UNIVERSITATEA BABEŞ-BOLYAI
Facultatea de Matematică şi Informatică

INTELIGENȚĂ ARTIFICIALĂ

Rezolvarea problemelor de căutare

Strategii de căutare adversială

Laura Dioşan

Sumar

A. Scurtă introducere în Inteligenţa Artificială (IA)

B. Rezolvarea problemelor prin căutare
- Definirea problemelor de căutare
- Strategii de căutare
  - Strategii de căutare neinformate
  - Strategii de căutare informate
  - Strategii de căutare locale (Hill Climbing, Simulated Annealing, Tabu Search, Algoritmi evolutivi, PSO, ACO)
  - Strategii de căutare adversială

C. Sisteme inteligente
- Sisteme care învaţă singure
  - Arbori de decizie
  - Reţele neuronale artificiale
  - Maşini cu suport vectorial
  - Algoritmi evolutivi
- Sisteme bazate pe reguli
- Sisteme hibride

Materiale de citit şi legături utile

- capitolul II.5 din *S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995*
- capitolul 6 din *H.F. Pop, G. Şerban, Inteligenţă artificială, Cluj Napoca, 2004*
- documentele din directorul *06_adversial_minimax*

Conţinut

Jocuri

- Un pic de istorie
- Câteva repere teoretice
- Jocurile şi căutarea
  - Definiţii, componente, clasificare
  - Strategii şi algoritmi de căutare

Jocuri – un pic de istorie

- Provocări ale inteligenţei
  - Dame (Mesopotamia, cc. 3000 îHr.)
  - Go (China, sec. VI î.Hr.)
  - Sah (India, sec. VI d.Hr.)

- Tradiţional, jocurile erau văzute formal, ca o extensie a algoritmilor de căutare
  - Căutarea clasică: un singur agent, încearcă fără piedici să îşi atingă obiectivul
  - Jocurile: căutare în prezenţa unui adversar (agent ostil) + aduce incertitudine

- Jocurile şi IA
  - Proiectarea şi testarea algoritmilor
  - Unul dintre cele mai vechi domenii ale IA

Jocuri – un pic de istorie

- Jucarea jocurilor
  - De către oameni
    - Activitate inteligentă
    - Compararea aptitudinilor
  - De către calculatoare
    - Mediu propice pentru dezvoltarea tehnicilor de IA
      - Joc = problemă
        - structurată (succes sau eşec)
        - rezolvabilă prin căutare (simplă sau euristică)
    - Limite

Jocuri – câteva repere teoretice

- Probleme dificile cu o structură iniţială minimă de cunoştiinţe
- Stări şi acţiuni uşor de reprezentat
- Puţine cunoştiinţe necesare despre mediu
- Jocurile sunt un caz particular al problemelor de căutare
- Deseori au spaţii de căutare foarte mari
  - Complexitatea jocurilor → incertitudine
    - datorată insuficienţei de timp pentru a calcula consecinţele tuturor mutărilor şi
    - nu lipsei de informaţie
- Jocurile sunt interesante ☺

Jocurile şi căutarea

- Formalizare
- Reprezentarea jocurilor
- Tipuri de jocuri
- Spaţiul de căutare
  - Reprezentare
  - Metode de explorare

Jocurile şi căutarea – formalizare

- Rezolvarea unei probleme de căutare
  - Etapa x: definirea spaţiului de căutare
    - Spaţii liniare
    - Spaţii arborescente
      - Arbori
      - Grafe
  - Etapa x + 1: alegerea unei strategii de căutare
    - Care mutare/strategie?
      - Care determină câştigarea jocului
      - Care poate fi determinată cât mai repede (complexitate temporală redusă)
      - Care poate fi determinată cu efort fizic cât mai mic (complexitate spaţială redusă)

- Problema:
  - cum se poate determina cea mai bună mutare următoare într-un timp cât mai scurt?
- O soluţie:
  - căutarea între mutările posibile şi consecinţele lor

Jocurile şi căutarea – formalizare

- Căutarea adversială este folosită în jocurile în care unii jucători încearcă să-şi maximizeze scorul, dar se confruntă cu unul sau mai mulţi adversari

- Reprezentarea jocurilor ca probleme de căutare
  - **Stări**
    - configuraţiile (tablei) de joc + jucătorul care trebuie să mute
    - totalitatea stărilor posibile → arborele de căutare
  - **Operatori (acţiuni, funcţie succesor)**
    - mutările permise
  - **Starea iniţială**
    - configuraţia iniţială a jocului
  - **Starea scop (finală)**
    - configuraţia (câştigătoare) care termină jocul
  - **Funcţie de utilitate (funcţie scor)**
    - asociază o valoare numerică unei stări

- Dificultăţi
  - Jocurile pot fi probleme de căutare foarte dificile
    - totuşi usor de formalizat
  - Găsirea soluţiei optime poate fi nefezabilă
    - o soluţie care învinge adversarul este acceptabilă
    - o soluţie „inacceptabilă” nu numai că induce costuri mai mari, dar provoacă înfrângerea

Jocurile şi căutarea – formalizare

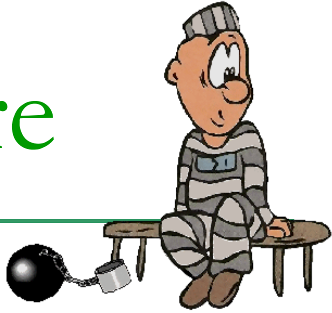
- Definiţii cheie
  - Situaţie conflictuală
    - Situaţie în care acţionează mai multe părţi care au scopuri contrare
    - Consecinţa acţiunii unei părţi depinde de reacţia celorlalte părţi
  - Joc
    - Modelare simplificată a unei situaţii conflictuale
    - Înşiruire de decizii (acţiuni, mutări) luate de părţi cu interese contrastante
  - Mutare
    - Funcţie H : {poziţiile jocului} → {poziţiile jocului}
    - Dacă p – o poziţie în joc, H(p) – o nouă poziţie
  - Reguli
    - Sistem de condiţii privind mutările posibile
  - Strategie
    - Ansamblu de reguli care definesc mutările libere în funcţie de situaţia concretă ivită

Jocurile şi căutarea – reprezentare

- Componente
  - Jucători
  - Mutări (strategii)
  - Criterii de câştig

- Forme de reprezentare pentru un joc
  - Forma strategică → matrice
    - Jucătorii
    - Strategiile
    - Câştigurile asociate fiecărui jucător şi fiecărei strategii
  - Forma extinsă → arbore
    - Nivel → jucător
    - Nod → alegere (mutare)

Jocurile şi căutarea – reprezentare



- Forma strategică → matrice
  - Ex. Dilema prizonierului: doi prizonieri sunt chestionaţi de poliţie. Poliţia ştie ceva despre ceea ce au făcut ei, dar nu are toate informaţiile. Pentru a afla, cei doi prizonieri sunt închişi în două celule şi sunt interogaţi. Prizonierii au 2 opţiuni:
    - Pot spune toaţă povestea (pot să se trădeze unul pe altul)
    - Pot să nu spună nimic (pot să coopereze)

    Nici un prizonier nu ştie ce va răspunde celălalt. În funcţie de răspunsurile lor, pedepsele sunt următoarele:
    - Dacă amândoi tac (cooperează), sentinţa este uşoară (1 an fiecare)
    - Dacă unul vorbeşte (trădează) şi unul tace (cooperează), trădătorul este eliberat, iar pârâtul primeşte 10 ani
    - Dacă amândoi vorbesc (se trădează reciproc), sentinţa este de 5 ani fiecare

    Ce vor face cei doi prizonieri?

J1 \ J2	Cooperare	Trădare
Cooperare	(1,1)	(10,0)
Trădare	(0,10)	(5,5)

Jocurile şi căutarea – reprezentare

- Forma strategică → matrice
  - Ex. Vânătoarea de cerbi: doi indivizi merg la vânătoare. Fiecare poate alege să vâneze:
    - un cerb sau
    - un iepure,
  - fără însă să ştie ce a ales celălalt individ. Vânarea
    - cerbilor, mai profitoare (câştig=4), se poate face doar în doi,
    - iepurilor, mai puţin valoroasă (câştig=1), se poate face individual.
  - Ce vor alege să vâneze cei doi indivizi?

J1 \ J2	Cerb	Iepure
Cerb	(4,4)	(1,3)
Iepure	(3,1)	(3,3)

Jocurile şi căutarea – reprezentare

- Forma strategică → matrice
  - Ex. economic
    - Reclama făcută de 2 firme pe aceeaşi piaţă
    - Înţelegerile la nivel de cartel pentru stabilirea preţurilor
  - Ex. sportiv
    - Doi ciclişti aflaţi în faţa plutonului poartă consecutiv trena (cooperează) pentru a nu fi ajunşi din urmă. De multe ori, doar unul duce trena (cooperează), iar la linia de sosire este trădat de adversar.

  - Ex. sociologic
    - Cand cunoaştem o nouă persoană, tindem să fim foarte atenţi pentru a avea o poziţie de siguranţă (competiţie). Ambele persoane pot semnala dorinţa de a se muta de la poziţiile defensive către
      - interacţiune şi
      - recunoaşterea unei intenţii comune.

Jocurile şi căutarea - tipologie

- Numărul jucătorilor
  - 1
  - 2
  - Mai mulţi
- Comunicarea între jucători
  - Cooperative
  - Ne-cooperative
- Strategiile de joc urmate
  - Simetrice
  - Asimetrice

Jocurile şi căutarea – tipologie

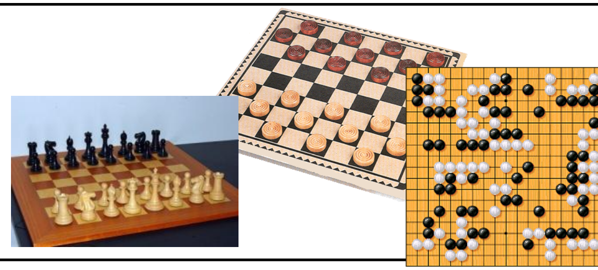
- Câştigurile jucătorilor
  - Sumă zero
    - câştigul unui jucător = pierderea celorlaţi jucători → un jucător trebuie să câştige sau jocul se sfârşeşte cu remiză
  - Sumă diferită de zero
    - câştigul unui jucător ≠ pierderea celorlaţi jucători

- Natura mutărilor efectuate
  - Cu mutări libere
    - mutarea este aleasă conştinet dintr-o mulţime de acţiuni posibile → jocuri deterministe
  - Cu mutări întâmplătoare
    - factor aleator (zar, cărţi de joc, monede) → jocuri non-deterministe

