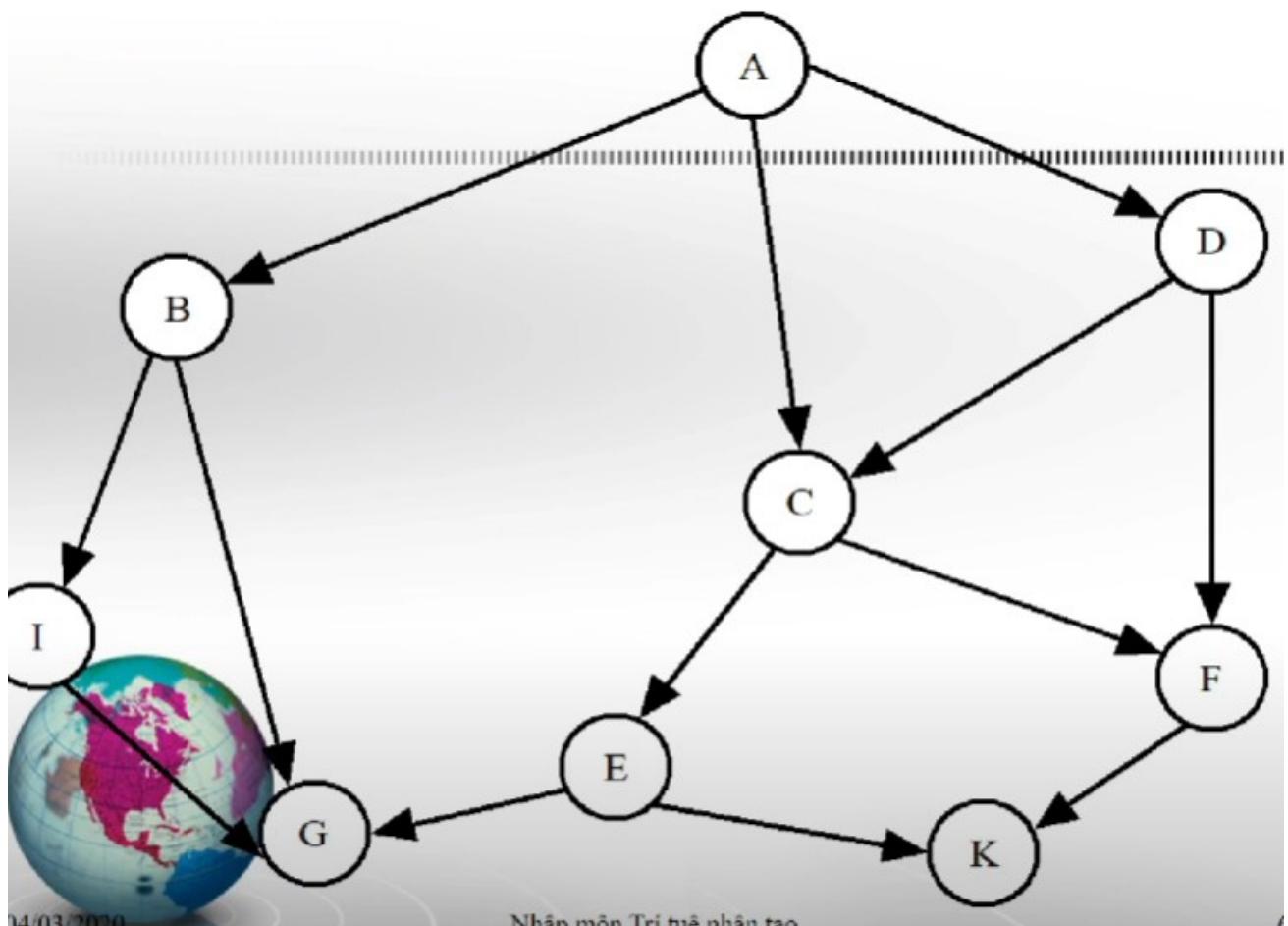
Tìm kiếm theo chiều sâu

- $O = [S]; C=\{\}$;
- While O khác rỗng
- {
 - Lấy p từ đầu O
 - Duyệt p
 - Bỏ p vào C
 - Với mỗi q kề p , q không thuộc C : bỏ q vào đầu O
- }



