

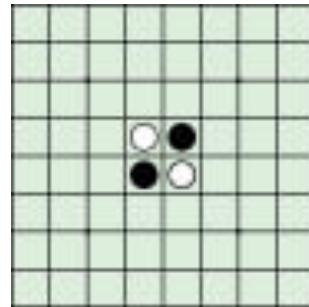
ပိုဒ္ဓါ ၂

ကဏ္ဏရေးပုံမှန်စာတို့သိမ်းကြပ်ယူပို့ဆောင်ရွက်

- ▶ ຄົນເຮົາຕ້ອງການທີ່ຈະຊະນະ ບໍ່ມີໃຜທີ່ຈະຍອມແຜ້
- ▶ ຫຼິ້ນເກມແຜ່ອຢາກຊະນະ
- ▶ ເມື່ອມີຄອມຜິວເຕີຈຶ່ງຢາກສ້າງຄອມຜິວເຕີໃຫ້ສາມາດຫຼິ້ນເກມໄດ້
- ▶ Charles Babbage ໄດ້ສ້າງເຄື່ອງຫຼິ້ນເກມ Tic-Tac-Toe ໃນປີ 1953
- ▶ ຕໍ່ມາ Alan Turing ໄດ້ອະທິບາຍໂປຣແກຣມການຫຼິ້ນໝາກເສີກ (ບໍ່ໄດ້ສ້າງໂປຣແກຣມຈຶ່ງ)

ເກມຖືກນຳມາທິດລອງຫາງ AI ແພະ

- ▶ ເກມເປັນໂຄງສ້າງທີ່ດີສໍາລັບການທິດສອບ ຄວາມສໍາເລັດ ຫຼື ຄວາມປໍ່ສໍາລັດ
- ▶ ເກມຕ້ອງການຖານຄວາມຮູ້ທີ່ບໍ່ຫຼາຍເກີນໄປ ແລະ ສາມາດເຂົ້າໃຈໄດ້ງ່າຍ



Two-player game

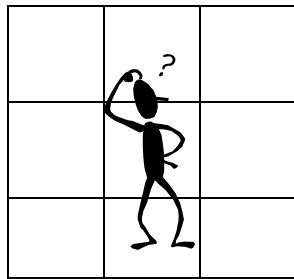
- ▶ มีຜູ້ຫົ່ນ 2 ຄົນ ໂດຍປ່ຽນກັນຫົ່ນເທື່ອລະຄົມ
- ▶ ເຊັ່ນ ເງມ Tic-Tac-Toe ຫຼື “OX” ຫາກດ້າມ ຫາກເສີກຜູ້ຫົ່ນ ແຕ່ລະຝ່າຍຕ້ອງໄປເຜື່ອໃຫ້ຕົນເອງມີໂອກາດຊະນະຝ່າຍກົງກັນຂໍ້າມ
- ▶ ມີການເບິ່ງເຫັນໂອກາດລວງໜ້າ
- ▶ ການໄປແຕ່ລະຄົງຕ້ອງຢູ່ໃນກົດລະບຽບຂອງເກມນັ້ນ



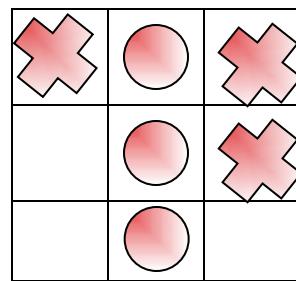


- ▶ เกมเป็นFormal task จึงแก้บันทາด้วยการคั่นหาได้ เมื่อ
 - ▶ 1. มีจุดเริ่มต้น (Initial state)
 - ▶ 2. มีการปั่นธุบแบบป่างແມ່ນອນດ้วยກົດລະບຽບຂອງການໜຶ່ນ (Set of operators)
 - ▶ 3. ຈຸດສິນສຸດທີ່ຊະນະ ຫຼື ສະເໜີ (Terminal state)
 - ▶ 4. มີ utility function ທີ່ສາມາດປະເມີນຄ່າທີ່ ຈຸດສິນສຸດວ່າ ຊະນະ ຫຼື ສະເໜີ