Jocurile şi căutarea – tipologie

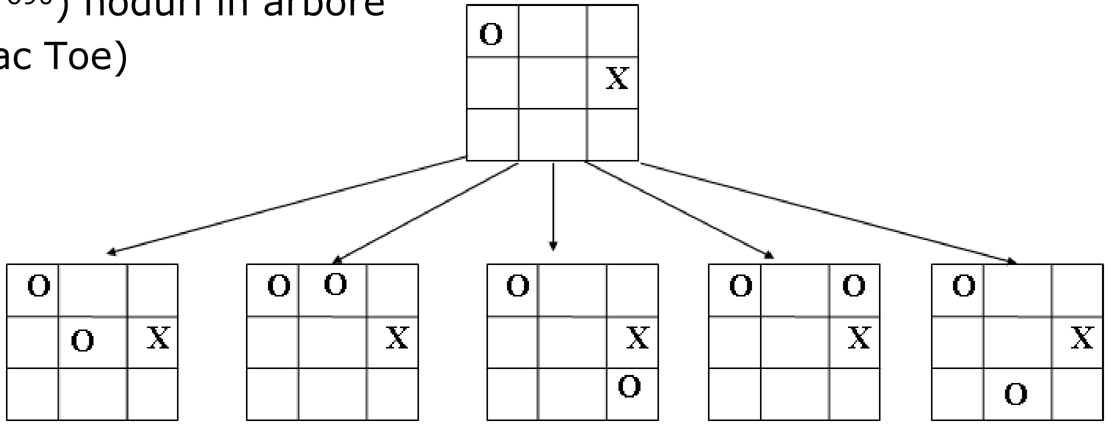
- Informaţiile despre joc
  - Cu informaţie perfectă
    - un jucător, înainte de a executa o mutare, cunoaşte rezultatele tuturor mutărilor precedente (ale lui şi ale adversarilor);
    - de obicei, jocul e cu mutări secvenţiale
  - Cu informaţie imperfectă
    - un jucător nu cunoaşte toate efectele mutărilor precedente

Jocurile şi căutarea – tipologie

	Deterministic	Non-deterministic (aleator)
Informaţie perfectă		 
Informaţie imperfectă	Vaporaşe/Avioane/ Submarine	 

Jocurile şi căutarea – spaţiul de căutare

- Reprezentare
  - Spaţiu liniar
  - Spaţiu arborescent → Arborele jocului
    - Identificarea strategiei de câştig → explorarea întregului arbore
    - foarte mare pt anumite jocuri
      - Şah („drosofila IA”)
        - factor de ramificare ≈ 35
        - ≈ 50 de mutări pe jucător
        - ≈ 35^{100} (10^{154}) noduri
        - 10^{40} noduri distincte (dimensiunea grafului de căutare)
      - Go
        - ≈ 200 mutări/stare, 300 niveluri
        - 200^{300} (10^{690}) noduri în arbore
    - exemplu – XO (Tic Tac Toe)



Jocurile şi căutarea – spaţiul de căutare

- Strategia de joc (a unui jucător)
  - Definire
    - ansamblul mutărilor unui jucător care
    - ţine cont de:
      - regulile jocului
      - starea curentă a jocului
    - ≠ mutare
  - Tipologie
    - Scop
      - Strategii pas cu pas
        - În fiecare etapă a jocului se identifică mutarea cea mai bună
      - Strategii complete
        - Se identifică o succesiune de mutări

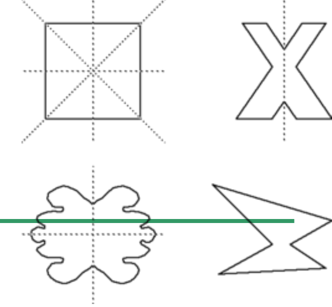
Jocurile şi căutarea – spaţiul de căutare

- Strategia de joc
  - Pas cu pas
    - Ex.: XO, Dame, Şah
    - Algoritmi – pot lucra cu structuri:
      - Liniare
        - Strategia simetriei
        - Strategia perechilor
        - Strategia parităţii
        - Programare dinamică
        - Alte strategii
      - Arborescente
        - Arbori AndOr
        - MiniMax (cu tăieturi Alpha-Beta)
  - Completă
    - Ex.: ieşirea dintr-un labirint, deplasarea pe o hartă între 2 locaţii date
    - Algoritmi pot să identifice
      - Un drum optim de la o locaţie la alta
      - O succesiune de acţiuni care să deplaseze jucătorul de la o locaţie la alta

Jocurile şi căutarea – spaţiul de căutare

- Strategia de joc
  - Pas cu pas
    - Ex.: XO, Dame, Şah
    - Algoritmi – pot lucra cu structuri:
      - **Liniare**
        - Strategia simetriei
        - Strategia perechilor
        - Strategia parităţii
        - Programare dinamică
        - Alte strategii
      - Arborescente
        - Arbori AndOr
        - MiniMax (cu tăieturi Alpha-Beta)
  - Completă
    - Ex.: ieşirea dintr-un labirint, deplasarea pe o hartă între 2 locaţii date
    - Algoritmi pot să identifice
      - Un drum optim de la o locaţie la alta
      - O succesiune de acţiuni care să deplaseze jucătorul de la o locaţie la alta

Jocurile şi căutarea – spaţiul de căutare



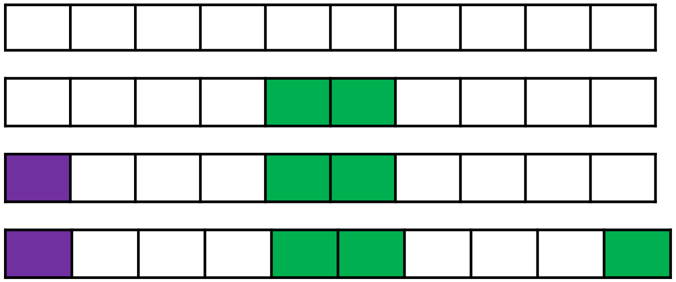
Strategii de joc → Strategia simetriei

- Aspecte teoretice
  - Jucătorul B imită mutările jucătorului A pe baza unei (unui) axe (centru) de simetrie
  - Dacă A mai poate muta, atunci şi B mai poate muta
    - Jocul se termină când A nu mai poate muta
  - E posibil ca prima mutare să nu aibă simetric

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Strategia simetriei

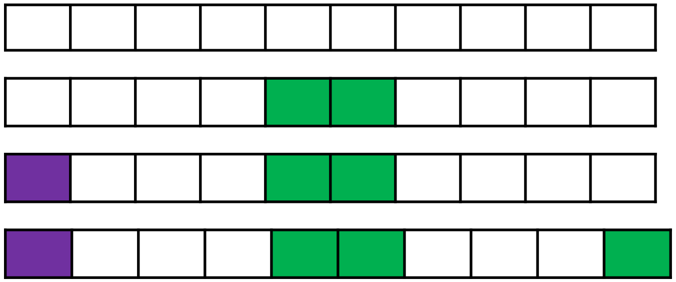
- Exemplu: jocul *haşurează căsuţe*
  - Se dă:
    - O bandă de hârtie este împărţită în *n* căsuţe. Alternativ doi jucători A şi B haşurează câte maxim *k* căsuţe adiacente, nehaşurate (*n* şi *k* au aceaşi paritate). Cel care nu mai poate muta pierde. Iniţial mută jucătorul A.
  - Se cere:
    - Să se determine dacă jucătorul A are strategie sigură de câştig. În caz afirmativ, să se programeze mutările lui A, cele ale lui B fiind citite de la tastatură.
  - Caz concret:
    - n = 10, k = 4



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Strategia simetriei

- Exemplu: jocul *haşurează căsuţe*
  - Se dă:
    - O bandă de hârtie este împărţită în *n* căsuţe. Alternativ doi jucători A şi B haşurează câte maxim *k* căsuţe adiacente, nehaşurate (*n* şi *k* au aceaşi paritate). Cel care nu mai poate muta pierde. Iniţial mută jucătorul A.
  - Se cere:
    - Să se determine dacă jucătorul A are strategie sigură de câştig. În caz afirmativ, să se programeze mutările lui A, cele ale lui B fiind citite de la tastatură.
  - Caz concret:
    - n = 10, k = 4



  - Soluţie
    - Dacă n – nr. par
      - Prima mutare → jucătorul A haşurează un nr. par de căsuţe din mijlocul benzii
      - La următoarele mutări jucătorul A îl imită pe jucătorul B simetric faţă de mijlocul benzii
    - Dacă n – nr. impar
      - Prima mutare → jucătorul A haşurează un nr. impar de căsuţe din mijlocul benzii
      - La următoarele mutări jucătorul A îl imită pe jucătorul B simetric faţă de mijlocul benzii

Jocurile şi căutarea – spaţiul de căutare

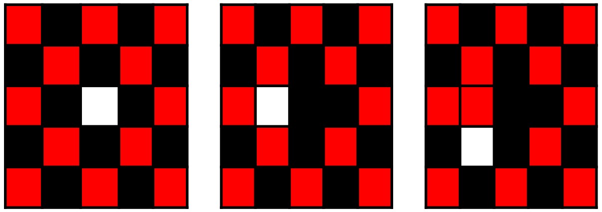
Strategii de joc → Strategia perechilor

- Aspecte teoretice
  - Generalizare a strategiei simetriei
  - Gruparea mutărilor în perechi:
    - (mutare jucător A, mutare jucător B)
  - Dacă A mai poate muta, atunci şi B mai poate muta
    - Jocul se termină când A nu mai poate muta
  - E posibil ca prima mutare să nu poată fi împerecheată

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Strategia perechilor

- Exemplu - Jocul *pătratelor alunecătoare*
  - Se dă:
    - În fiecare căsuţă a unei table pătratice *n*n* (*n* – nr. impar)se află un pătrat roşu sau negru astfel încât tabla se aseamănă cu o tablă de şah. Căsuţa din mijlocul tablei este goală (nu conţine un pătrat). Alternativ, doi jucători A (cu pătratele roşii) şi B (cu pătratele negre) mută câte unul din pătratele lor în căsuţa liberă de pe tablă. Pătratul mutat trebuie să se afle iniţial într-o căsuţă vecină (pe orizontală sau verticală) cu căsuţa liberă. Jucătorul care nu mai poate muta nici un pătrat de-al lui pierde jocul. Jucătorul B mută primul.
  - Se cere:
    - Să se determine dacă jucătorul A are strategie sigură de câştig. În caz afirmativ, să se programeze mutările lui A, cele ale lui B fiind citite de la tastatură.
  - Caz concret:
    - n = 5 (jocul propus iniţial de către G.W.Lewthwaite)

