Thuật toán A*

- Trường hợp đặc biệt : h(u) = 0 với mọi u
  - f(u) = g(u) + h(u) = g(u)
  - Khi đó, A* là TK tốt nhất đầu tiên với hàm đánh giá g(u)
- Định lý: Nếu h(u) là chấp nhận được, thì A* là TK tối ưu

- Chứng minh:
  - Giả sử OPEN chứa:
    - Đích tối ưu cục bộ G' và
    - Trạng thái u nằm trên đường tới đích tối ưu toàn cục G
  - Ta có: g(u)+h(u) < g(u)+h*(u) vì h là chấp nhận được
  - Suy ra f(u) < g(G)
  - Mà g(G) < g(G') vì G' là tối ưu cục bộ
  - Suy ra f(u) < g(G')
  - Hay f(u) < f(G') vì h(G') = 0
  - A* không bao giờ chọn G' để phát triển

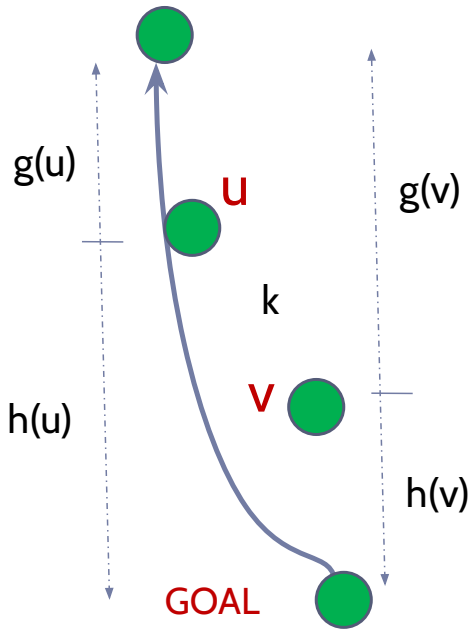
Thuật toán nhánh cận

- Branch-and-bound search
- Thuật toán nhánh cận là sự cải tiến của thuật toán tìm kiếm leo đồi
  - TK leo đồi: gặp nghiệm thì dừng tìm kiếm. Kết quả là 1 tối ưu cục bộ
  - TK nhánh cận: gặp nghiệm thì vẫn tiếp tục tìm kiếm nghiệm tốt hơn. Kết quả là nghiệm tối ưu
- Mục đích: chuyển từ tìm kiếm nghiệm tối ưu cục bộ sang tìm kiếm tốt nhất toàn cục

Thuật toán nhánh cận

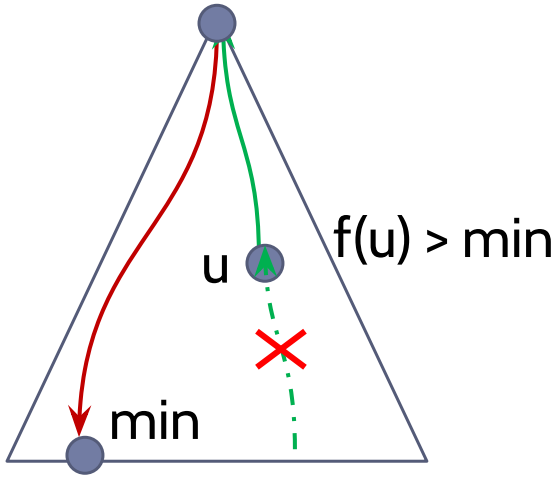
Khi sử dụng hàm đánh giá h(u) chấp nhận được, dọc theo 1 đường đi, f luôn tăng dần

Thật vậy, ta có :
Mà $h(u \rightarrow GOAL) \leq h(u \rightarrow v) + h(v \rightarrow GOAL)$
Tức $h(u) \leq k + h(v)$
$\leftrightarrow$ $g(u)+h(u) \leq g(u)+k+h(v)$
$\leftrightarrow$ $g(u)+h(u) \leq g(v)+h(v)$
$\leftrightarrow$ $f(u) \leq f(v)$



