Lec 5
Tìm kiếm tối ưu – Tìm kiếm có đối thủ

Nội Dung

- **Các kỹ thuật tìm đường đi ngắn nhất**
  - Thuật toán A*
  - Thuật toán nhánh-cận
- **Các kỹ thuật tìm kiếm đối tượng tốt nhất**
  - Tìm kiếm leo đồi
  - Tìm kiếm Gradient
  - Tìm kiếm mô phỏng luyện kim
- **Tìm kiếm bắt chước sự tiến hoá**: thuật toán di truyền

Tìm đường đi ngắn nhất

Trạng thái u gọi là ***trạng thái đạt tới*** nếu có đường đi từ trạng thái ban đầu u_0 tới u .

- Hàm đánh giá:
  - Độ dài đường đi ngắn nhất từ u_0 tới u: g(u)
    - Nếu u không phải trạng thái đích thì đường đi từ u_0 tới u gọi là đường đi một phần
    - Nếu u là trạng thái đích thì đường đi từ u_0 tới u gọi là đường đi đầy đủ
  - Độ dài đường đi ngắn nhất từ u tới trạng thái đích: h(u)

hàm đánh giá: f(u) = g(u) + h(u)

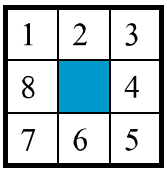
Cài Đặt Hàm Đánh Giá (Evaluation Function)

Xét trò chơi 8-puzzle. Cho mỗi trạng thái n một giá trị f(n):

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

g(n) = khoảng cách thực sự từ n đến trạng thái bắt đầu

h(n) = hàm heuristic đánh giá khoảng cách từ trạng thái n đến mục tiêu.

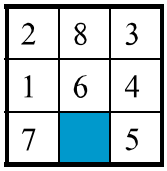


goal

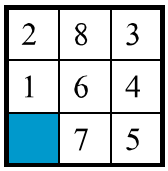
h(n): số lượng các vị trí còn sai

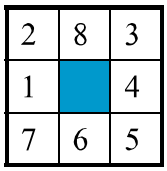
g(n) = 0

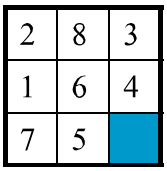
start



g(n) = 1







f(n) = **6** **4** **6**

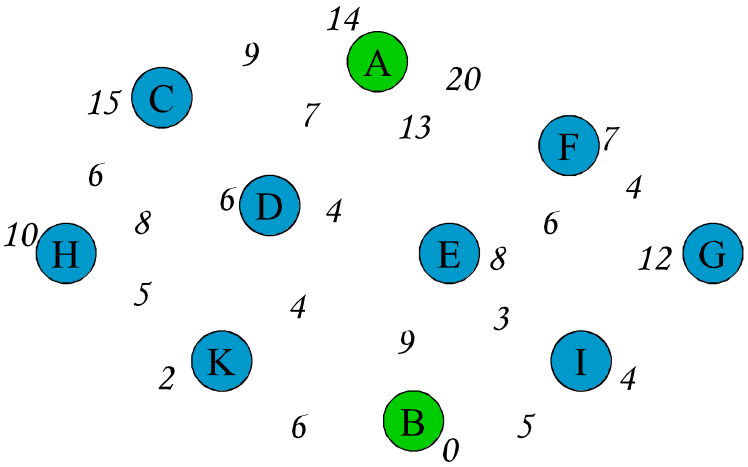
Thuật toán A*

- Tìm kiếm tốt nhất đầu tiên + hàm đánh giá f(u)