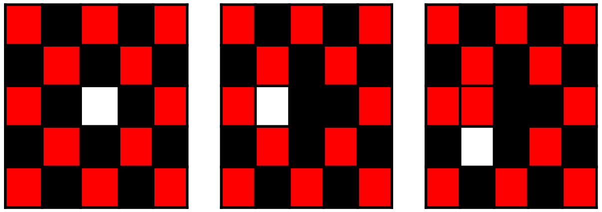


a b c

Jocurile şi căutarea – spaţiul de căutare

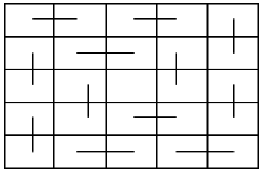
Strategii de joc → Strategia perechilor

- Exemplu - Jocul *pătratelor alunecătoare*
  - Se dă:
    - În fiecare căsuţă a unei table pătratice *n*n* (*n* – nr. impar)se află un pătrat roşu sau negru astfel încât tabla se aseamănă cu o tablă de şah. Căsuţa din mijlocul tablei este goală (nu conţine un pătrat). Alternativ, doi jucători A (cu pătratele roşii) şi B (cu pătratele negre) mută câte unul din pătratele lor în căsuţa liberă de pe tablă. Pătratul mutat trebuie să se afle iniţial într-o căsuţă vecină (pe orizontală sau verticală) cu căsuţa liberă. Jucătorul care nu mai poate muta nici un pătrat de-al lui pierde jocul. Jucătorul B mută primul.
  - Se cere:
    - Să se determine dacă jucătorul A are strategie sigură de câştig. În caz afirmativ, să se programeze mutările lui A, cele ale lui B fiind citite de la tastatură.
  - Caz concret:
    - n = 5 (jocul propus iniţial de către G.W.Lewthwaite)

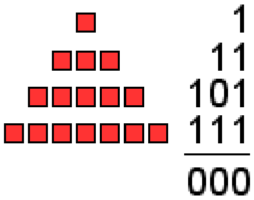


a b c

  - Soluţie
    - Se împarte tabla în *Domino-uri*, căsuţa din mijlocul tablei (iniţial goală) nefacând parte din nici un *Domino*.
    - După fiecare mutare a jucătorului B, căsuţa liberă se află acoperită de acelaşi *Domino* care acoperă şi o piesă de culoare roşie. Această piesă o va muta A, având în acest fel întotdeauna ceva de mutat.
    - acoperirea cu *Domino-uri* în spirală, plecându-se din colţul stânga sus, spre dreapta



Jocurile şi căutarea – spaţiul de căutare



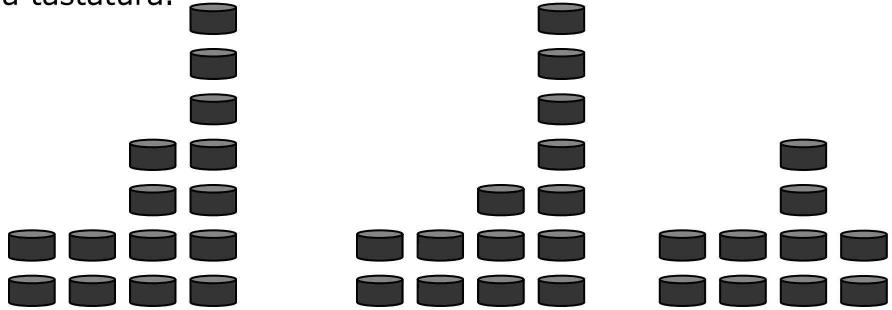
Strategii de joc → Strategia parităţii

- Aspecte teoretice
  - 2 tipuri de poziţii în joc
    - Poziţie pară (singulară)
    - Poziţie impară (nesingulară)
  - Teoreme
    - T1: poziţia impară, printr-o mutare **convenabilă**, → poziţie pară (jucătorul A)
    - T2: poziţia pară, prin **orice** mutare, → poziţie impară (jucătorul B)
  - Câştigător
    - → poziţia finală = poziţie pară

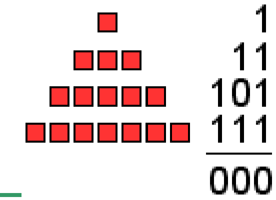
Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Strategia parităţii

- Exemplu - Jocul *NIM*
  - Se dau:
    - N stive, fiecare conţinând p_i obiecte. 2 jucători, A şi B, extrag, alternativ, dintr-o singură stivă, oricâte obiecte. Jucătorul care execută ultima extragere câştigă jocul. Jucătorul A mută primul.
  - Se cere:
    - Să se determine dacă jucătorul A are strategie sigură de câştig. În caz afirmativ, să se programeze mutările lui A, cele ale lui B fiind citite de la tastatură.
  - Exemplu:
    - 4 stive cu 2, 2, 4 şi, respectiv, 7 obiecte

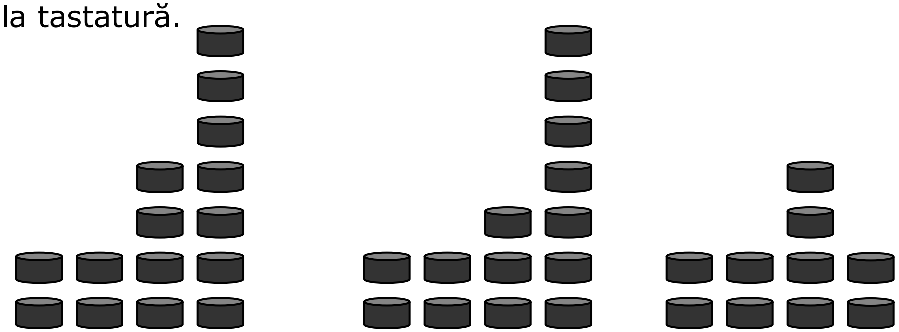


Jocurile şi căutarea – spaţiul de căutare



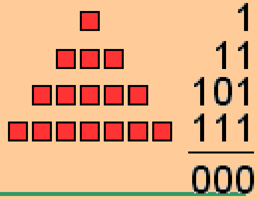
Strategii de joc → Strategia parităţii

- Exemplu - Jocul *NIM*
  - Se dau:
    - N stive, fiecare conţinând p_i obiecte. 2 jucători, A şi B, extrag, alternativ, dintr-o singură stivă, oricâte obiecte. Jucătorul care execută ultima extragere câştigă jocul. Jucătorul A mută primul.
  - Se cere:
    - Să se determine dacă jucătorul A are strategie sigură de câştig. În caz afirmativ, să se programeze mutările lui A, cele ale lui B fiind citite de la tastatură.
  - Exemplu:
    - 4 stive cu 2, 2, 4 şi, respectiv, 7 obiecte



  - Soluţie
    - Nr de obiecte din fiecare stivă se reprezintă binar (ca sumă a puterilor lui 2)
      - fiecare număr natural admite o descompunere unică ca sumă de puteri distincte ale lui 2
    - Stare pară – toate puterile lui 2 din reprezentarea jocului apar de un număr par de ori (în toate stivele)
    - Stare impară – orice stare care nu este pară
      - T1 → dintr-o stare impară se ajunge într-o stare pară printr-o mutare convenabilă
      - T2 → dintr-o stare pară se ajunge într-o stare impară prin orice fel de mutare
    - Extragerea de pe stiva care conţine cel mai seminificativ bit pe poziţia k
      - K poziţia celui mai semnificativ bit in suma NIM a tuturor stivelor

Jocurile şi căutarea – spaţiul de căutare



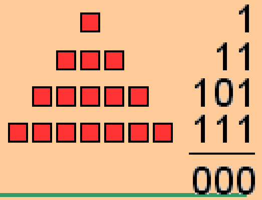
Strategii de joc → Strategia parităţii

- Exemplu - Jocul *NIM*
  - Demonstrarea celor 2 teoreme T1 şi T2
    - S_1: $2 = 2^1$, S_2: $3 = 2^1 + 2^0$, S_3: $4 = 2^2$, S_4: $5 = 2^2 + 2^0$
    - $2^0 - 2$, $2^1 - 2$, $2^2 - 2$ → stare pară

    - Extragerea dintr-o stivă cu număr par de obiecte
      - A unui număr par de obiecte (ex. 2 obiecte din S_3) = >
        - S_1: $2 = 2^1$, S_2: $3 = 2^1 + 2^0$, S_3: $4\text{-}2 = 2^1$, S_4: $5 = 2^2 + 2^0$
        - $2^0 - 2$, $2^1 - 3$, $2^2 - 1$ ➔ stare impară
      - A unui număr impar de obiecte (ex. 1 obiect din S_1)
        - S_1: $2\text{-}1 = 2^0$, S_2: $3 = 2^1 + 2^0$, S_3: $4 = 2^2$, S_4: $5 = 2^2 + 2^0$
        - $2^0 - 3$, $2^1 - 1$, $2^2 - 2$ ➔ stare impară

    - Extragerea dintr-o stivă cu număr impar de obiecte
      - A unui număr par de obiecte (ex. 2 obiecte din S_4) = >
        - S_1: $2 = 2^1$, S_2: $3 = 2^1 + 2^0$, S_3: $4 = 2^2$, S_4: $5\text{-}2 = 2^1 + 2^0$
        - $2^0 - 2$, $2^1 - 3$, $2^2 - 1$ ➔ stare impară
      - A unui număr impar de obiecte (ex. 1 obiect din S_2)
        - S_1: $2 = 2^1$, S_2: $3\text{-}1 = 2^1$, S_3: $4 = 2^2$, S_4: $5 = 2^2 + 2^0$
        - $2^0 - 1$, $2^1 - 2$, $2^2 - 2$ → stare impară

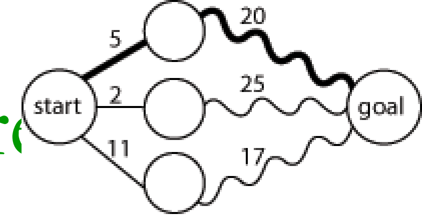
Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → Strategia parităţii

- Exemplu - Jocul *NIM*
  - algoritm
    - Pp. următorul joc:
      - S_1: 3 = 2^1+2^0 → 011, S_2: 4 = 2^2 → 100, S_3: 5 = 2^2+2^0 → 101
    - Paşi:
      1. Se calculează suma Nim (S_{Nim}) a tuturor stivelor
         - S_{Nim} = S_1 XOR S_2 XOR S_3 = 3 XOR 4 XOR 5 = 011 XOR 100 XOR 101 = 010= 2
      2. Se identifică poziţia *k* a celui mai reprezentativ 1 în suma Nim
         - S_{Nim} = 2 = $010_{(2)}$ ➔ poziţia a 2-a (k = 2)
      3. Se caută stiva cu cel mai reprezentativ bit pe poziţia k (stiva care descreşte dacă se face XOR între ea si suma Nim)
         - S_1: 3 = 010, S_2: 4 = 100, S_3: 5 = 101 => S_1 sau
         - S_1 XOR S_{Nim} = 3 XOR 2 = 011 XOR 010 = 001 = 1
         - S_2 XOR S_{Nim} = 4 XOR 2 = 100 XOR 010 = 110 = 6
         - S_3 XOR S_{Nim} = 5 XOR 2 = 101 XOR 010 = 111 = 7 ➔ S_1
      4. Se extrage din această stivă un număr de obiecte a.î. restul stivelor să formeze o stare pară; nr de obiecte care rămân pe stivă este dat de XOR-ul între stiva respectivă şi suma Nim
         - S_1 XOR S_{Nim} = 3 XOR 2 = 011 XOR 010 = 001 = 1 ➔ se extrag 2 (=3 – 1) obiecte de pe stiva 1
      5. Se reia pasul 1