Two-Player Game ຕົວຢ່າງ Tic-Tac-Toe

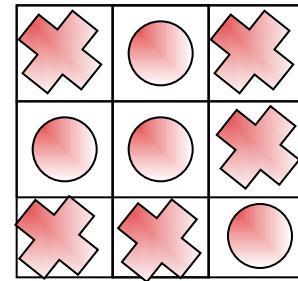


ຈຸດເລີ່ມຕົ້ນ



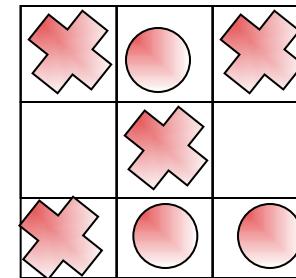
-1

Lose



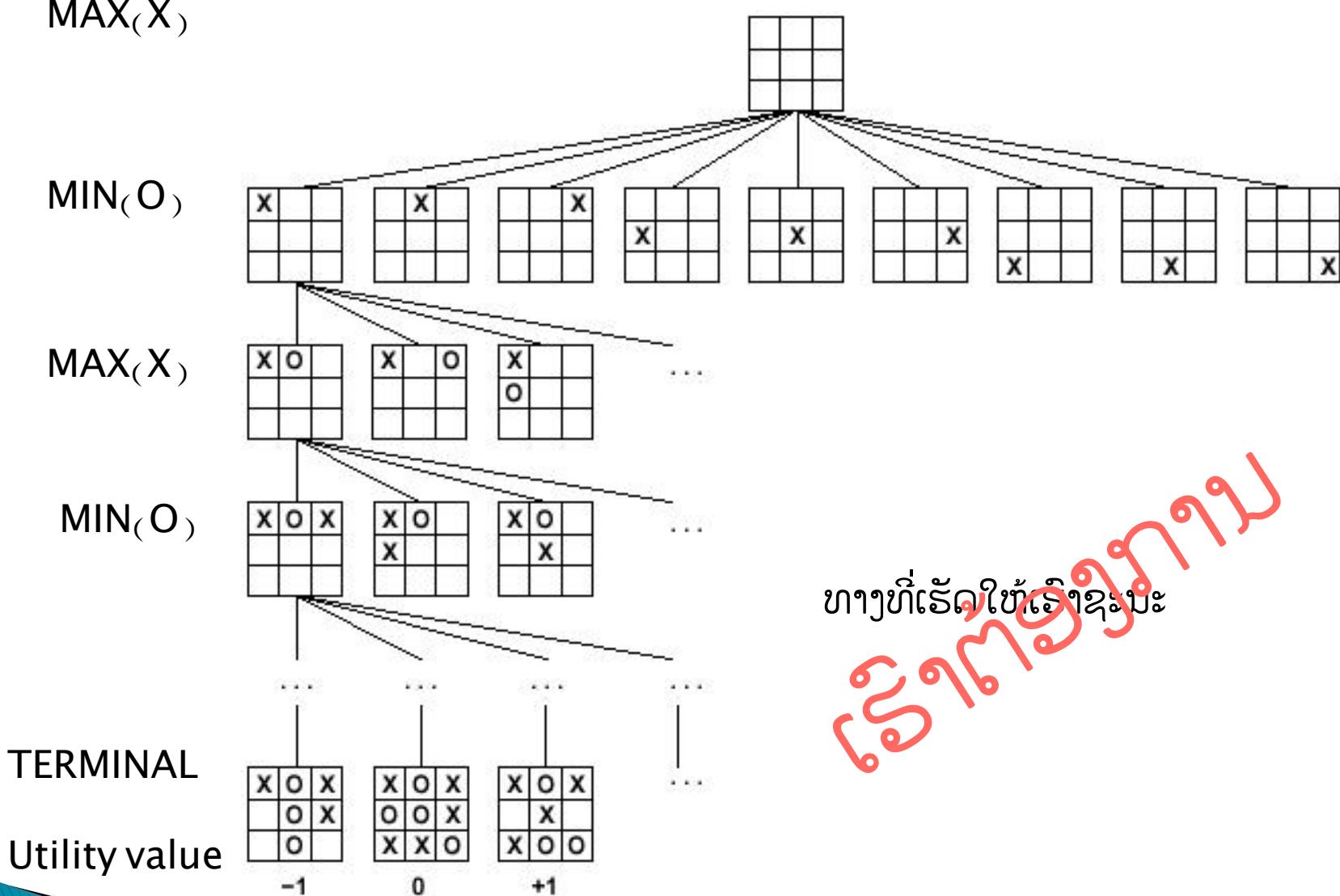
0

Even



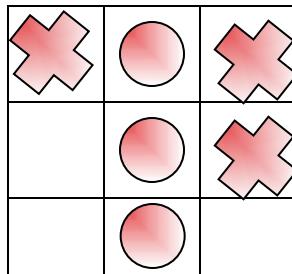
1





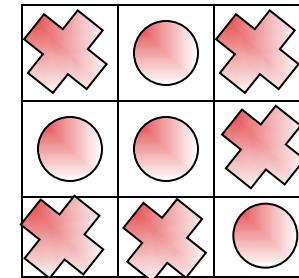
Utility function

- ໃຊ້ປະເມີນຄ່າໂອກາດຊະນະຂອງ ໂນດ (ໂນດສິ້ນສຸດເກມ)



-1

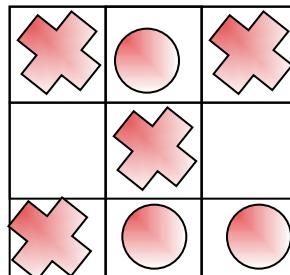
-1 ຮອດ -10 = ຝ່າຍກົງກັນຂໍາມຈະຊະນະ ຫຼາຍ
ຫຼື ນ້ອຍ ຂຶ້ນຢູ່ກັບຄ່ານີ້ ຄ່ານ້ອຍໆສະແດງວ່າມີ
ໂອກາດຊະນະຫຼາຍ



0

0 = ຜູ້ຫຼັ້ນກັບເຝ່າຍກົງຂໍາມສະໜີກັນ

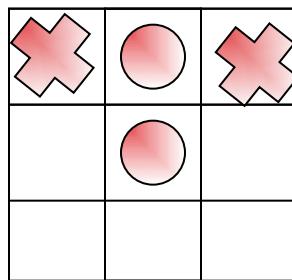
1



1 - 10 = ຜູ້ຫຼັ້ນຈະຊະນະ ຫຼາຍ ຫຼື ນ້ອຍ ຂຶ້ນຢູ່ກັບ
ຄ່ານີ້ ຄ່າຫຼາຍໆສະແດງວ່າມີໂອກາດຊະນະຫຼາຍ

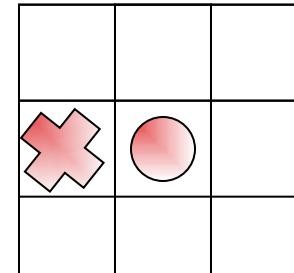
Evaluation function

- ใช้ประเมินถ้าโอกาสชนะของ โนด (ที่บ่อมีผลสัมฤทธิ์) โดยการคาดคะเนว่า
- เข้ม ให้ 1 สำลับทุกๆ ป่อนที่มีแล้วจะชนะ



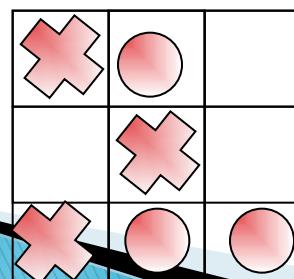
-1

-1 รอง -10 = ฝ่ายกิงข้ามมีโอกาสชนะ



0

0 = มีโอกาสชนะ



2

1 - 10 = ผู้ทึนมีโอกาสชนะ

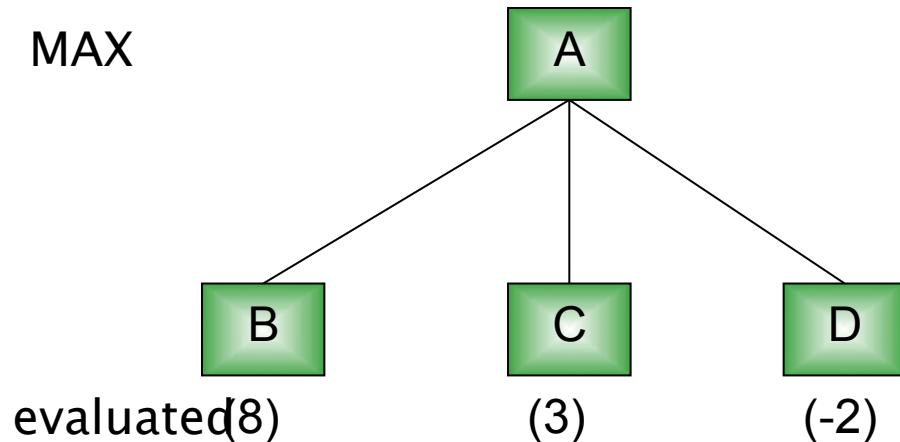
$EV() =$ บวก 1 คะแนน ให้บ่อนที่มีแล้วจะชนะ

ໂອກາດຊະນະ

- ຄວາມຖືກຕ້ອງ Utility function ຫຼື Evaluation function
 - ຂຶ້ນຢູ່ກັບປະສົບການຂອງເກມນັ້ນງ
- ການເບິ່ງເກມລ່ວງໜ້າ
 - ເບິ່ງລ່ວງໜ້າ 1 ຕາ (One-ply search)
 - ເບິ່ງລ່ວງໜ້າ 2 ຕາ (Two-ply search)
 - ເບິ່ງລ່ວງໜ້າ n ຕາ (n -ply search)
 - ເບິ່ງລ່ວງໜ້າຈົນຮອດຈຸດສິນສຸດ