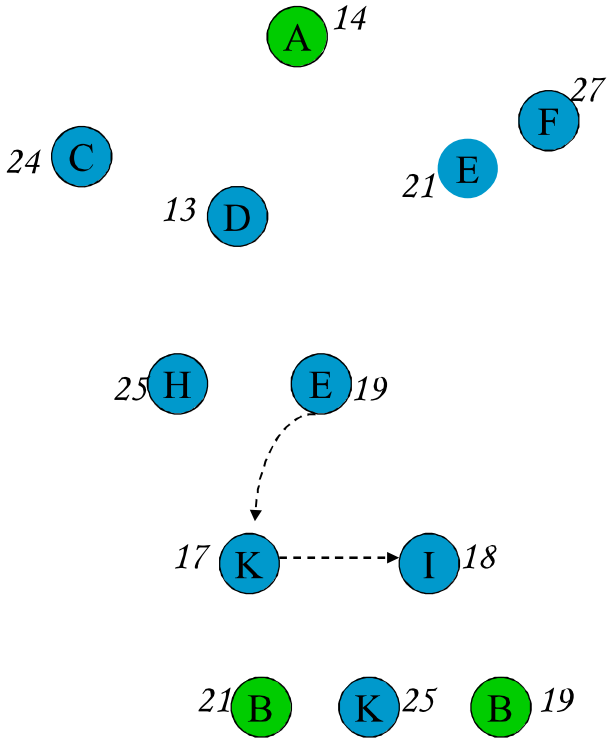
Procedure A*;
Begin
1. Khởi tạo danh sách L chỉ chứa trạng thái đầu;
2. **Loop do**
    2.1 **If** L rỗng **then** {thông báo thất bại; stop};
    2.2 Loại trạng thái u ở đầu danh sách L;
    2.3 **If** u là trạng thái kết thúc **then**
        {thông báo thành công; stop};
    2.4 **For** mỗi trạng thái v kề u **do**
        {g(v)←g(u)+k(u,v)
        f(v)←g(v)+h(v);
        đặt v vào danh sách L;}
    2.5 Sắp xếp L theo thứ tự tăng dần của hàm f;

End;

Ví dụ: thuật toán A*



Đồ thị không gian trạng thái với hàm đánh giá



Cây tìm kiếm theo thuật toán A*

Nhận xét về thuật toán A*

- Nếu h(u) là đánh giá thấp (đặc biệt h(u)=0 với mọi trạng thái u), thì A* là thuật toán tối ưu, tức là nghiệm tìm được là tối ưu.
- Nếu độ dài các cung không nhỏ hơn một số dương δ nào đó thì A* là thuật toán đầy đủ, tức là nó luôn dừng và tìm ra nghiệm.

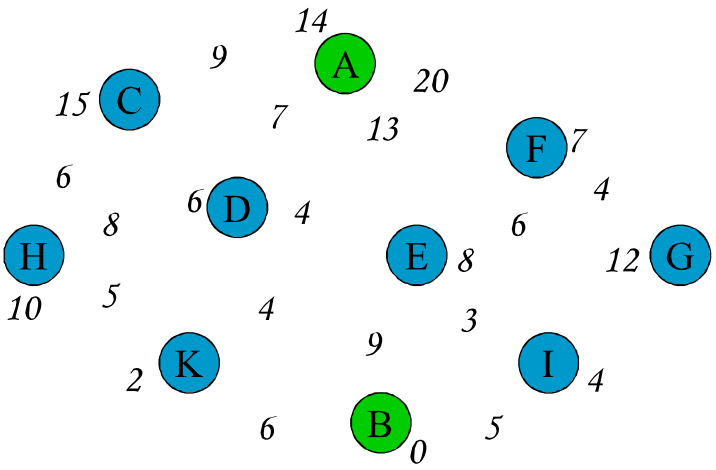
Thuật toán tìm kiếm nhánh-cận

- Tìm kiếm leo đồi + hàm đánh giá f(u)