Thuật toán nhánh cận

- min : chi phí ngắn nhất tạm thời tìm thấy từ lúc bắt đầu tìm kiếm
- Khi xét nút u, nếu f(u) > min thì sẽ cắt bỏ nhánh con của u
  - Toàn bộ các nút con/cháu v của u đều có f(v) > f(u) > min nên không thể là nghiệm tốt hơn (tối ưu hơn)
- Nếu tìm thấy 1 đường đi mới tốt hơn đường đi tốt nhất tạm thời (có chi phí min), cập nhật lại min và đường đi tốt nhất tạm thời đó



Thuật toán nhánh cận

- Branch-and-bound search
- Thuật toán nhánh cận là sự cải tiến của thuật toán tìm kiếm leo đồi
  - TK leo đồi: gặp nghiệm thì dừng tìm kiếm. Kết quả là 1 tối ưu cục bộ
  - TK nhánh cận: gặp nghiệm thì vẫn tiếp tục tìm kiếm nghiệm tối ưu. Kết quả là nghiệm tối
- Mục đích: chuyển từ tìm kiếm nghiệm tối ưu cục bộ sang tìm kiếm tốt nhất toàn cục

Thuật toán nhánh cận

begin

1. Khởi tạo danh sách OPEN chỉ chứa trạng thái ban đầu;

   Gán giá trị ban đầu cho min ← +∞;

2. loop do

   2.1 if OPEN rỗng then stop;

   2.2 Loại trạng thái u ở đầu danh sách OPEN;

   2.3 if u là trạng thái kết thúc then

   if f(u) < min then {min ←f(u); Quay lại 2.1}; // cập nhật lại min

   2.4 if f(u) > min then Quay lại 2.1; //cắt bỏ nhánh con

   2.5 for mỗi trạng thái v kề u do

   { g(v) ← g(u) + k(u,v);

   f(v) ← g(v) + h(v);

   Đặt v vào danh sách L

   }

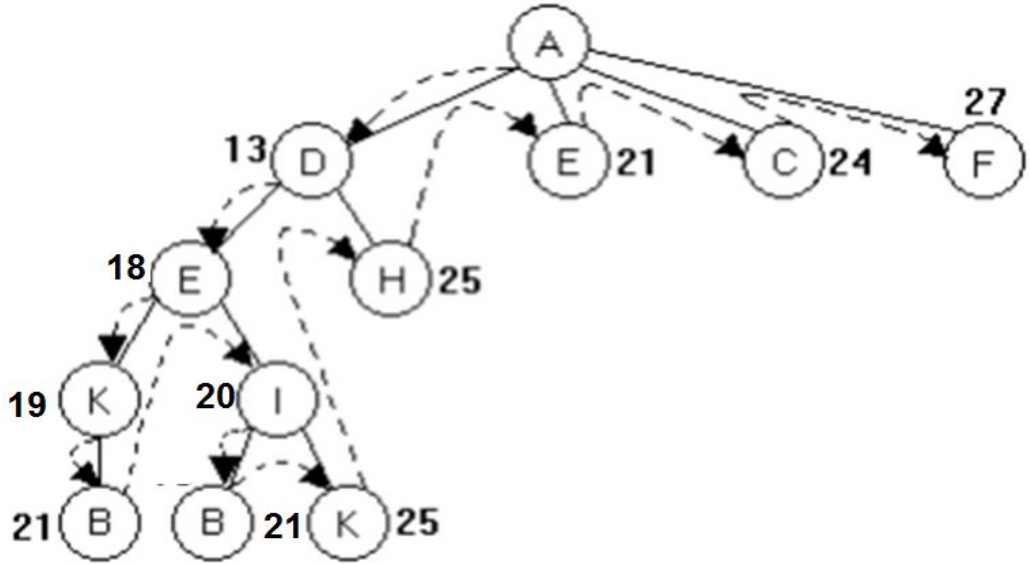
   2.6 Sắp xếp L theo thứ tự tăng của hàm f;

   2.7 Chèn L vào đầu danh sách OPEN

end;