ຕົວຢາງການຄົ້ນຫາແບບ One-Ply Search

MAX



ຈຸດເຕັມ

- ຜູ້ຫຼິ້ນຈະເລືອກທາງໄປທີ່ໃຫ້ຄ່າດີທີ່ສຸດ
- ໃຊ້ Evaluation function
- ວ່ອງໄວ

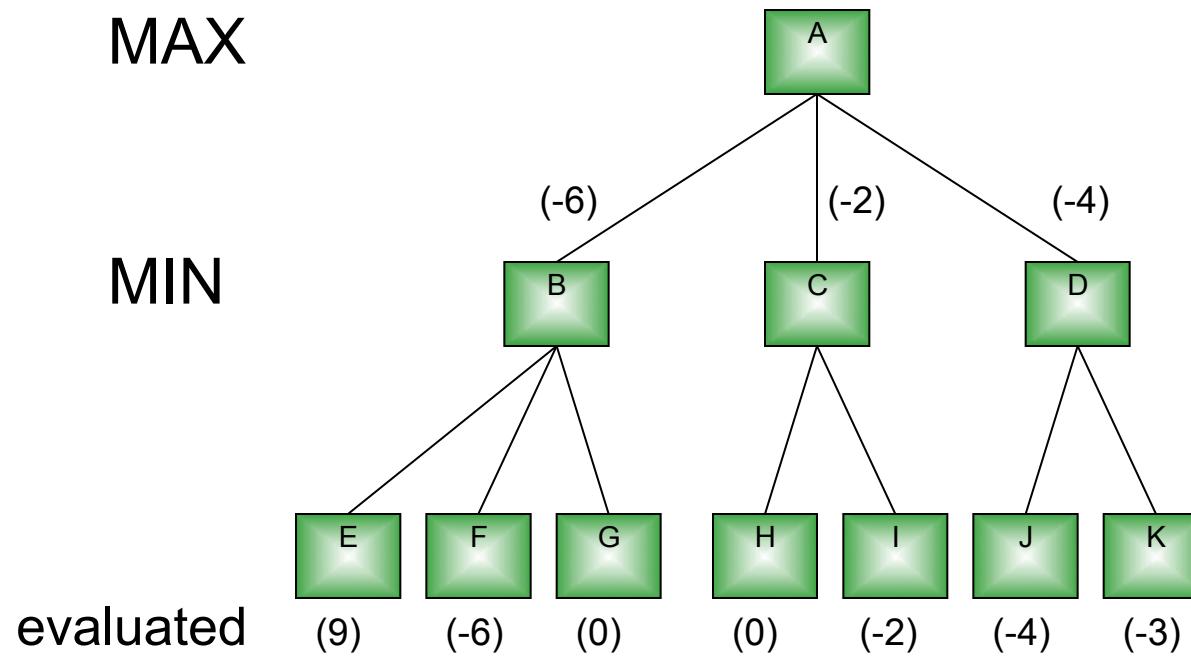
ຈຸດດ້ອຍ

- ເປົ່າລ່ວງໜ້າພຽງຄົງດຽວອາດບໍ່ຖືກຕ້ອງ
ທັງໝົດ
- ເມື່ອໄປທີ່ B ແລ້ວຝ່າຍກົງກັນຂໍ້າມອາດ
ເລືອກທາງໄປແບບອື່ນແລ້ວເຮັດໃຫ້ຜູ້ຫຼິ້ນແຍ
ປຽບໄດ້

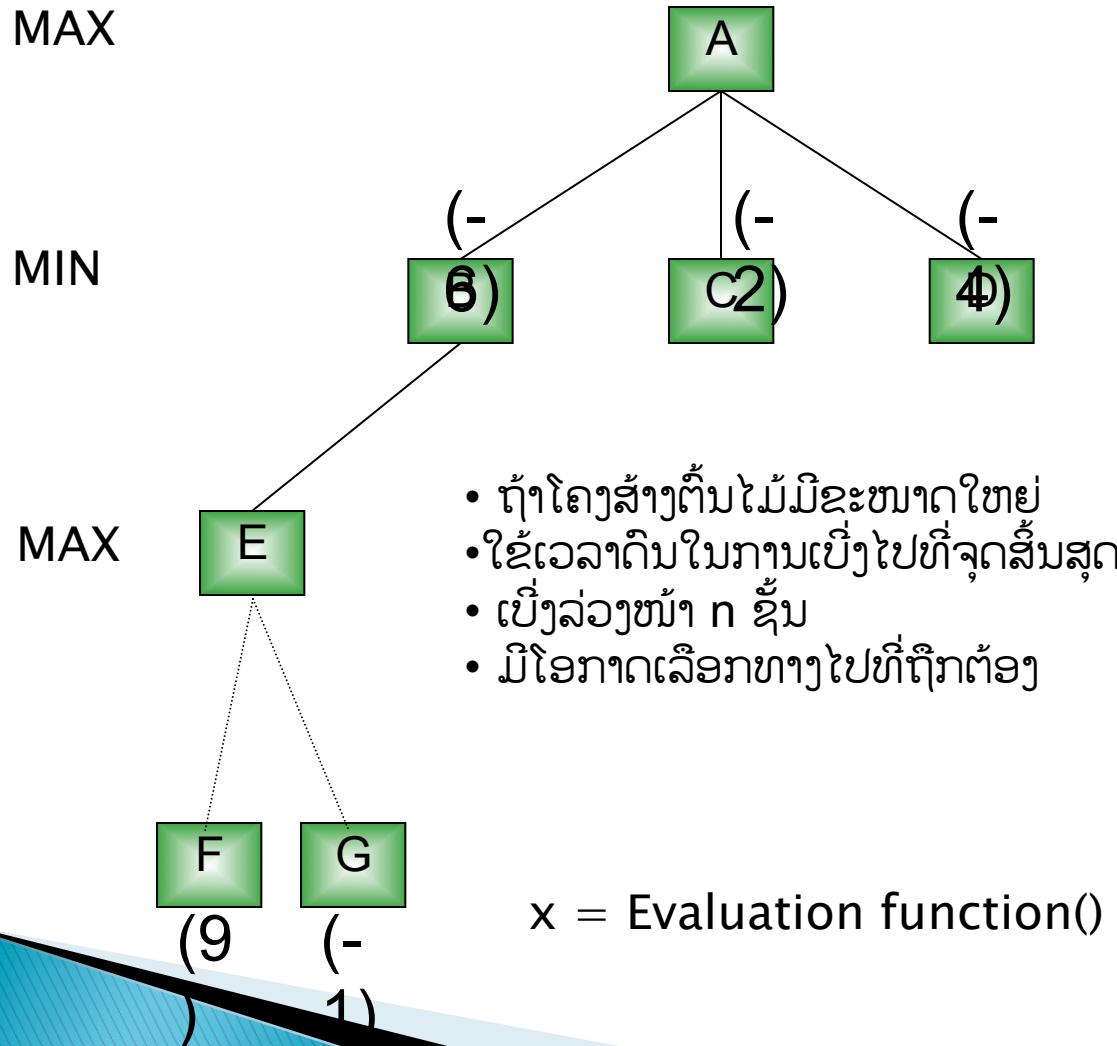
ການຄົ້ນຫາແບບ Two-Ply Search

- MAX ເລືອກທາງໄປທີ່ມີຄ່າຫຼາຍທີ່ສຸດ

- MIN ເລືອກທາງໄປທີ່ມີຄ່ານ້ອຍທີ່ສຸດ



ເບີງລ່ວງໜ້າ n ຕາ (n-ply search)



- ຖ້າໂຄງສ້າງຕົ້ນໄມ້ມີຂະໜາດໃຫຍ່
- ໃຊ້ເວລາດິນໃນການເບີງໄປທີ່ຈຸດສິ້ນສຸດ
- ເບີງລ່ວງໜ້າ n ຊັ້ນ
- ມີໂອກາດເລືອກຫາງໄປທີ່ຖືກຕ້ອງ