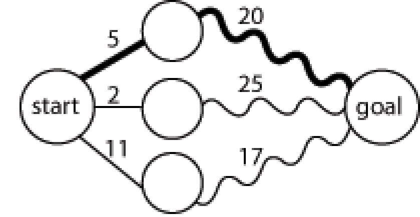
Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → Programare dinamică (PD)

- Aspecte teoretice
  - Paşi în rezolvarea unei probleme cu PD:
    - Descompunerea problemei în sub-probleme
    - Rezolvarea sub-problemelor
    - Combinarea sub-soluţiilor pentru obţinerea soluţiei finale
  - Paşi în rezolvarea unui joc cu PD:
    - Descompunerea jocului în sub-jocuri
    - Găsirea unei strategii perfecte de câştig pentru fiecare sub-joc
    - Combinarea sub-strategiilor pentru obţinerea strategiei finale

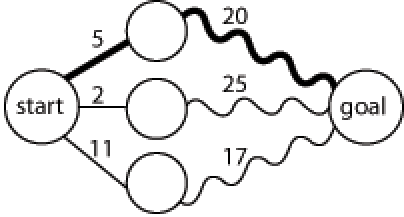
Jocurile şi căutarea–spaţiul de căutare



Strategii de joc → Programare dinamică (PD)

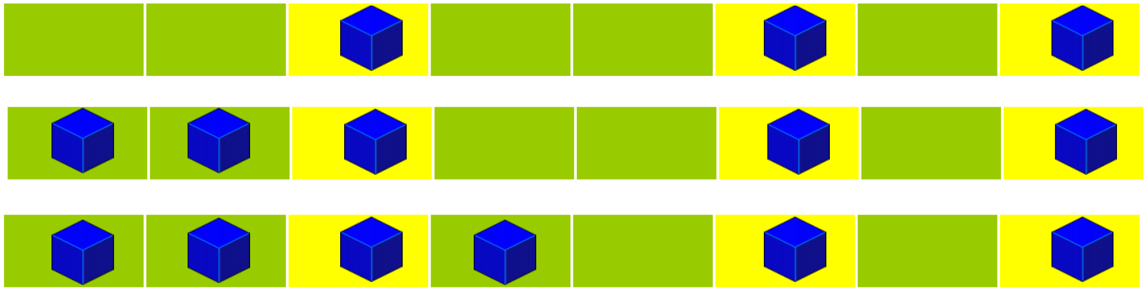
- Aspecte teoretice
  - Cum se rezolvă un joc cu programare dinamică?
    - descompunerea jocului în sub-jocuri J_h (sub-jocuri de pe nivelul cel mai de jos) şi
      - etichetarea fiecărui joc J_h cu T (*true*) sau F (*false*) în funcţie de posibilitatea jucătorului (care trebuie să mute din acea poziţie) de a avea strategie sigură de câştig
    - un sub-joc J_k de pe un nivel interior ($k < h$) va fi etichetat cu:
      - T, dacă există cel puţin un sub-joc J_i, $k < i$, etichetat cu F, în care poate fi transformat jocul J_k
      - F, dacă toate sub-jocurile J_i, $k < i$, în care se poate ajunge printr-o mutare din jocul J_k sunt etichetate cu T

Jocurile şi căutarea–spaţiul de căutare

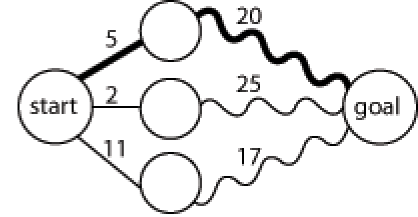


Strategii de joc → Programare dinamică (PD)

- Exemplu - Jocul *şirului de căsuţe*
  - Se dă:
    - În cele *n* căsuţe ale unui şir se află amplasate cuburi (maxim un cub în fiecare căsuţă). Scopul jocului este umplerea tuturor căsuţelor cu cuburi. Alternativ, doi jucători A şi B umplu cu cuburi (complet sau parţial) cel mai din stânga şir de căsuţe libere. Jucătorul care nu mai poate muta pierde. Jucătorul A mută primul.
  - Se cere:
    - Să se determine dacă jucătorul A are strategie sigură de câştig. În caz afirmativ, să se programeze mutările lui A, cele ale lui B fiind citite de la tastatură.
  - Exemplu
    - n = 8



Jocurile şi căutarea–spaţiul de căutare



Strategii de joc → Programare dinamică (PD)

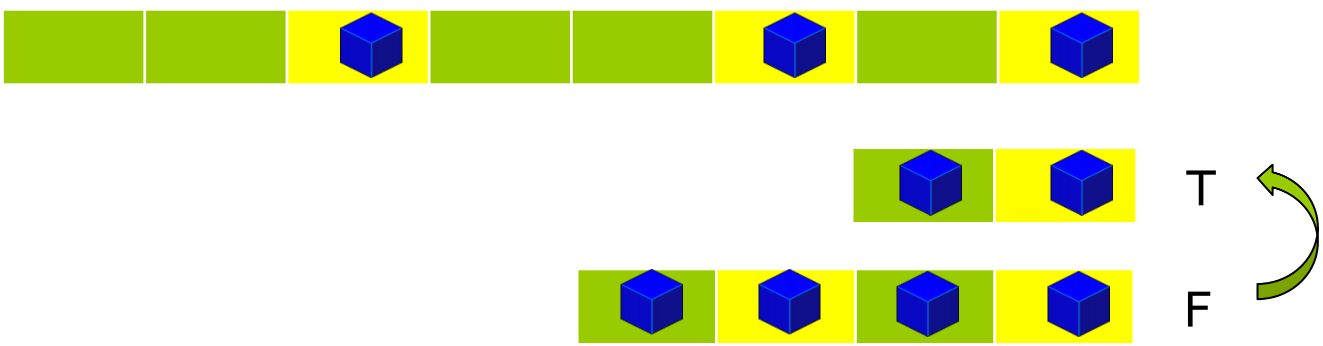
- Exemplu - Jocul *şirului de căsuţe*
  - Soluţie:

T

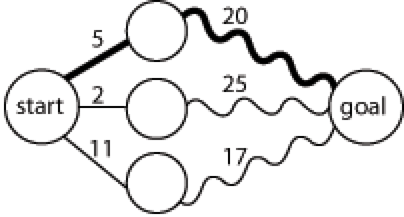
Jocurile şi căutarea–spaţiul de căutare

Strategii de joc → Programare dinamică (PD)

- Exemplu - Jocul *şirului de căsuţe*
  - Soluţie:

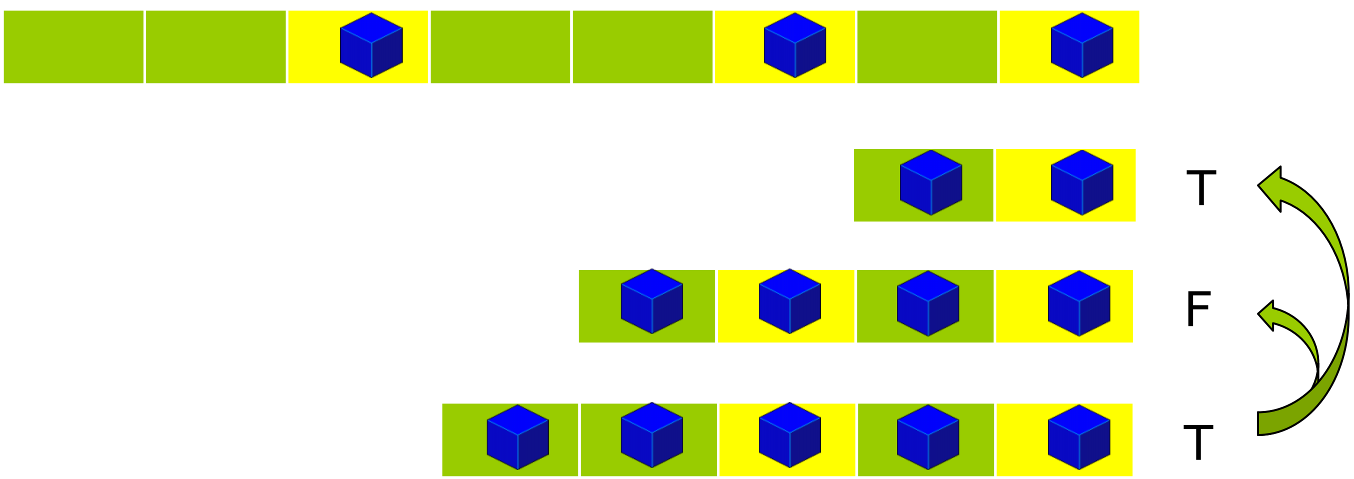


Jocurile şi căutarea–spaţiul de căutare



Strategii de joc → Programare dinamică (PD)

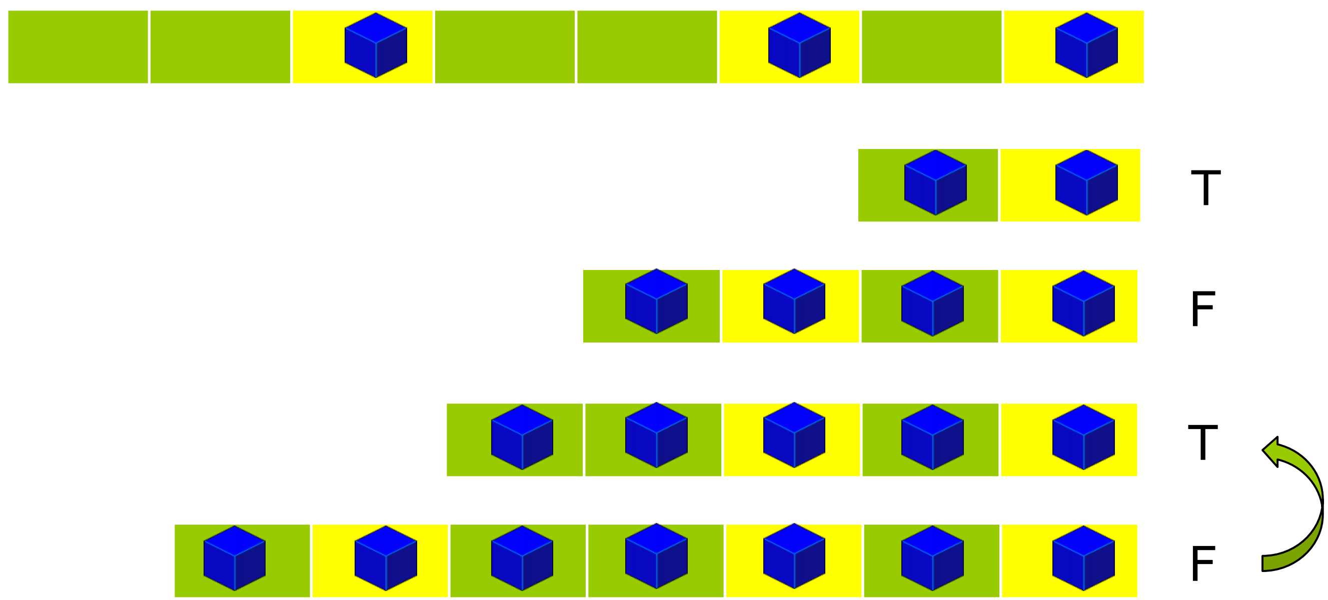
- Exemplu - Jocul *şirului de căsuţe*
  - Soluţie:

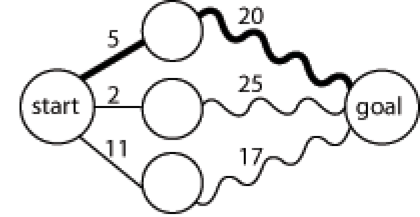


Jocurile şi căutarea–spaţiul de căutare

Strategii de joc → Programare dinamică (PD)

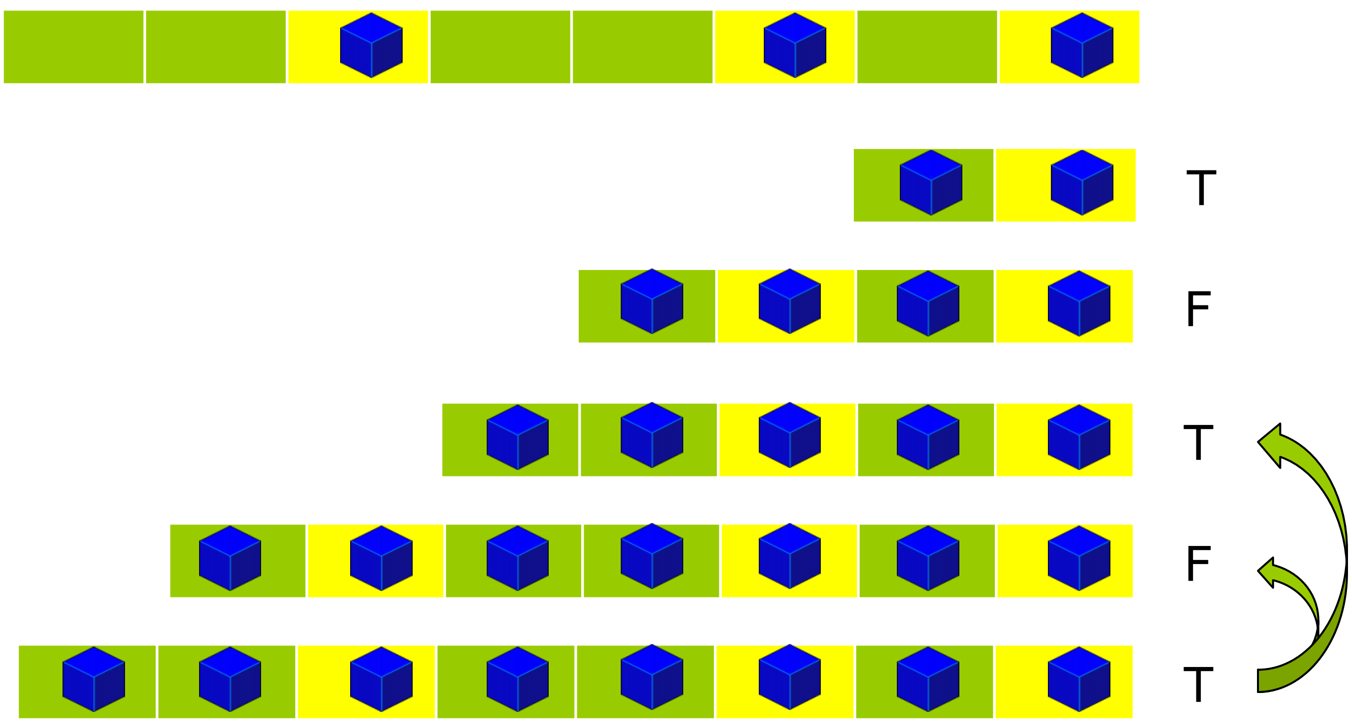
- Exemplu - Jocul *şirului de căsuţe*
  - Soluţie:





Strategii de joc → Programare dinamică (PD)

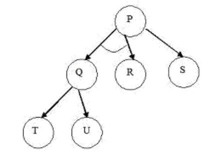
- Exemplu - Jocul *şirului de căsuţe*
  - Soluţie:



Jocurile şi căutarea – spaţiul de căutare

- Strategia de joc
  - Pas cu pas
    - Ex.: XO, Dame, Şah
    - Algoritmi – pot lucra cu structuri:
      - Liniare
        - Strategia simetriei
        - Strategia perechilor
        - Strategia parităţii
        - Programare dinamică
        - Alte strategii
      - **Arborescente**
        - **Arbori AndOr**
        - MiniMax (cu tăieturi Alpha-Beta)
  - Completă
    - Ex.: ieşirea dintr-un labirint, deplasarea pe o hartă între 2 locaţii date
    - Algoritmi pot să identifice
      - Un drum optim de la o locaţie la alta
      - O succesiune de acţiuni care să deplaseze jucătorul de la o locaţie la alta

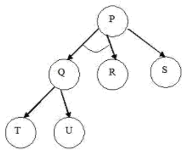
Jocurile şi căutarea–spaţiul de căutare



Strategii de joc - Arbori And-Or (AAO)

- Aspecte teoretice
  - **Reprezentarea spaţiului de căutare** cu ajutorul AAO pentru un joc cu 2 jucători:
    - Partea (nodul) OR
      - → selectarea unei mutări pentru jucătorul curent A
      - → există o posibilitate (mutare) pentru jucătorul A să ajungă într-un nod AND
    - Partea (nodul) AND
      - → considerarea tuturor mutărilor posibile ale adversarului (jucătorul B)
      - → prin orice mutare jucătorul B ajunge într-un nod OR
    - Astfel:
      - Rădăcina → problema jucătorului câştigător (care începe jocul din starea iniţială)
      - Mutările posibile ale primului jucător (A) se reunesc prin disjuncţie (OR)
      - Mutările posibile ale celui de-al doilea jucător (B) se reunesc prin conjuncţie (AND)
    - Jocul poate fi câştigat dacă începe cu un nod OR
  - **Dificultăţi:**
    - Arbori foarte foarte mari

Jocurile şi căutarea–spaţiul de căutare



Strategii de joc - Arbori And-Or (AAO)

- Aspecte teoretice
  - Paşi în rezolvarea unui joc cu AAO:
    - Descompunerea jocului în sub-jocuri
      - noduri pe mai multe nivele,
      - nodurile de pe un nivel corespunzând mutărilor posibile ale unui jucător
    - Găsirea unei strategii perfecte de câştig pentru fiecare sub-joc (nod)
      - Etichetarea nodurilor cu T sau F în funcţie de posibilitatea jucătorului A de a avea o strategie sigură de câştig pentru acel sub-joc
      - Reguli de etichetare:
        - Frunzele se etichtează cu T sau F în funcţie de configuraţia jocului
        - Nodurile interne se etichetează:
          - Pentru jucătorul A, cu T dacă **cel puţin un nod** fiu a fost etichetat cu T – regula OR
          - Pentru jucătorul B, cu T dacă **toate nodurile** fiu au fost etichetat cu T – regula AND
    - Combinarea sub-strategiilor pentru obţinerea strategiei finale

Jocurile şi căutarea–spaţiul de căutare

Strategii de joc - Arbori And-Or (AAO)

- Exemplu

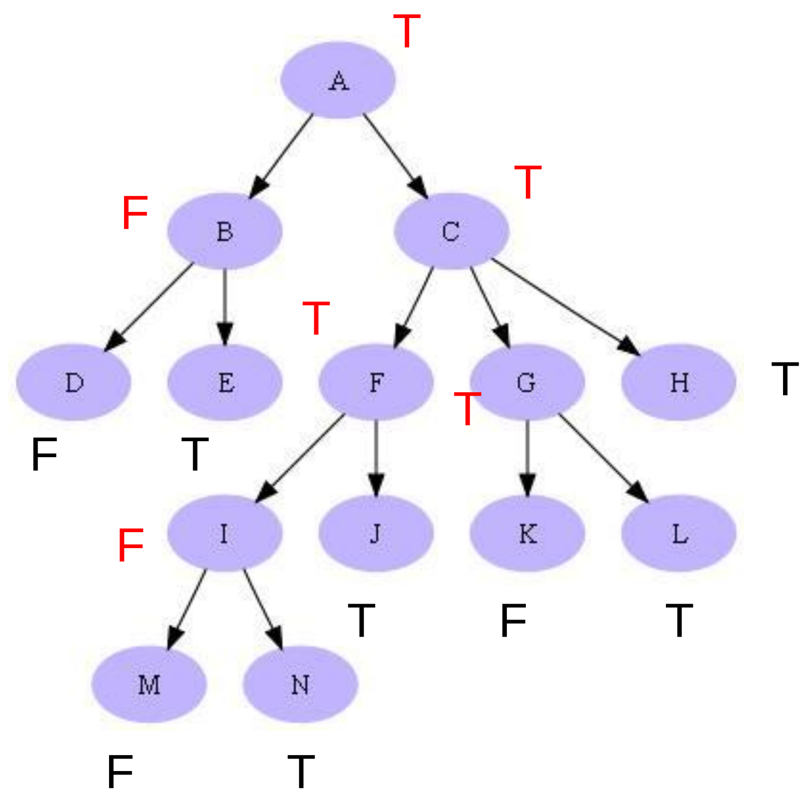
A (OR)

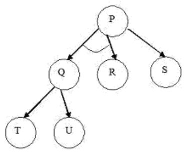
B (AND)

A (OR)

B (AND)

A (OR)

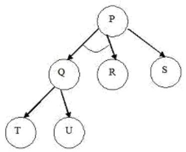




Strategii de joc - Arbori And-Or (AAO)