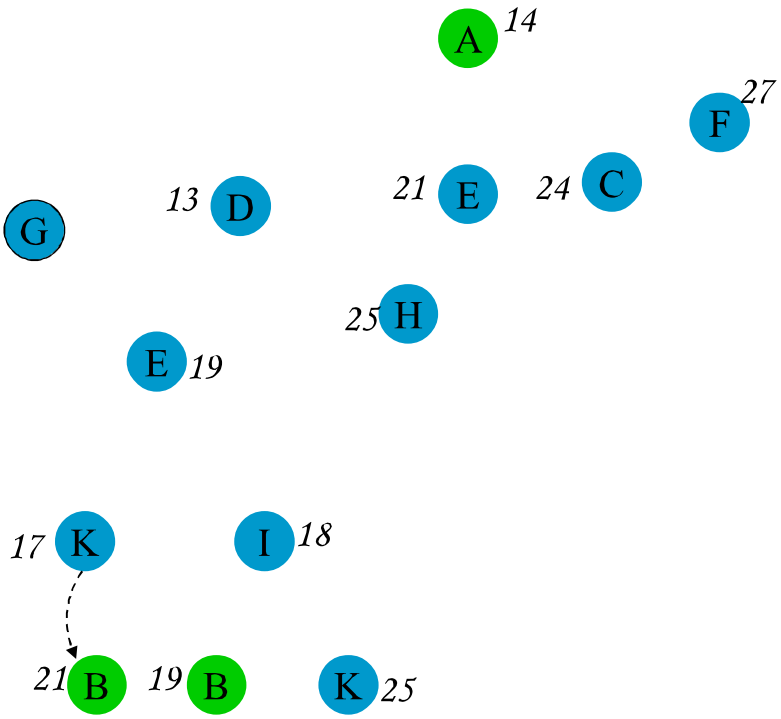
Procedure Branch-and-Bound;
Begin
1. Khởi tạo danh sách L chỉ chứa trạng thái đầu;
   Gán giá trị ban đầu cho cost;
2. **Loop do**
   2.1 **If** L rỗng **then** {thông báo thất bại; stop};
   2.2 Loại trạng thái u ở đầu danh sách L;
   2.3 **If** u là trạng thái kết thúc **then**
   if g(u)<=cost then {cost ←g(u); quay lại 2.1};
   2.4 **if** f(u)>cost **then** quay lại 2.1;
   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_1};
   2.6 Sắp xếp L_1 theo thứ tự tăng dần của hàm f;
   2.7 Chuyển danh sách L_1vào đầu danh sách L sao cho L_1 ở đầu danh sách L;

End;

Ví dụ: thuật toán nhánh-cận



Đồ thị không gian trạng thái
với hàm đánh giá



Cây tìm kiếm nhánh-cận

Nhận xét

- Thuật toán nhánh-cận cũng là thuật toán đầy đủ và tối ưu nếu h(u) là hàm đánh giá thấp và độ dài các cung không nhỏ hơn một số dương δ nào đó.

Tìm đối tượng tốt nhất

Trên không gian tìm kiếm U, mỗi đối tượng x được xác định với một hàm giá cost(x)→ cần tìm đối tượng mà hàm giá đạt giá trị lớn nhất, gọi là ***đối tượng tốt nhất***.

Tìm đối tượng tốt nhất

Tìm kiếm leo đồi

Tương tự kỹ thuật tìm kiếm leo đồi để tìm trạng thái kết thúc đã xét, tuy nhiên trong thuật toán này, từ một đỉnh u ta chỉ leo lên đỉnh tốt nhất v (được xác định bởi hàm giá cost) trong lân cận u nếu đỉnh này *cao hơn* u, tức là cost(v)>cost(u).

Thuật toán dừng ngay khi không leo lên đỉnh cao hơn được nữa.

Thuật toán di truyền

- TTDT bắt chước sự *chọn lọc tự nhiên* và *di truyền*:
  - Chọn lọc tự nhiên: các cá thể khoẻ, có khả năng thích nghi tốt với môi trường sẽ được tái sinh và nhân bản ở các thế hệ sau
  - Di truyền: Trong quá trình sinh sản, các cá thể con thừa hưởng các phẩm chất của cha mẹ và có những đột biến.
- Mỗi cá thể được mã hoá bởi một cấu trúc DL mô tả cấu trúc gien của cá thể đó, gọi là *nhiễm sắc thể*.
- Một *thế hệ* là một quần thể ứng với một giai đoạn phát triển.
- TTDT bắt chước chọn lọc tự nhiên và di truyền để biến đổi các thế hệ.

Thuật toán di truyền

Các toán tử biến đổi các thế hệ

- **Toán tử tái sinh**: các cá thể tốt được lựa chọn để đưa vào thế hệ sau, sự chọn lọc dựa vào độ thích nghi với môi trường.
- **Toán tử lai ghép**: hai cá thể cha mẹ trao đổi gen để tạo ra hai cá thể con.
- **Toán tử đột biến**: Một cá thể thay đổi một số gen để tạo thành cá thể mới.