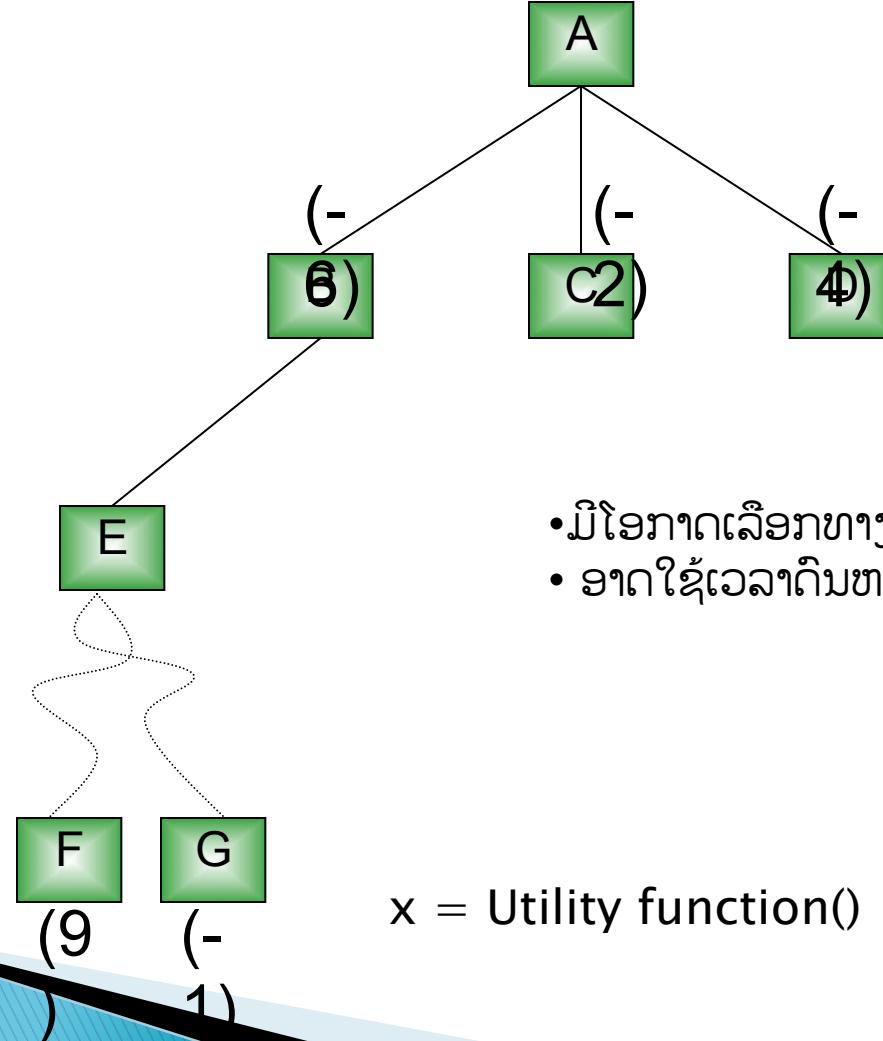
ຄົນຫາແບບລ່ວງໜ້າຈິນຮອດຈຸດສິນສຸດ

MAX

MIN

MAX

utility



- ມີໂອກາດເລືອກຫາງໄປທີ່ຖືກຕ້ອງຫຼາຍທີ່ສຸດ
- ອາດໃຊ້ເວລາດົນຫາກຕົ້ນໄມ້ມີຂະໜາດໃຫຍ່

MINIMAX ALGORITHM

- มีวิธีการเดาที่ใช้ Depth-first search ถึง ภาระสั้นโนดลึกไปทางเล็กกว่า จนกว่าจะถูกต้องตาม (อาจเป็นจุดสิ้นสุดของเกม หรือ จุดลึก N ใดหนึ่ง)
- ใช้ Utility Function หรือ Evaluation function เพื่อหาค่าความดีของโนดทุกตัว
- สูงถ้าที่ได้รับขึ้นไปในระดับที่ผู้เล่นเป็นฝ่ายชนะ ต่ำถ้าเป็นฝ่ายแพ้
- ใช้เทคนิคการเรียกซ้ำ (Recursive) ได้ ถึง Algorithm หน้าต่อไป

PSUEDOCODE OF MINIMAX ALGORITHM

1. ส້າງໂຄງຮ່າງຕົນໄມ້ ທັງໝົດລົງໄປຫາ terminal node
2. ໃຊ້ utility function ເພື່ອຫາຄ່າຂອງໂນດນີ້ນ
3. ສົ່ງຄ່າຜົນຮັບກັບໄປຫາໂນດຝ່າຍ ຕາມກົດດັ່ງນີ້
ຖ້າໂນດຝ່າຍເປັນຜູ້ຫຼິນ (MAX) ຈະຄືນຄ່າ Maximum ຂອງ ໂນດ
ລູກ
ຖ້າໂນດຝ່າຍເປັນຜູ້ຫຼິນຝ່າຍກົງຂໍາມ (MIN) ຈະຄືນຄ່າ Minimum
ຂອງໂນດລູກ
4. MAX ຈະຖືກເລືອກ ໂນດທີ່ໃຫ້ຄ່າທາງໄປທີ່ສູງ (Maximum)
ທີ່ສຸດ

MINIMAX ALGORITHM

```
FUNCTION MINIMAX(N) {  
    IF N is a leaf then  
        return the estimated score of this leaf  
    ELSE  
        Let N1, N2, ..., Nm be the successors  
        of N;  
        IF N is a Min node then  
            return min{MINIMAX(N1), ...,  
            MINIMAX(Nm)}  
        ELSE  
            return max{MINIMAX(N1), ...}
```

Time Complexity: $O(b^m)$

Space Complexity: $O(bm)$

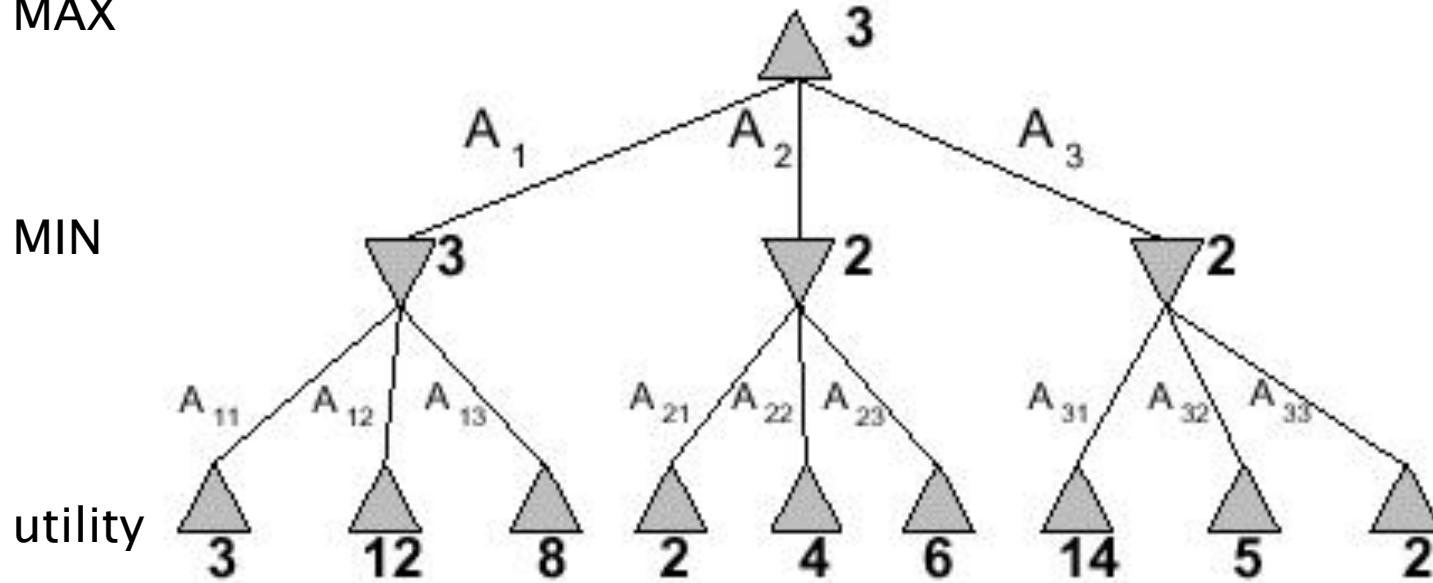
b ถ้าจำนวนโหนดลูก และ m ถ้า ความสูงของต้นไม้