- Algoritm

```
bool backAndOr(Node N, int level){
        //level = 0 for the node in the top of the tree.
    if (N is a terminal) {
        if (the first player won)
                    return true;
        else
                    return false;
    }
    else{
        if (level % 2){            //B is about to move; AND
                    result = true;
                    for each child N_i of N
                                result = result && backAndOr(N_i, level+1);
        }
        else{                      // A is about to move; OR
                    result = false;
                    for each child N_i of N
                                result = result || backAndOr(N_i, level+1);
        }
        return result;
    }
}
```



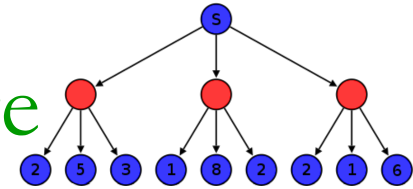
Strategii de joc - Arbori And-Or (AAO)

- Avantaje
  - Pot fi aplicaţi pentru rezolvarea oricărui joc
- Dezavantaje
  - Necesită multă memorie
  - Necesită mult timp de calcul
- →algoritmul mini-max

Jocurile şi căutarea – spaţiul de căutare

- Strategia de joc
  - Pas cu pas
    - Ex.: XO, Dame, Şah
    - Algoritmi – pot lucra cu structuri:
      - Liniare
        - Strategia simetriei
        - Strategia perechilor
        - Strategia parităţii
        - Programare dinamică
        - Alte strategii
      - Arborescente
        - Arbori AndOr
        - **MiniMax** (cu tăieturi Alpha-Beta)
  - Completă
    - Ex.: ieşirea dintr-un labirint, deplasarea pe o hartă între 2 locaţii date
    - Algoritmi pot să identifice
      - Un drum optim de la o locaţie la alta
      - O succesiune de acţiuni care să deplaseze jucătorul de la o locaţie la alta

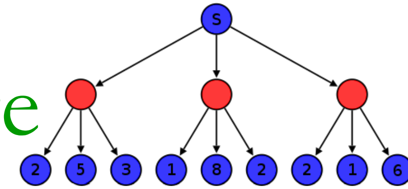
Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → MiniMax

- Aspecte teoretice
  - Propus de John von Neuman în 1944
  - Ideea de bază: maximizarea poziţiei unui jucător în timp ce poziţia adversarului este minimizată
  - Arborele de căutare constă în alternarea nivelelor pe care un jucător încearcă să-şi maximizeze câştigul cu nivelele pe care adversarul minimizează câştigul (primului jucător)
    - Primul jucător va încerca să-şi maximizeze câştigul (jucătorul MAX → mută primul)
    - Al doilea jucător va încerca să minimizeze câştigul primului jucător (jucătorul MIN → adversarul)
  - Identificarea celor mai bune mutări
    - Se construieşte arborele tuturor mutărilor posibile (fiecare mutare se aplică unei stări a jocului)
    - Se evaluează frunzele (care jucător câştigă)
    - Se propagă (în sus în arbore) câştigurile
    - Explorarea arborelui este de tip *Depth first search*
  - Generarea tuturor mutărilor
    - Posibilă doar în jocuri simple (ex. Tic-Tac-Toe)
    - Imposibilă (cu resurse limitate) pentru jocuri complexe
      - reţinând minimele în MIN
      - reţinând maximele în MAX

Jocurile şi căutarea – spaţiul de căutare

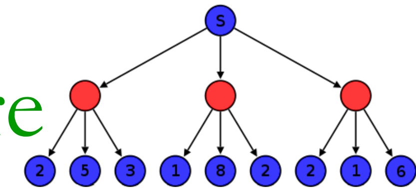


Strategii de joc → MiniMax

- Algoritm
  - Se construieşte arborele corespunzător tuturor mutărilor
  - Se evaluează frunzele
  - Cât timp se mai poate alege un nod cu toţi descendeţii evaluaţi
    - Se alege un nod
    - Dacă nodul este de pe nivel Min, el va fi evaluat la cea mai mică valoare a unui descendent
    - Dacă nodul este de pe nivel Max, el va fi evaluat la cea mai mare valoare a unui descendent
  - Se returnează valoarea nodului rădăcină (nod de pe nivel Max)

```
int backMiniMax(Node N, int level){
        //level = 0 for the node in the top of the tree.
    if (level == MaxLevel)
        return the quality of N computed with a heuristic;
    else if (level < MaxLevel){
        if (level % 2){
                                        //B is about to move; minimize
                    result = MaxInt;
                    for each child Ni of N
                                        result = minim(result, backMiniMax(Ni, level+1));
        }
        else{
                                        // A is about to move; Maximize
                    result = -MaxInt;
                    for each child Ni of N
                                        result = maxim(result, backMiniMax(Ni, level+1));
        }
        return result;
    }
}
```

Jocurile şi căutarea – spaţiul de căutare



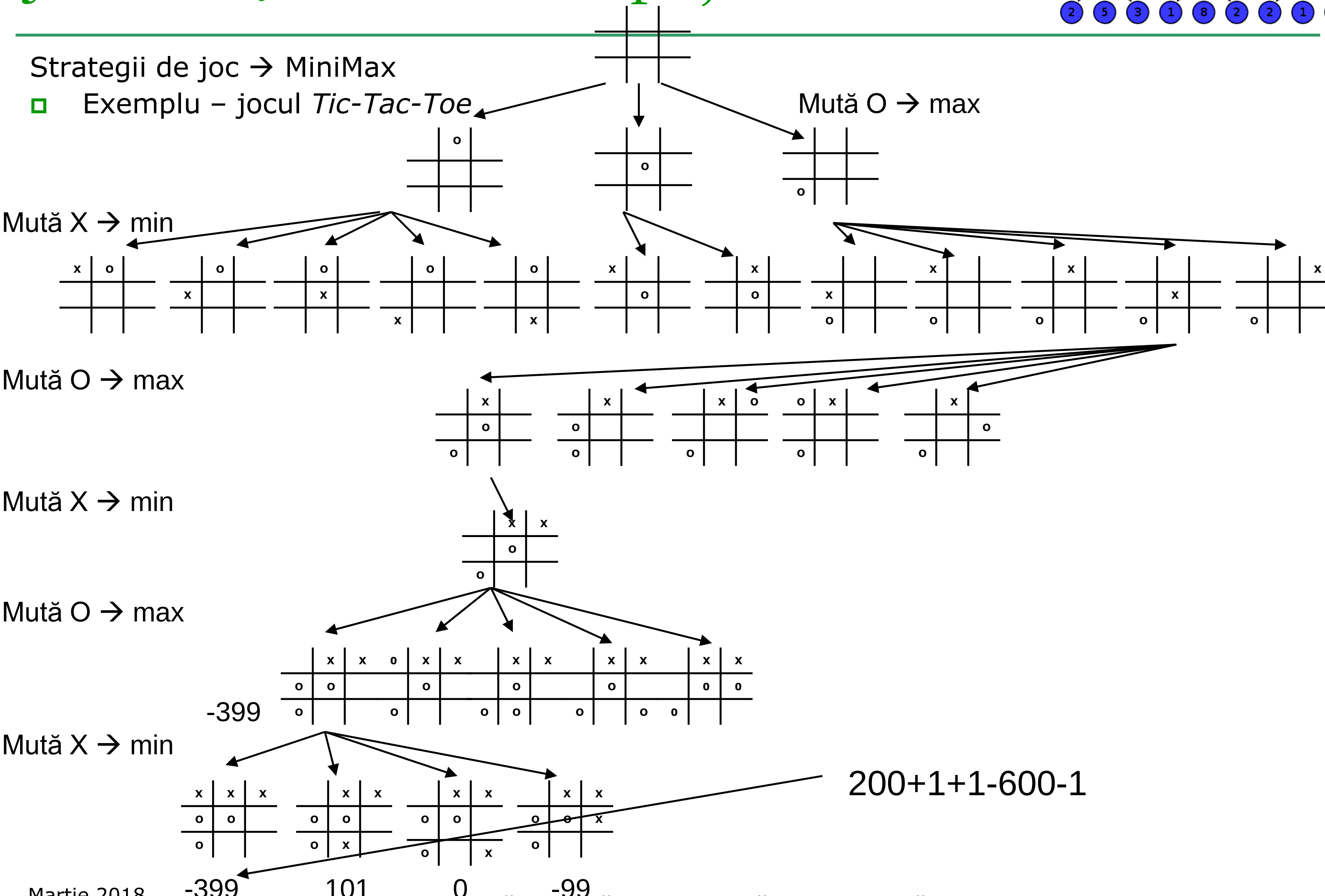
Strategii de joc → MiniMax

- Exemplu – jocul *Tic-Tac-Toe*
  - Funcţia de evaluare a unui nod:
    - Dacă există o linie aproape completă (cu 2 semne la fel) → 200 puncte
    - Dacă există 2 linii aproape complete (cu 2 semne la fel) → 300 puncte
    - O linie completă → 600 puncte
    - Fiecare linie potenţială → 1 punct
    - Punctele jucătorului care urmează la mutare se adună
    - Punctele celuilalt jucător se scad