Thuật toán di truyền

Procedure Genetic-Algorithm;
Begin
    t ← 0;
    Khởi tạo thế hệ ban đầu P(t)
    Đánh giá P(t) (dựa vào hàm thích nghi);
    repeat
        t ← t + 1;
        Sinh ra thế hệ mới P(t) từ P(t-1) bởi:
            Chọn lọc
            Lai ghép
            Đột biến;
        Đánh giá P(t);
    until điều kiện kết thúc được thoả mãn;
End;

Tìm kiếm có đối thủ

- Bài toán tìm kiếm có đối thủ (chơi cờ) được biểu diễn trong không gian trạng thái:
  - *Trạng thái ban đầu*: sự sắp xếp các quân cờ của hai bên lúc bắt đầu chơi.
  - *Các toán tử:* các nước đi hợp lệ
  - *Các trạng thái kết thúc*: các tình thế mà cuộc chơi dừng.
  - *Hàm kết cuộc*: ứng mỗi trạng thái kết thúc với một giá trị nào đó.

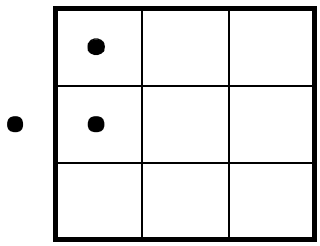
Tìm kiếm có đối thủ

Cây trò chơi

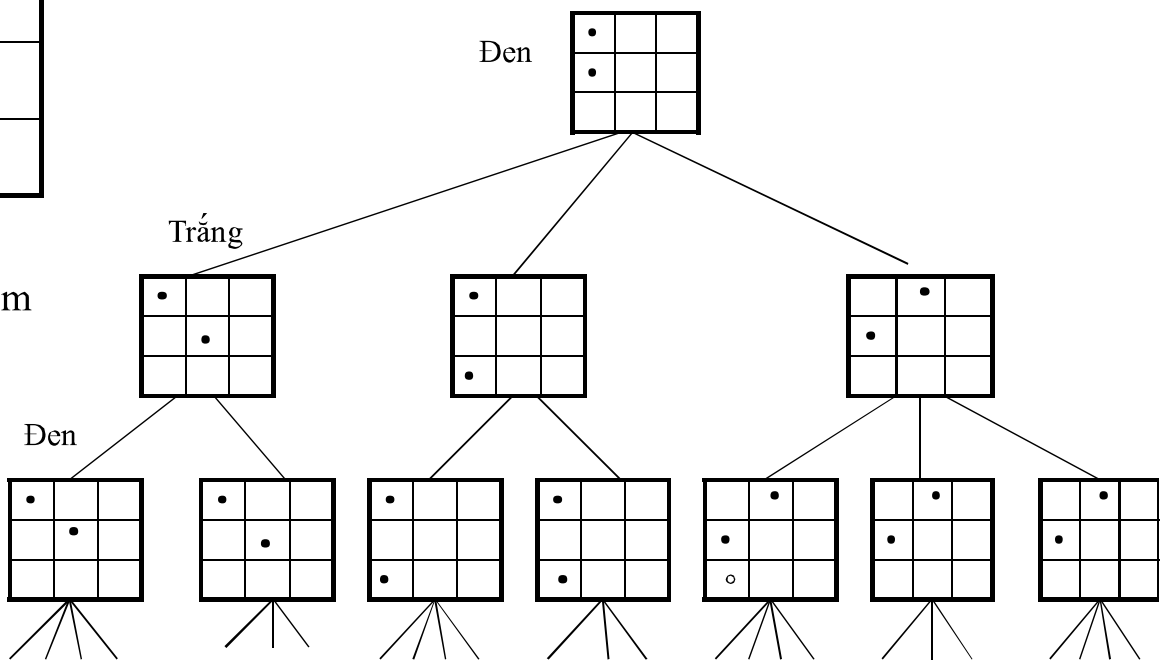
– Gốc ứng với trạng thái đầu
– Đỉnh ứng với trạng thái mà Trắng (Đen) đưa ra nước đi gọi là đỉnh Trắng (Đen)
– Các đỉnh con của đỉnh Trắng (Đen) biểu diễn trạng thái u là tất cả các đỉnh biểu diễn trạng thái v, v nhận được từ u do Trắng (Đen) thực hiện nước đi hợp lệ nào đó.
– Lá của cây ứng với trạng thái kết thúc.