ติอยปາງ

MAX

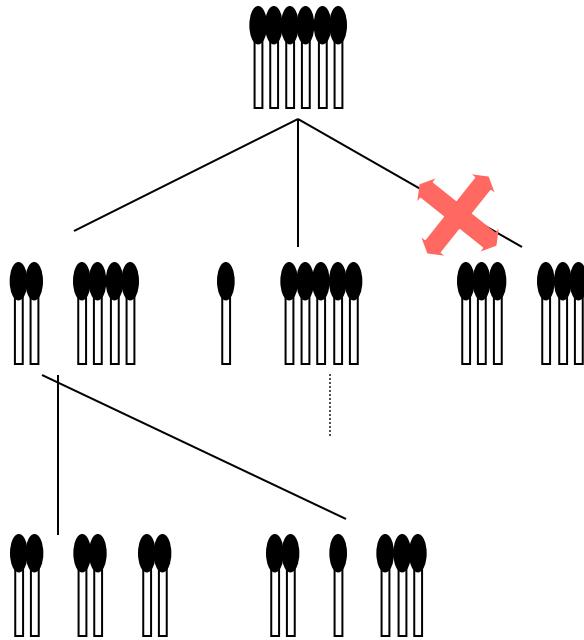
MIN

utility



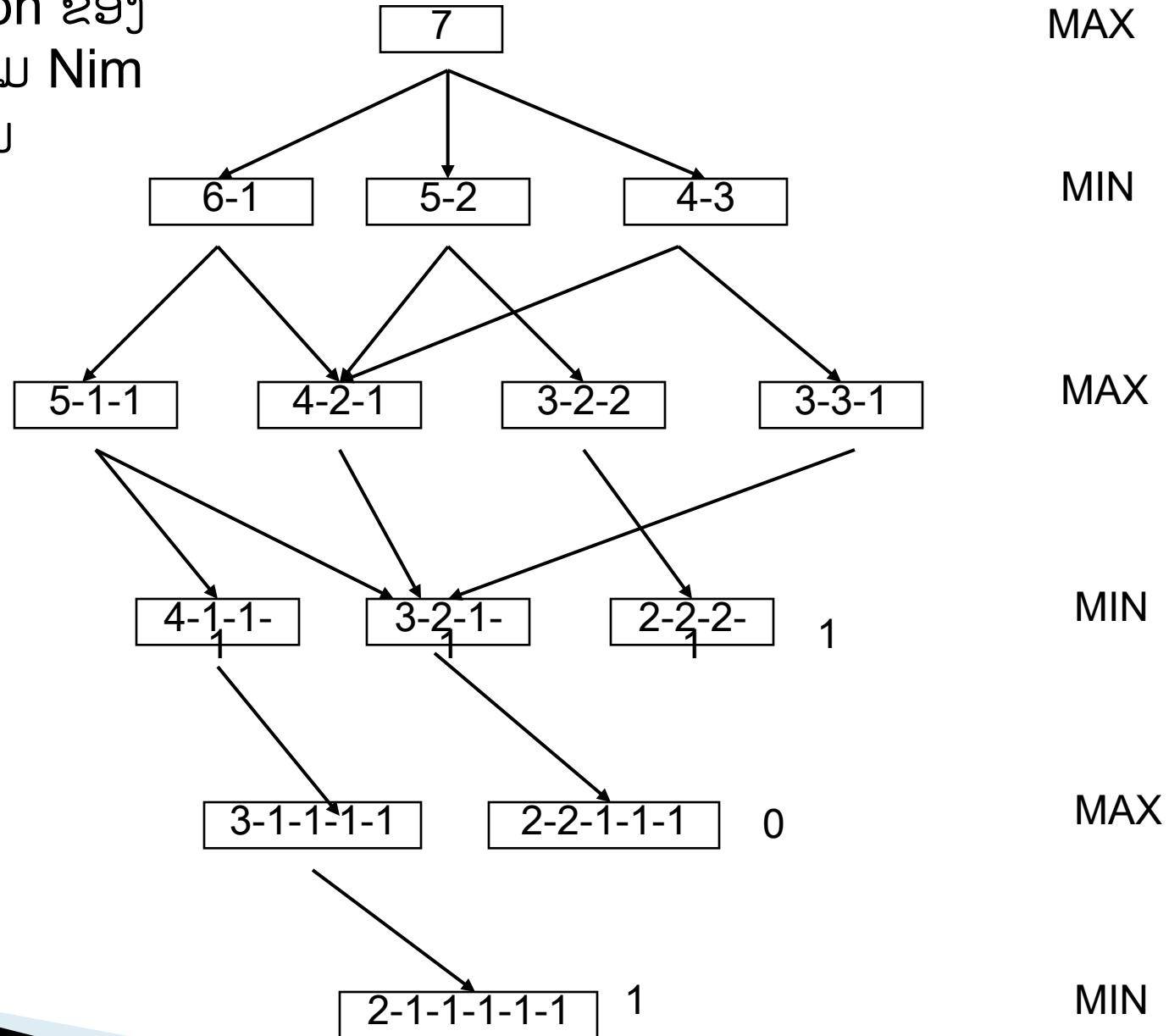
Utility function ໃຫ້ຄ່າຄວາມດີທີ່ terminal node ຈາກນັ້ນ ທີ່ລະດັບ A1 ຊຶ່ງເປັນ
MIN ຈະເລືອກຄ່າທີ່ນ້ອຍທີ່ສຸດ ສ່ວນໂນດ MAX ເທິງສຸດຈະສົ່ງຄ່າໂນດລູກໃຫ້ຄ່າໜ້າຍ
ທີ່ສຸດ

GAME: Game of Nim (ເກມແຫ່ງຄວາມຫຼັກແຫຼມ)



- ມີກອງໄມ້ຂີດຈຳນວນໜຶ່ງຢູ່ເກິງໂຕະ ລະຫວ່ງຜູ້ຫຼັນທັງ 2 ຄືນ
- ຜູ້ຫຼັນຈະປ່ຽນກັນແບ່ງຈຳນວນໄມ້ຂີດອອກເປັນກອງຍ່ອຍໆ
- ການແບ່ງແຕ່ລະເຫື້ອ ຈະຕ້ອງແບ່ງໃຫ້ໄມ້ຂີດຍໍ່ເກິງກັນ ແຊ່ນ ມີໄມ້ຂີດ 6 ກັນ ອາດແບ່ງເປັນ 5 -1 ຫຼື 4-2 ແຕ່ບໍ່ແມ່ນ 3-3
- ຄືນທີ່ບໍ່ສາມາດແບ່ງໄດ້ຈະບໍ່ຊະນະ

- ໂຄງສ້າງ graph ຂອງ
ການຄົ່ນຫາ ແລມ Nim
- ດຳເນີນ 7 ກໍານ



ການສ້າງໂຄງຮ່າງຕົນໄມ້ທັງໝົດເຮັດບໍ່ໄດ້



- ບ້າກເສີກ 1 ຕາ ຜູ້ຫຼັນສາມາດເຄືອນໄດ້ປະມານ 35 ວິທີ
- ໃນ 1 ກະດານ ສາມາດໄປໄດ້ ສະເລ່ຍ 50 ຄັ້ງ
- ດັ່ງນັ້ນ ຫາກໃຊ້ໂຄງຮ່າງຕົນໄມ້ຈຳລວງໂນດການເລືອກ ຈະຕ້ອງສ້າງໂນດຮອດ 35^{100} ໂນດ



ການສ້າງໂຄງຮ່າງຕົ້ນໄມ້ທັງໝົດເຮັດບໍ່ໄດ້



- ເຄື່ອງຄອມຜິວເຕີໂດຍທີ່ວໄປສາມາດຄົ້ນໄດ້ ປະມານ 1000 ໂນດໃນ 1 ວິນາທີ
 - ໃນການແຂ່ງຂັນ ແຕ່ລະຄົນມີເວລາ 150 ວິນາທີ ສໍາລັບການຍ້າຍ 1 ຄັ້ງ
 - ດັ່ງນັ້ນຄອມຜິວເຕີຈະຄຳນວນໄດ້ ປະມານ 150,000 ໂນດ ໃນ 1 ຕາ₂₂
 - ແຕ່ບໍ່ແມ່ນ 35^{100} ໂນດ

ເຮັດແນວໃດ ຈະໃຫ້ຄອມພິວເຕີ ຄໍານວນໄດ້

ເມື່ອເບິ່ງໄປທີ່ຈຸດສິ້ນສຸດຂອງເກມເຮັດບໍ່ໄດ້ ຈະເຮັດແນວໃດ ??

1. ຜັດທະນາໃຫ້ຄອມພິວເຕີຄໍານວນໃຫ້ໄວ??
2. ຫຼືເຫັນລ່ວງໜ້າໄປ ກ ລະດັບໃກ້??
3. ເມື່ອເບິ່ງລົງໄປ x ລະດັບ ທີ່ຫຼາຍກວ່າ ກ ແລ້ວຫຼຸດການຄໍານວນບາງສ່ວນລົງ??