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax

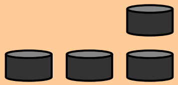
- Exemplu – jocul *Tic-Tac-Toe*



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax

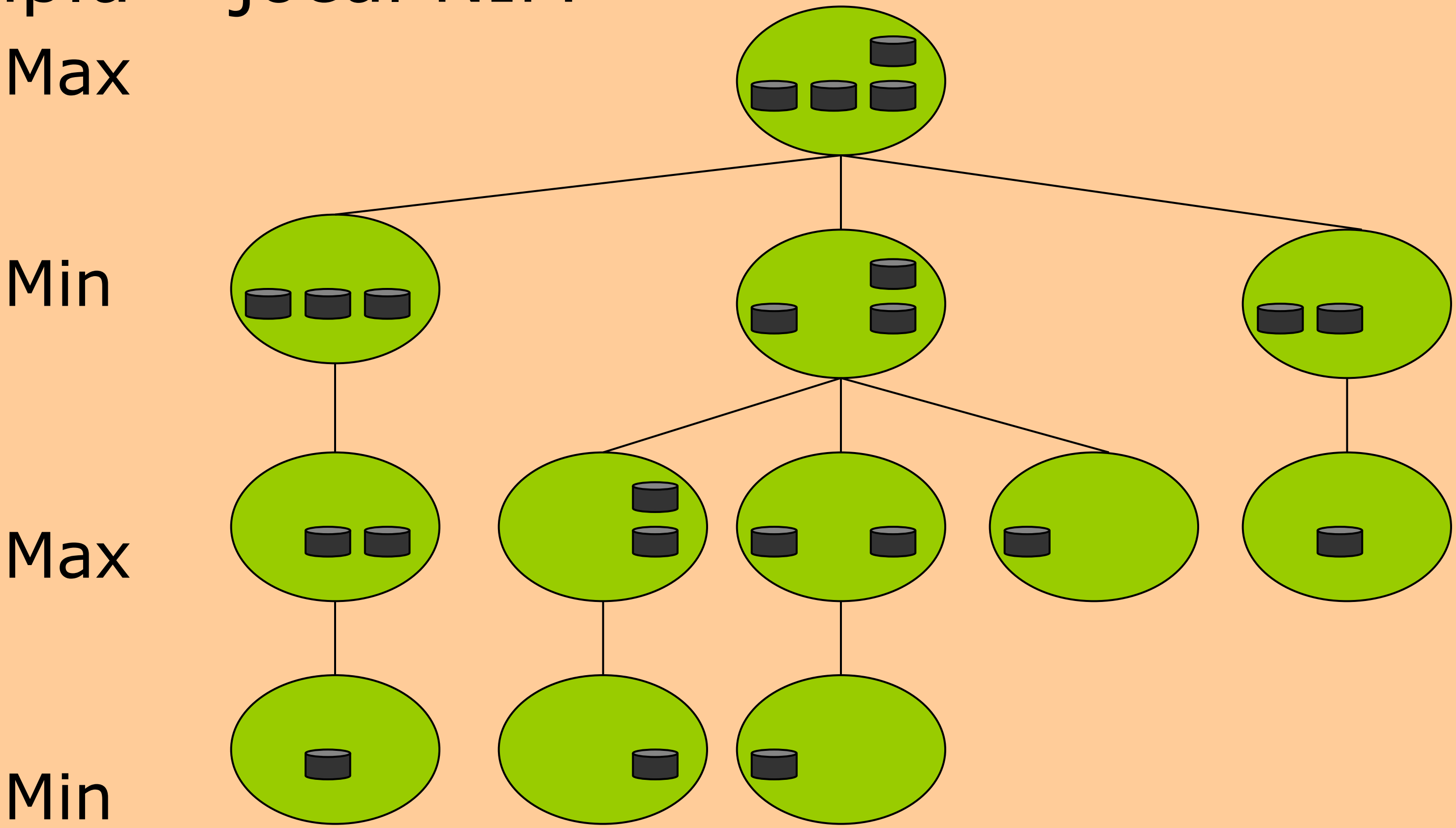
- Exemplu – *Jocul NIM*
  - Se dă:
    - N stive conţin fiecare p_i obiecte. 2 jucători A şi B extrag, alternativ, dintr-o singură stivă oricâte obiecte. Jucătorul care execută ultima extragere câştigă jocul. Jucătorul A mută primul.
  - Se cere:
    - Să se determine dacă jucătorul A are strategie sigură de câştig. În caz afirmativ, să se programeze mutările lui A, cele ale lui B fiind citite de la tastatură.
  - Exemplu:
    - 3 stive cu 1, 1 şi, respectiv, 2 obiecte