Thuật toán nhánh cận

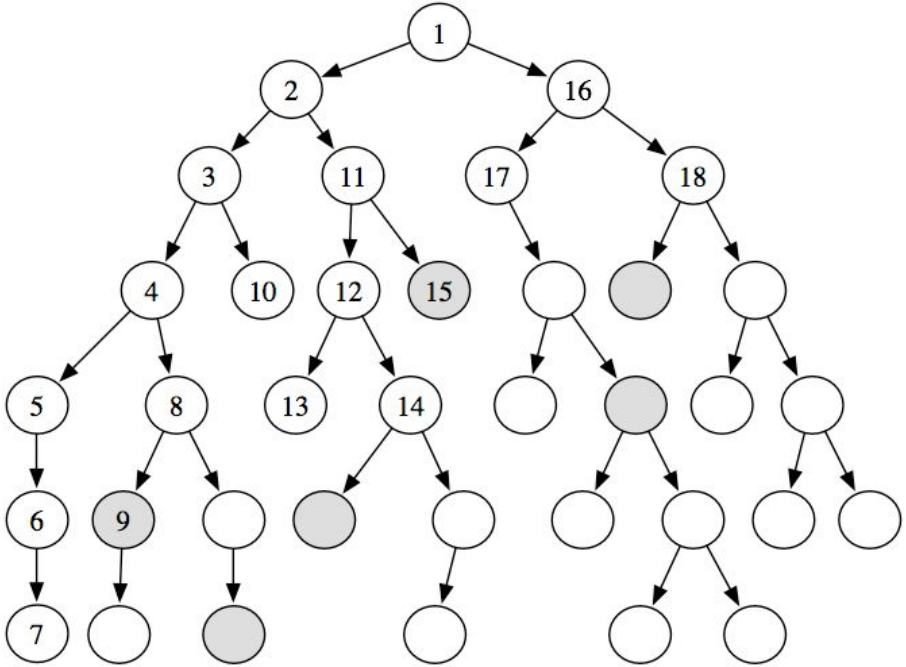
Khuyết h(u). Thông tin trên mỗi nút là chi phí g(u)



Đinh Mạnh Tường, Trí tuệ nhân tạo, Nhà xuất bản khoa học kỹ thuật, 2002

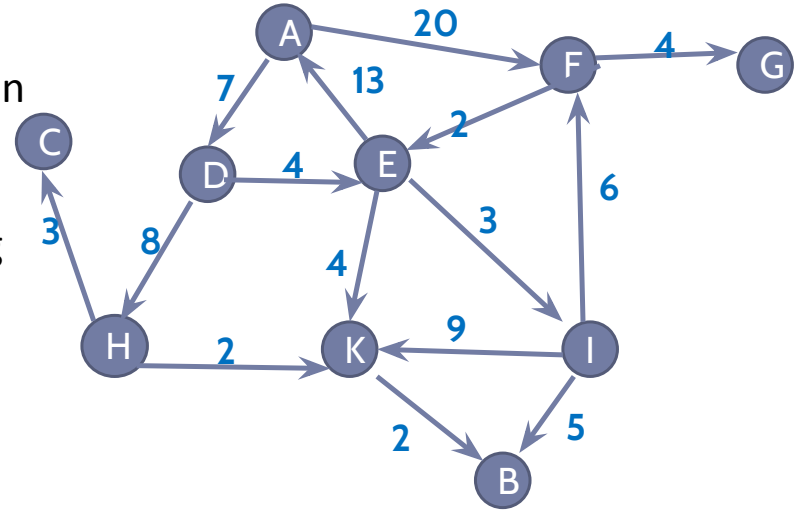
Thuật toán nhánh cận

Khuyết h(u). Thông tin ghi trên mỗi nút là chi phí tổng chi phí g(u) từ nút start đến u.