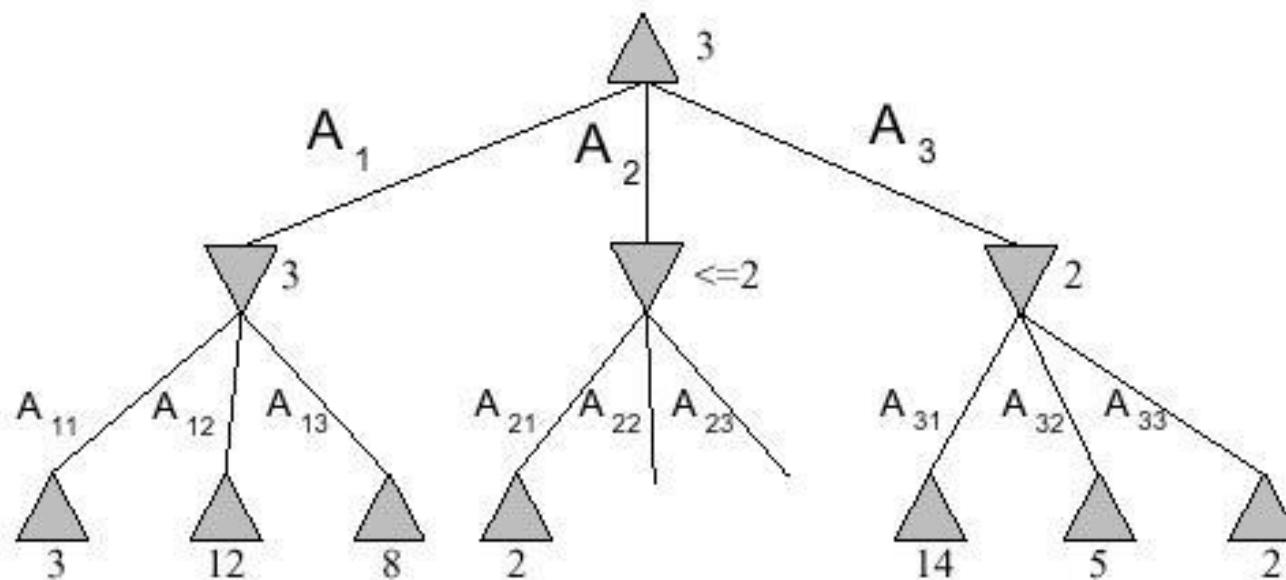
วิธีแก้ไข

- บ์ต้องใช้ Utility function ที่จะใช้ค่าเมื่อรถ
จุดสิ้นสุด
- แต่ใช้ Evaluation function ที่ใช้ประเมิน
โอกาสชนะ
- เทกนิก
 - เปรีงลิงไปที่ n จะดับไก้ (ใช้ได้แต่มีโอกาส
ชนะน้อย)
 - ใช้เทกนิก Alpha-beta cutoffs โดยเลือกบ่
ถำนวนบางโนดที่บ่ มีผิบต่ำกว่าคัดสิ้นใจ จะเพิ่ม
จะเพิ่มเวลาคำนวณท่าได้หลายชั้น m จะดับ

Alpha-beta cutoffs



- หูอี้มว่า Alpha-beta pruning
- ถ้ามีงานตัดสาขาที่บ่อมีผินตัวมาตัดสินใจออกໄປ เร็ดให้ ถ้าฯ งานฯ ลากตื๊ฯ



Alpha-beta cutoffs

- Algorithm นี้ปั้บปูງจาก Minimax Algorithm โดยเพิ่มการเรัดవຽກສໍາລັບເລືອກບໍ່ຄໍານວນບາງກິງ
- ຄ່າ α ແລະ β ຖືກເຜີມລົງໄປແຕ່ລະໂນດທີ່ກໍາລັງຄົ້ນຫາ
- ຄ່າ α ເກັບຄ່າທີ່ດີທີ່ສຸດຂອງ MAX ໂນດທີ່ເຄີຍເຫັນມາ ໂດຍທີ່ຄ່ານີ້ຈະບໍ່ມີຫາງຫຼຸດລົງ (MAX ຕ້ອງການເລືອກຄ່າທີ່ຫຼາຍທີ່ສຸດ)
- ຄ່າ β ເກັບຄ່າທີ່ດີທີ່ສຸດຂອງ MIN ໂນດທີ່ເຄີຍເຫັນມາ ໂດຍທີ່ຄ່ານີ້ຈະບໍ່ມີຫາງເຜີມຂຶ້ນ (MIN ຕ້ອງການເລືອກຄ່າທີ່ນ້ອຍທີ່ສຸດ)

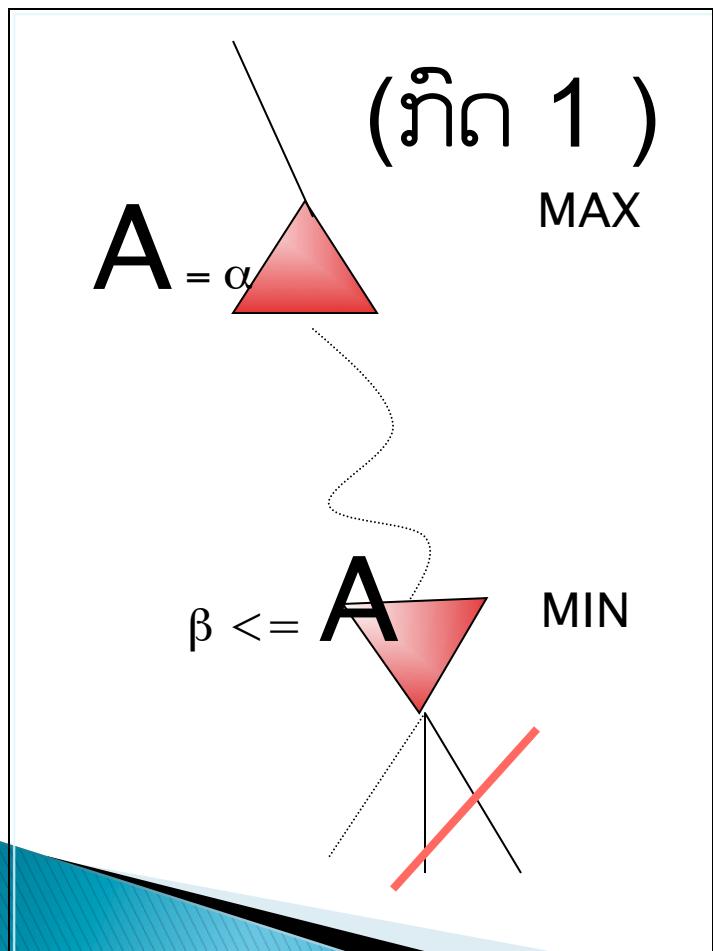
ກິດຂອງ Alpha-beta cutoffs

1. ການຄົ້ນຫາຈະຢຸດທີ່ໄມດໃດໜຶ່ງ ຂອງ MIN ເມື່ອ
ຄ່າ $\beta <= A$ ຂອງໂນດ MAX ຊຶ່ງເປັນໂນດຂອງຜ່ານ
*** A ຄົ້ນຄ່າ Alpha ຂອງໂນດຜ່ານ