- **Định nghĩa:**
 - Tìm kiếm theo chiều sâu (DFS) là một thuật toán để duyệt qua hoặc tìm kiếm cấu trúc dữ liệu dạng cây hoặc đồ thị. Thuật toán bắt đầu tại nút gốc (chọn một số nút tùy ý làm nút gốc trong trường hợp đồ thị) và kiểm tra từng nhánh càng xa càng tốt trước khi quay lui.
 - Kết quả của một DFS là một cây bao trùm (spanning tree). Cây khung (spanning tree) là một đồ thị không có vòng lặp.
- **Ý tưởng :**
 - Bước 1 :** Khởi tạo tập Open = $\{S\}$ (giả sử S là đỉnh ban đầu), Close = \emptyset
 - Bước 2 :** Thực hiện vòng lặp while trong tập Open :
 - 2.1 :** Chọn p ở đầu tập Open
 - 2.2 :** Duyệt p
 - 2.3 :** Bỏ p vào tập Close
 - 2.4 :** Với mỗi q kề p , với q không thuộc Close thì bỏ q vào đầu tập Open
 - Bước 3 :** Cập nhật lại tập Open và Close

Bài Tập Mẫu



DFS

Bước 1: $O = \{A\}$

$C = \{G\}$

Bước 1: $p = A$

$C = \{A\}$

$O = \{B, C, D\}$

Bước 2: $p = B$

$C = \{A, B\}$

$O = \{I, G, C, D\}$

Bước 3: $p = I$

$C = \{A, B, I\}$

$O = \{G, C, D\}$

Bước 4: $p = G$

$C = \{A, B, I, G\}$

$O = \{C, D\}$

Bước 5: $p = C$

$C = \{A, B, I, G, C\}$

$O = \{E, F, D\}$

Bước 6: $p = E$

$C = \{A, B, I, G, C, E\}$

$O = \{K, F, D\}$

Bước 7: $p = K$

$C = \{A, B, I, G, C, E, K\}$

$O = \{F, D\}$

Bước 8: $p = F$

$C = \{A, B, I, G, C, E, K\}$

$O = \{D\}$

Bước 9: $p = D$

$C = \{A, B, I, G, C, E, K\}$

$O = \{G\}$

vì O rỗng nên dừng

$C = \{A, B, I, G, C, E, K, F, D\}$

VI. Cây quyết định, mạng neural.

1. Cây quyết định

- Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượng có thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary), Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.

- Có hai thuật toán để tạo một cây quyết định :

1. CART (Classification and Regression Trees) → dùng Gini Index(Classification) để kiểm tra.
2. ID3 (Iterative Dichotomiser 3) → dùng Entropy function và Information gain để kiểm tra.

2. Mạng neural (Neural network)

- Neural là tính từ của neuron (nơ-ron), network chỉ cấu trúc đồ thị nên neural network (NN) là một hệ thống tính toán lấy cảm hứng từ sự hoạt động của các nơ-ron trong hệ thần kinh.

- Mạng neural là thuật ngữ nói đến một phương pháp giải quyết vấn đề – bài toán trên máy tính mô phỏng theo hoạt động của các tế - bào thần kinh trong não bộ.

- Mạng neural nhân tạo là sự mô phỏng cấu trúc của mạng neural sinh học. Mạng neural nhân tạo được tạo thành bởi sự nối kết giữa rất nhiều đơn vị thần kinh gọi là perceptron.

- Mạng neural là một mô hình tính toán bắt chước cách mà não của con người xử lý thông tin. Nó bao gồm một số lớp của các nơ-ron được kết nối với nhau, trong đó mỗi nơ-ron tính toán các giá trị đầu vào của nó và truyền giá trị đầu ra của nó đến các nơ-ron khác trong mạng. Như vậy, một mạng neural network có thể học cách giải quyết các vấn đề bằng cách tinh chỉnh trọng số kết nối giữa các nơ-ron trong quá trình huấn luyện.

- Hoạt động của các nơ-ron**

- Nơ-ron là đơn vị cơ bản cấu tạo hệ thống thần kinh và là một phần quan trọng nhất của não. Não chúng ta gồm khoảng 10 triệu nơ-ron và mỗi nơ-ron liên kết với 10.000 nơ-ron khác.

- Ở mỗi nơ-ron có phần thân (soma) chứa nhân, các tín hiệu đầu vào qua sợi nhánh (dendrites) và các tín hiệu đầu ra qua sợi trực (axon) kết nối với các nơ-ron khác. Hiểu đơn giản mỗi nơ-ron nhận dữ liệu đầu vào qua sợi nhánh và truyền dữ liệu đầu ra qua sợi trực, đến các sợi nhánh của các nơ-ron khác.
- Mỗi nơ-ron nhận xung điện từ các nơ-ron khác qua sợi nhánh. Nếu các xung điện này đủ lớn để kích hoạt nơ-ron, thì tín hiệu này đi qua sợi trực đến các sợi nhánh của các nơ-ron khác. => Ở mỗi nơ-ron cần quyết định có kích hoạt nơ-ron đấy hay không.
- Tuy nhiên NN chỉ là lấy cảm hứng từ não bộ và cách nó hoạt động, chứ không phải bắt chước toàn bộ các chức năng của nó. Việc chính của chúng ta là dùng mô hình đấy đi giải quyết các bài toán chúng ta cần.