Thuật toán nhánh cận

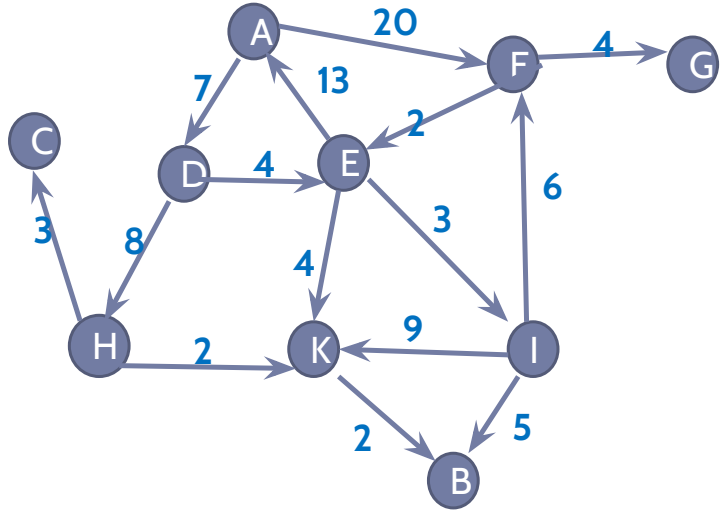
- Tìm đường đi từ A → B bằng giải thuật nhánh cận
  - Thông tin trên các cạnh là k_{uv}
  - Thuật toán nhánh cận cũng thường áp dụng với bài toán chỉ có thông tin g



Bước	u	Kề(u)	g(v)=g(u)+k(u,v)	f(v)=g(v)+h(v)	L	Open	Min
0						A	+∞
1	A	D F	g(D)=7 g(F)=20	f(D)=7 f(F)=20	D7 F20	D7 F20	+∞
2	.....						

Thuật toán nhánh cận

Bước	u	Kề(u)	g(v)=g(u)+k(u,v)	f(v)=g(v)+h(v)	L	Open	Min
0						A	+∞
1	A	D F	g(D)=7 g(F)=20	f(D)=7 f(F)=20	D7 F20	D7 F20	+∞
2	~~D~~	E H	G(E)=g(D)+k(D,E)=11 G(H)=15	F(E)=11 F(H)=15	E11 H15	~~E~~ H F	+∞
3	~~E~~	K I	G(K)=15 G(I)=11+3=14	F(K)=15 F(I)=14	I14 K15	~~I~~KHF	+∞
4	~~I~~	FBK	G(F)=14+6=20 g(B)=14+5=19 G(K)=14+9=23	F(F)=20 F(B)=19 F(K)=23	B19 F20 K23	BFKH	+∞
5	~~B~~					FKH	19
6	~~F~~	EG	G(E)=20+2=22 g(G)=20+4=24	F(E)=22 F(G)=24	E22 G24	KH	19
7	K	~~B~~	G(B)=15+2=17		B	BH	19
	B					H	17
8	H	CK	G(C)= 15+3=18 G(K)=15+2=17		K17 C18	KC	17