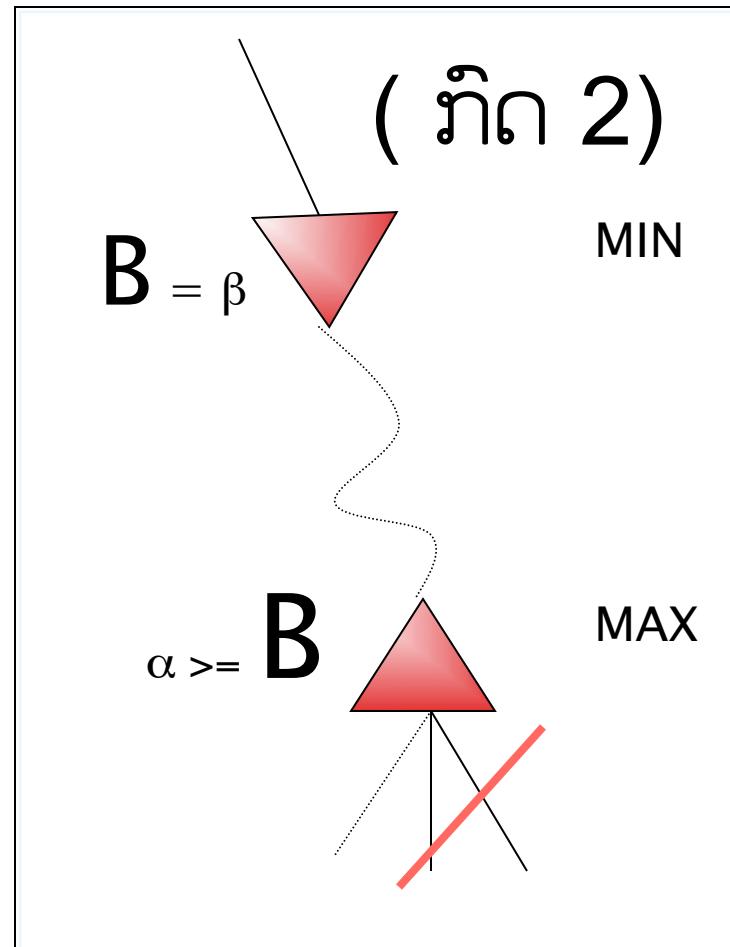
2. ການຄົ້ນຫາຈະຢຸດທີ່ໄມດຂອງ max ເມື່ອຄ່າ α
 $>= B$ ຂອງໂນດ MIN ຊຶ່ງເປັນໂນດຜ່ານ
*** B ຄົ້ນຄ່າ Beta ຂອງໂນດຜ່ານ

Alpha-beta cutoffs

ຕັດໄຟ MIN

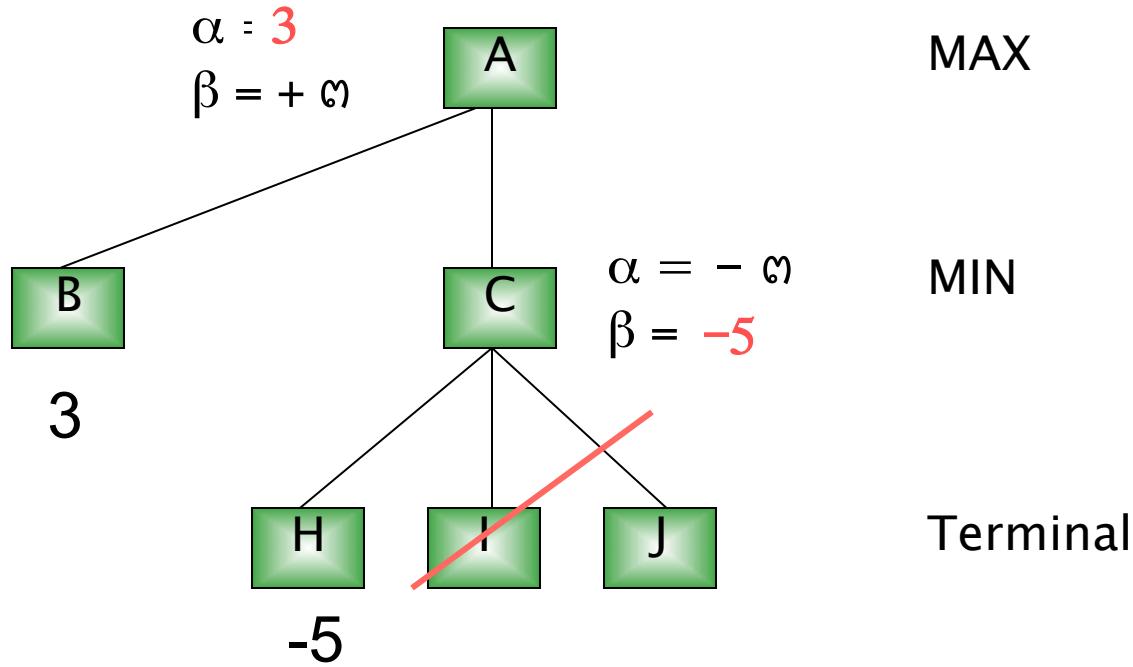


(ກີດ 2)



ຕັດໄຟ MAX

ຕົວຢາງການໃຊ້ ກິດທີ 1



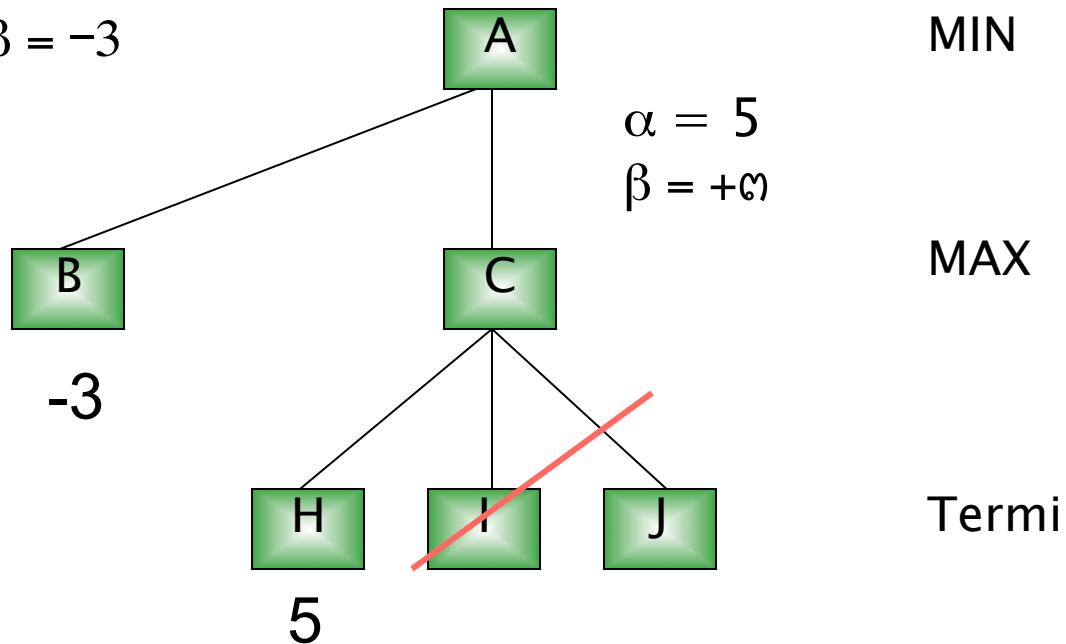
ຢຸດເມື່ອ $b \leq A$

A ຄືຄ່າ Alpha ຂອງໂນດັ່ງ

ຕົວຢາງການໃຊ້ ກິດທີ 2

$$\alpha = -3$$

$$\beta = -3$$



ປຸດເມື່ອ $a \geq B$

B ຄື່ອງ Beta ຂອງໄນດໍ່

Alpha–beta cutoffs Algorithm

```
function MINIMAX-AB(N, A, B) is // Here A is always less than B
```

```
begin
```

```
    if N is a leaf then
```

```
        return the estimated score of this leaf
```

```
    else
```

```
        Set Alpha value of N to -infinity and
```

```
        Beta value of N to +infinity;
```

```
        if N is a Min node then
```

```
            For each successor Ni of N loop
```

```
                Let Val be MINIMAX-AB(Ni, A, Min{B,Beta of N});
```

```
                Set Beta value of N to Min{Beta value of N, Val};
```

```
                When A >= Beta value of N then
```

```
                    Return Beta value of N endloop;
```

```
            Return Beta value of N;
```

```
        else
```

```
            For each successor Ni of N loop
```

```
                Let Val be MINIMAX-AB(Ni, Max{A,Alpha value of N}, B);
```

```
                Set Alpha value of N to Max{Alpha value of N, Val};
```

```
                When Alpha value of N >= B then
```

```
                    Return Alpha value of N endloop;
```

```
            Return Alpha value of N;
```

```
end MINIMAX-AB;
```