Tìm kiếm có đối thủ

Ví dụ: Cây trò chơi



Trò chơi Dodgem



Cây trò chơi Dodgem với Đen đi trước

Heuristic trong trò chơi có đối thủ

Chiến lược min-max

– Hai đấu thủ trong trò chơi được gọi là **MIN** và **MAX**.

– Mỗi nút lá có giá trị:

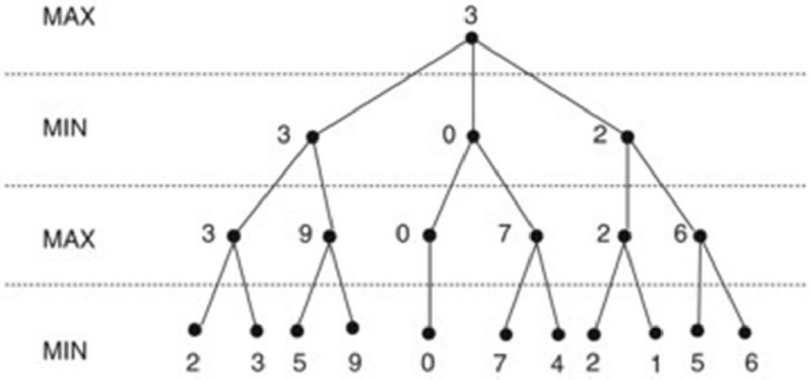
- **1** nếu là MAX thắng,
- **0** nếu là MIN thắng.

– Minimax sẽ truyền các giá trị này lên cao dần trên đồ thị, qua các nút cha mẹ kế tiếp theo các luật sau:

- Nếu trạng thái cha mẹ là **MAX**, gán cho nó giá trị **lớn nhất** có trong các trạng thái con.
- Nếu trạng thái cha mẹ là **MIN**, gán cho nó giá trị **nhỏ nhất** có trong các trạng thái con.

Minimax với độ sâu lớp cố định

- Minimax đối với một KGTT giả định.



- Các nút lá được gán các giá trị ***heuristic***
- Còn giá trị tại các nút trong là các giá trị nhận được dựa trên giải thuật Minimax

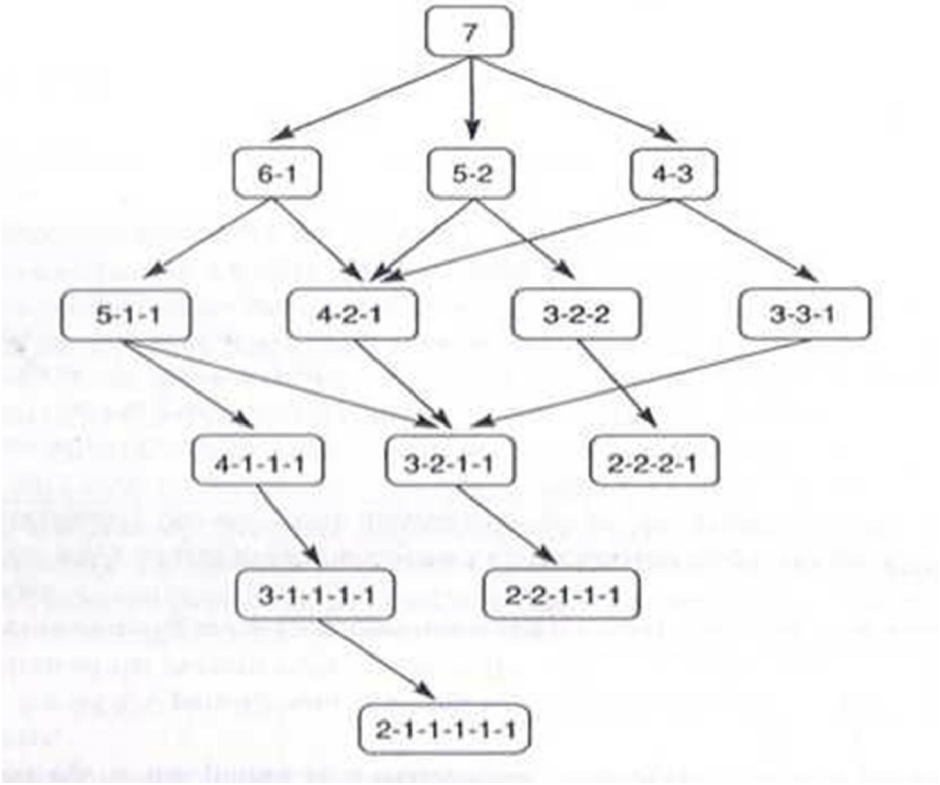
Giải thuật minimax

```
Function MaxVal(u);
begin
        if  u là đỉnh kết thúc  then  MinVal(u) ← f(u)
        else MinVal(u) ← min{MaxVal(v) | v là đỉnh con của u}
end;

Function MinVal(u);
begin
        if  u là đỉnh kết thúc  then  MaxVal(u) ← f(u)
        else MaxVal(u) ← max{MinVal(v) | v là đỉnh con của u}
end;

Procedure Minimax(u, v);
begin
        val ←-∝;
        for  mỗi w là đỉnh con của u  do
        if  val(u) <= MinVal(w)  then
            {val ← MinVal(w);  v ←  w}
end;
```