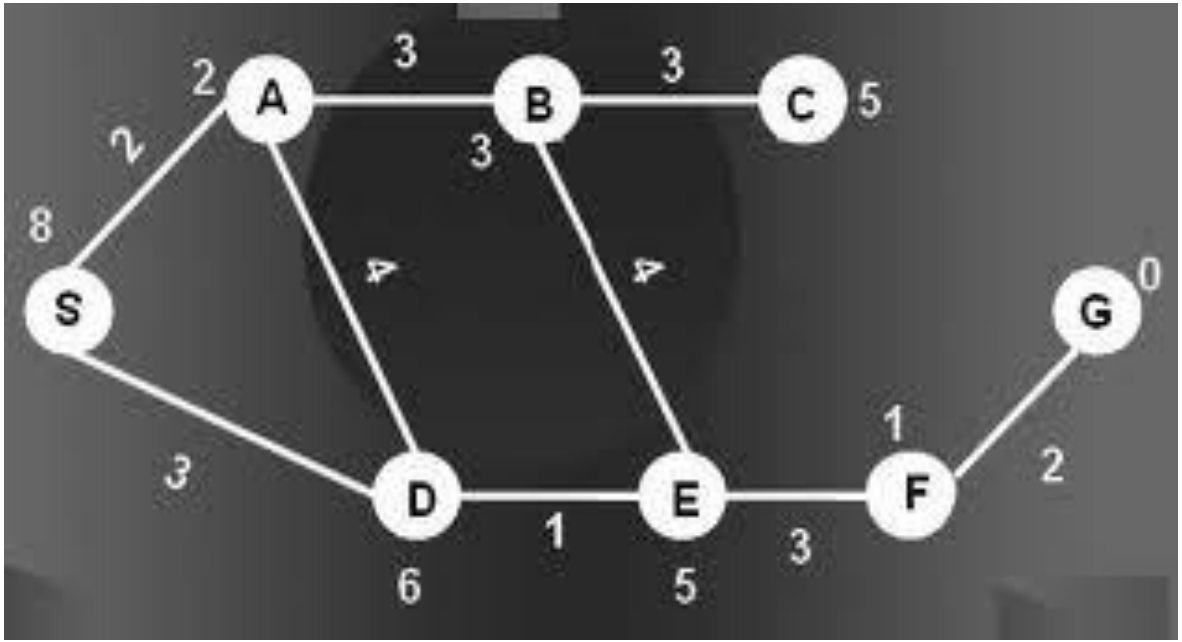


- Tìm đường đi từ A ⟶ B bằng giải thuật nhánh cận
  - Thông tin trên các cạnh là k_{uv}
  - Thuật toán nhánh cận cũng thường áp dụng với bài toán chỉ có thông tin g

B<-K<-E<-D<-A

Thuật toán nhánh cận

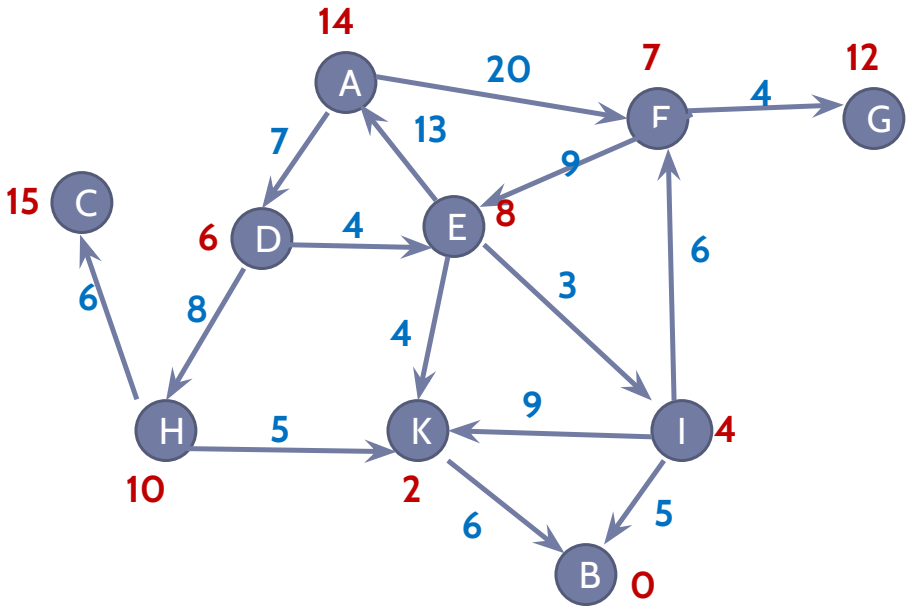
Bài tập 2: Cho sơ đồ nối các thành phố. Tìm đường đi từ S tới G bằng thuật toán nhánh cận



Thuật toán nhánh cận

Tìm đường đi từ A ⟶ B bằng giải thuật nhánh cận

Có thêm thông tin h(u) gắn tại các đỉnh)



Thuật toán nhánh cận

A*	Nhánh cận
❑Tìm thấy thì dừng ❑Phương án tìm được là tối ưu chỉ khi hàm h là hàm chấp nhận được ❑Khi hàm h không phải là chấp nhận được, phương án tìm được chỉ là phương án tìm thấy đầu tiên, chưa chắc đã là tối ưu nhất	❑Tìm thấy thì vẫn tiếp tục tìm các phương án khác để so sánh ❑Luôn tối ưu ❑Kết hợp đánh chặn (chặt bỏ) trước các nhánh không tốt ngay khi có thể (*phát hiện ra nhánh không tốt so với đường đi tốt nhất tạm thời tìm được*) ❑Cập nhật lại đường đi tốt nhất tạm thời nếu tìm thấy có đường đi khác tốt hơn