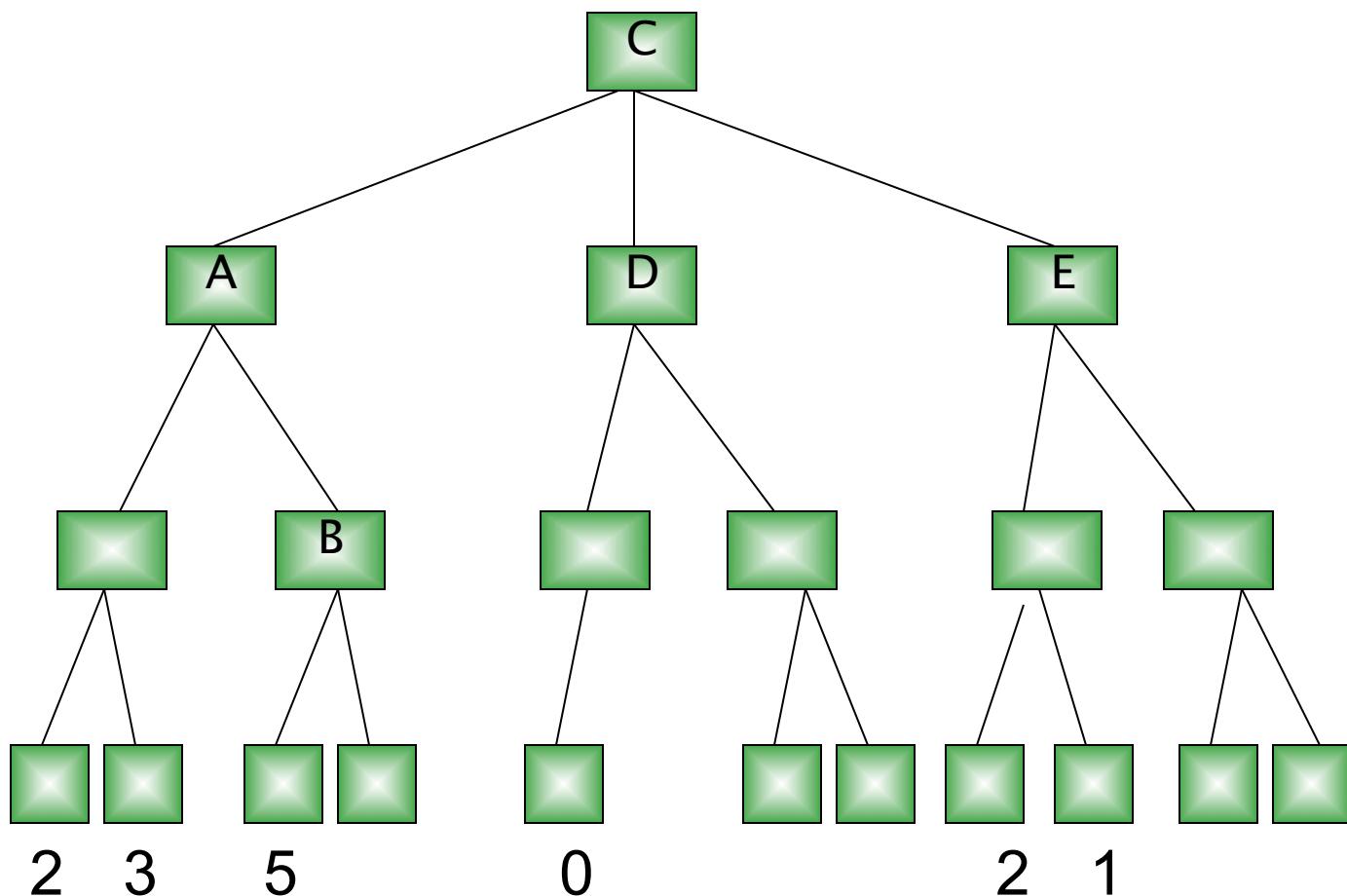
Example

MAX

MIN

MAX

MIN



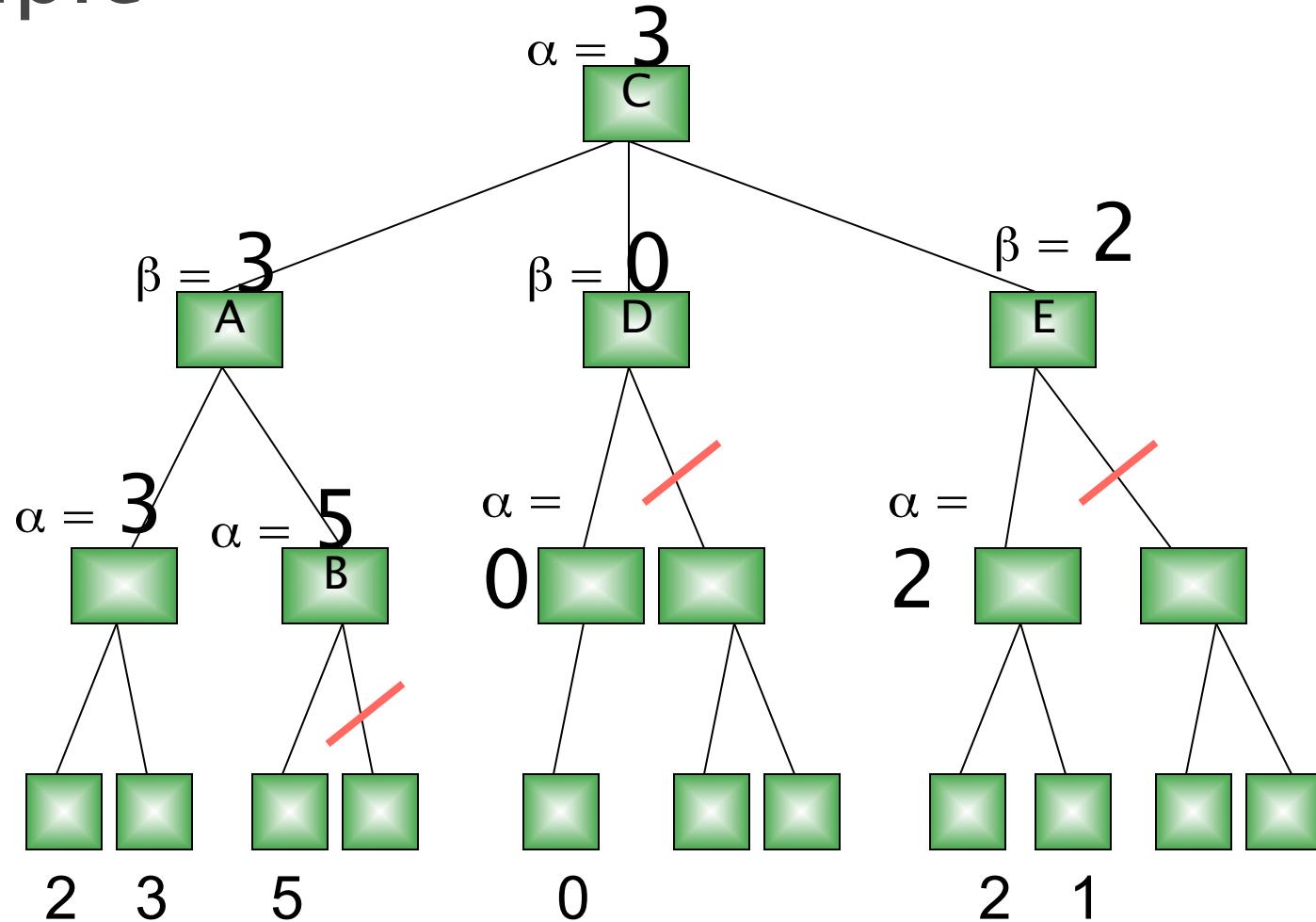
Example

MAX

MIN

MAX

MIN



END