Áp dụng GT Minimax vào trò chơi NIM



Heuristic trong trò chơi tic-tac-toe

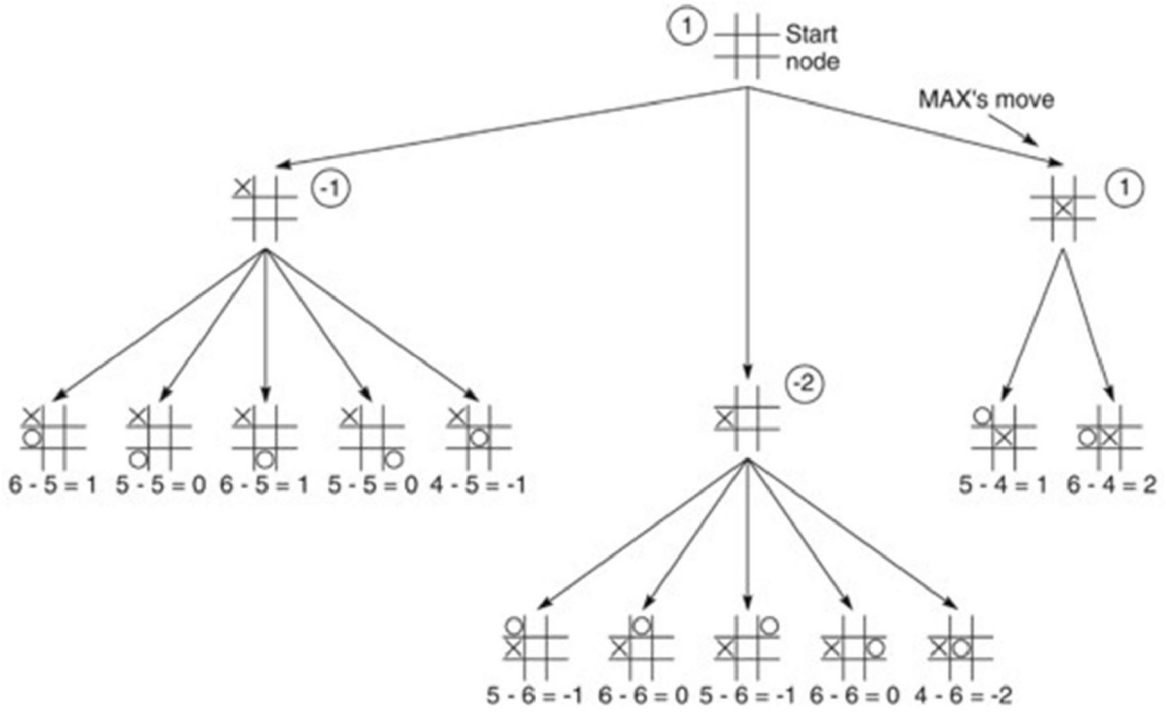
Hàm Heuristic: **E(n) = M(n) – O(n)**

Trong đó: M(n) là *tổng số đường thắng có thể của tôi*

O(n) là *tổng số đường thắng có thể của đối thủ*

E(n) là *trị số đánh giá tổng cộng cho trạng thái n*

Minimax 2 lớp được áp dụng vào nước đi mở đầu trong tic-tac-toe

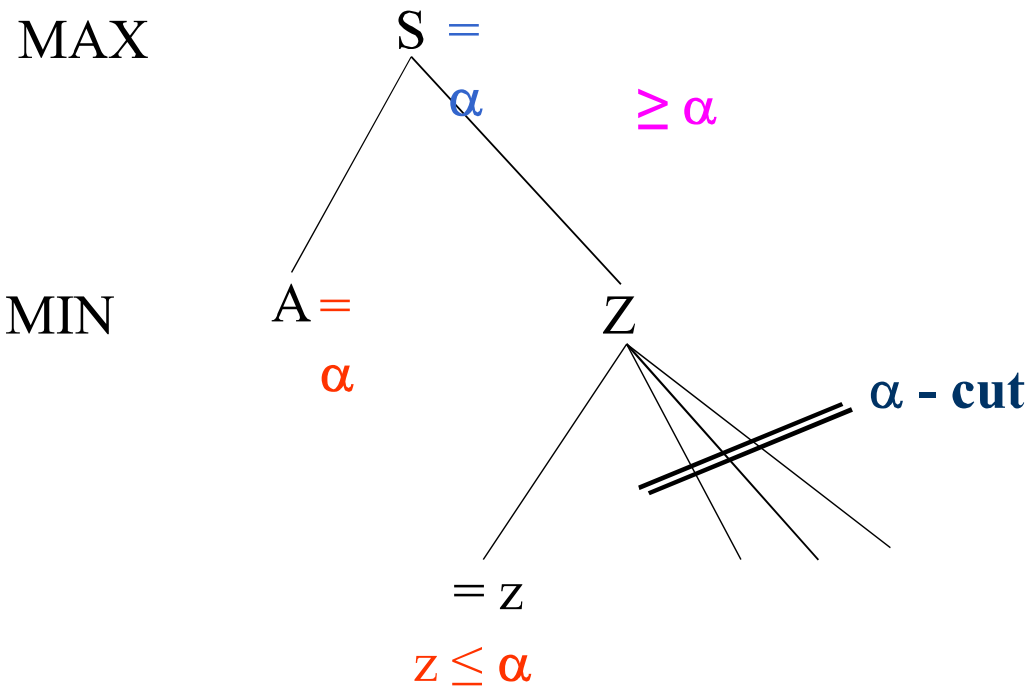


Trích từ Nilsson (1971).

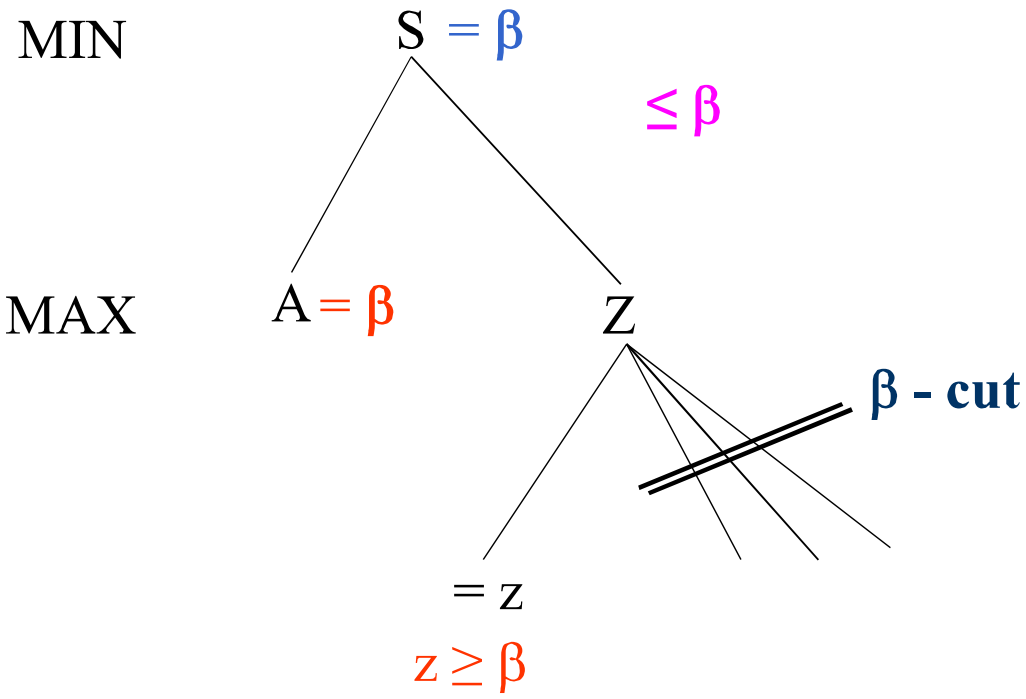
Giải thuật cắt tỉa α-β

- Tìm kiếm theo kiểu depth-first.
- Nút MAX có 1 giá trị α (luôn tăng)
- Nút MIN có 1 giá trị β (luôn giảm)
- **TK có thể kết thúc dưới bất kỳ**:
  - Nút MIN nào có $\beta \leq \alpha$ của bất kỳ nút cha MAX nào.
  - Nút MAX nào có $\alpha \geq \beta$ của bất kỳ nút cha MIN nào.
- Giải thuật cắt tỉa α-β thể hiện *mối quan hệ giữa các nút ở lớp n và n+2*, mà tại đó toàn bộ cây có gốc tại lớp n+1 có thể cắt bỏ.

Cắt tỉa α



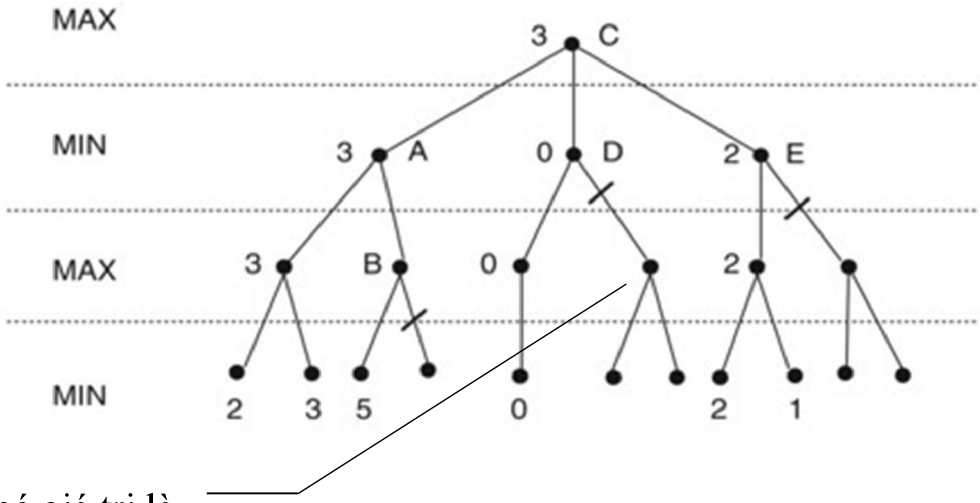
Cắt tỉa β



- **Function** MaxVal(u,α, β);
  begin
  if u là lá của cây hạn chế hoặc đỉnh kết thúc **then** MaxVal←eval(u)
  else **for** mỗi đỉnh v là con của u **do**
  {α ← max[α,MinVal(v,α,β)];
  if α ≥ β **then** exit};
  MaxVal ←α;
  end;
- **Function** MinVal(u,α, β);
  begin
  if u là lá của cây hạn chế hoặc đỉnh kết thúc **then** MinVal←eval(u)
  else **for** mỗi đỉnh v là con của u **do**
  {β ←min[β,MaxVal(v,α,β)];
  if α ≥ β **then** exit};
  MinVal ← β;
  end;

Procedure Alpha_beta(u,v);
begin
α←-∝;
β←∝;
for mỗi đỉnh w là con của u **do**
if α ≤ MinVal(w, α, β) **then**
{α ← MinVal(w, α, β);
v ←w; }
end;

GT cắt tỉa α-β áp dụng cho KGTT giả định



Các nút không có giá trị là các nút không được duyệt qua

A has $\beta = 3$ (A will be no larger than 3)
B is β pruned, since 5 > 3
C has $\alpha = 3$ (C will be no smaller than 3)
D is α pruned, since 0 < 3
E is α pruned, since 2 < 3
C is 3

Bài Tập Chương 4