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax

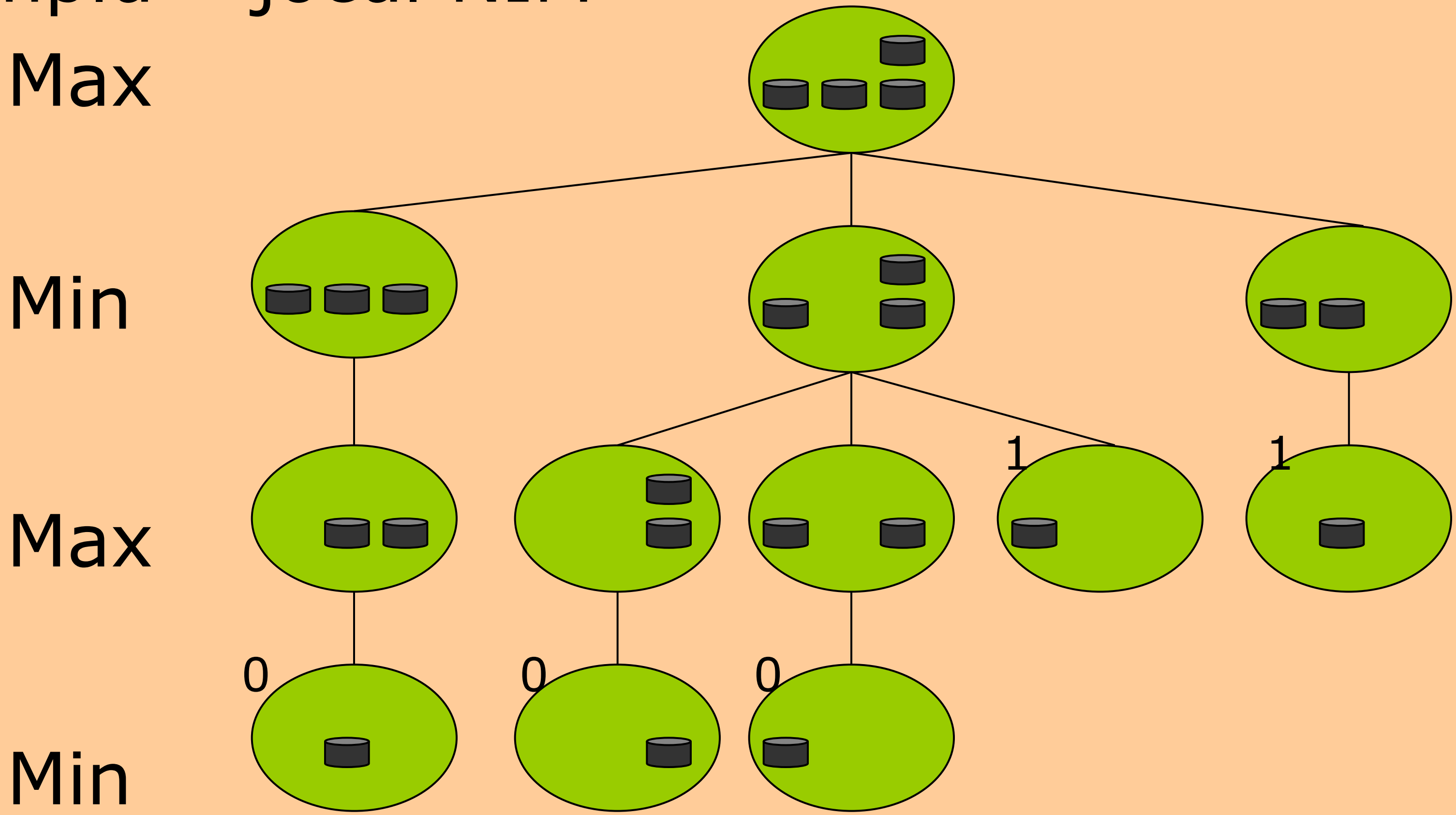
- Exemplu – jocul NIM



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax

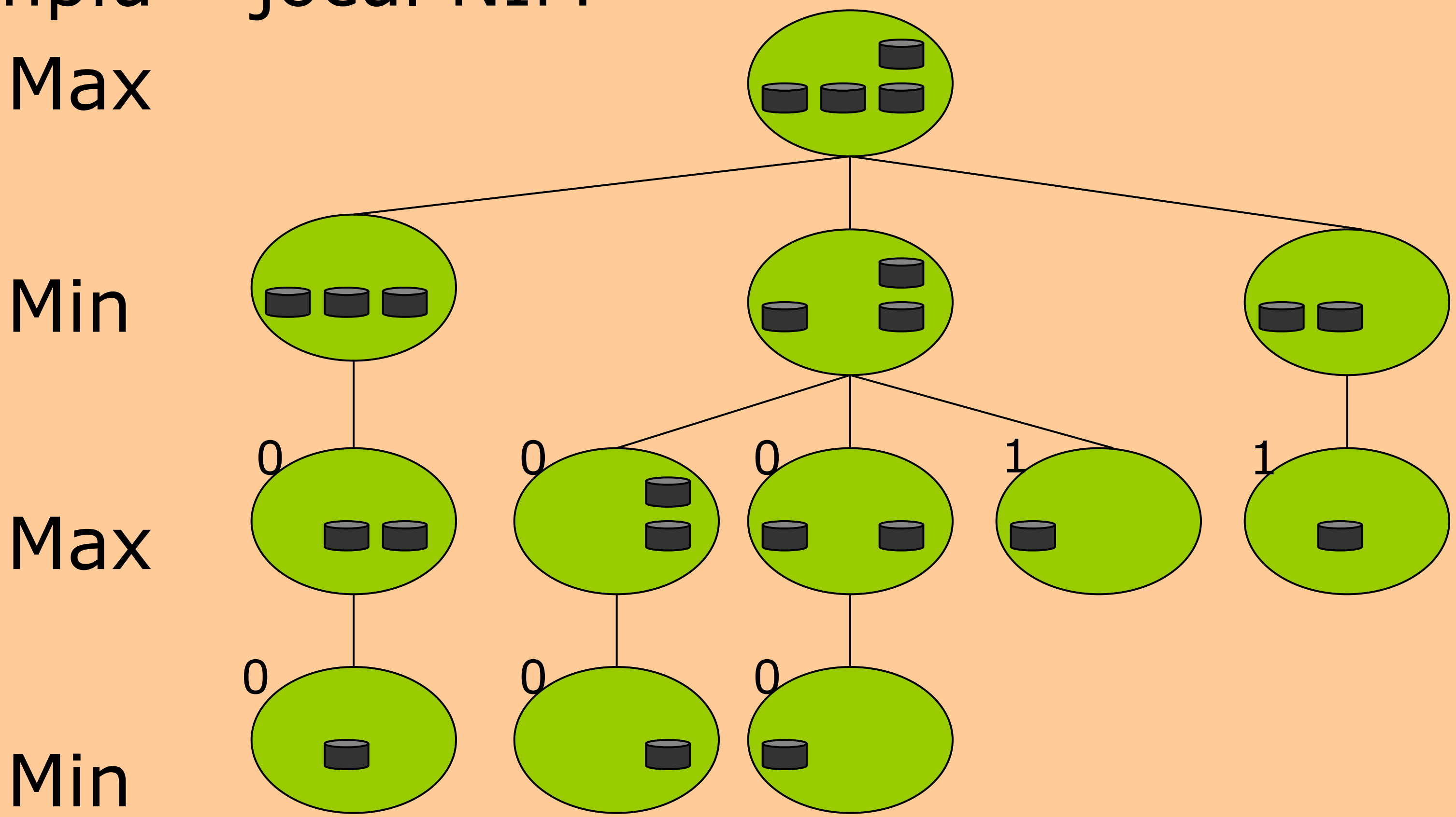
- Exemplu – jocul NIM



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax

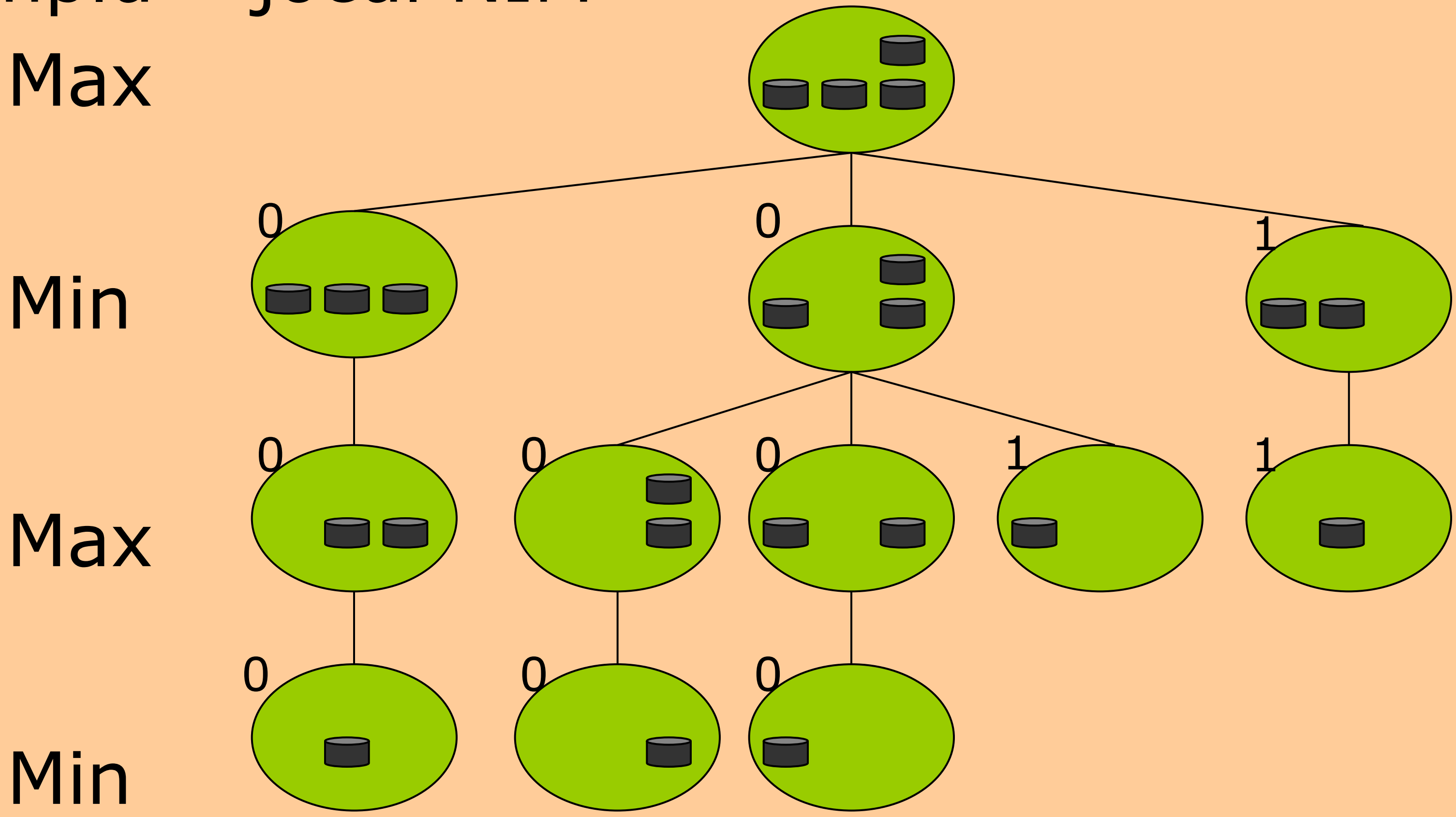
- Exemplu – jocul NIM



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax

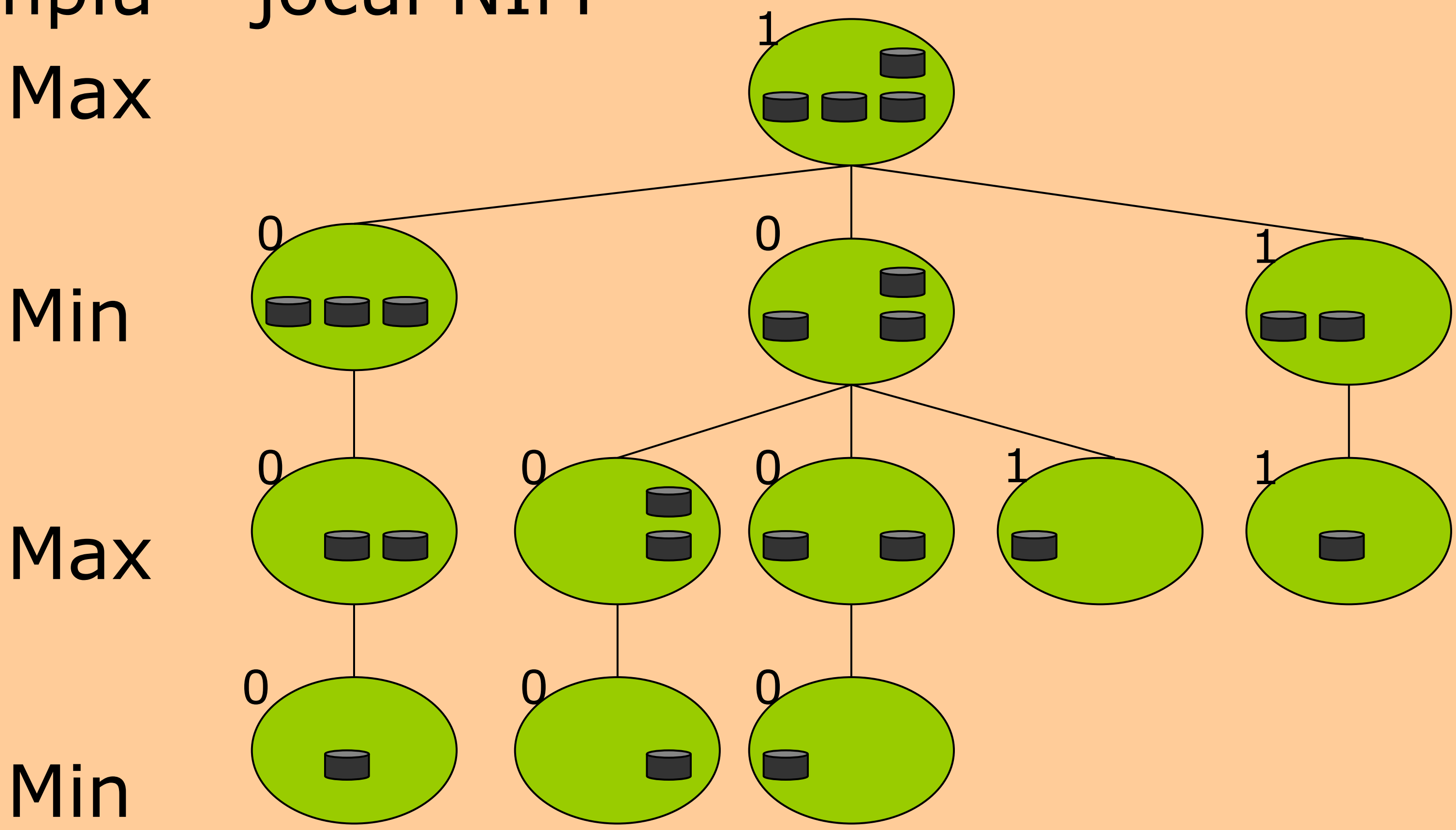
- Exemplu – jocul NIM



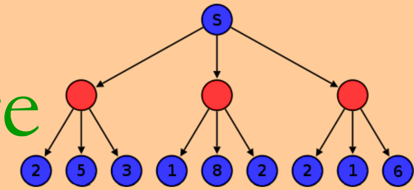
Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax

- Exemplu – jocul NIM



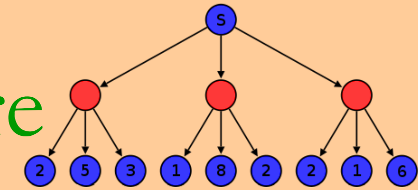
Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → MiniMax

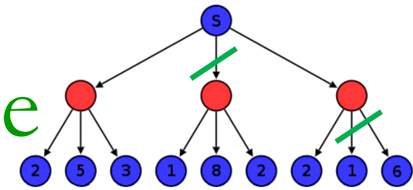
- Dificultăţi
  - Nu explorează întregul arbore
    - Căutare depth-first
  - La începutul jocului se fixează:
    - O adâncime maximă
      - Care este adâncimea optimă?
        - Adâncime mare → căutare lungă
        - Adâncime mică → anumite drumuri pot fi ratate (sacrificii timpurii pentru câştiguri târzii)
    - O funcţie de evaluare
      - Fiecare nod (stare) trebuie evaluat(ă)
      - Cum? ➔ folosind euristici

Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → MiniMax

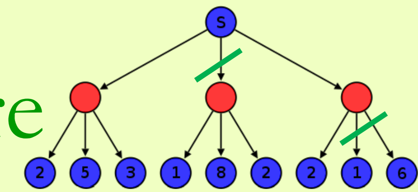
- Dezavantaje
  - Doar 100 noduri sunt explorate intr-o secundă
  - Sah – timp mutare = 150s → 150 000 de poziţii → doar 3 - 4 mutări în avans
- Soluţia
  - Evitarea anumitor noduri prin retezarea unor ramuri (redundante) ➔ retezare (reducere) alpha-beta ➔ MiniMax cu retezare α-β



Strategii de joc → MiniMax cu retezare α-β

- Minimax
  - Crează întreg arborele
  - Se propagă valorile de jos în sus

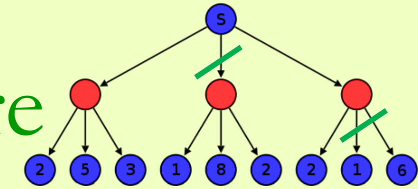
- MiniMax cu retezare α-β
  - Crearea şi propagarea în acelaşi timp
  - Dacă se ştie că un drum este rău, nu se mai consumă energie ca să se afle cât de rău este acel drum
    - Se pot elimina anumite evaluări inutile
    - Se pot elimina anumite expandări ale nodurilor



Strategii de joc → MiniMax cu retezare α-β

- Apecte teoretice
  - Descoperit de McCarthy în 1956, dar publicat pentru prima dată într-un raport tehnic de la MIT
  - Ideea de bază
    - Valoarea minimax a rădăcinii poate fi determinată fără examinarea tuturor nodurilor de pe frontiera căutării
    - Similar algoritmului *branch and bound*
  - Denumirea α-β
    - α - valoarea celei mai bune alegeri (celei mai mari) efectuate de-a lungul drumului urmat de MAX
      - dacă o valoare unui nod este mai slabă decât α atunci MAX o va evita şi va reteza sub-arborele cu rădăcina în acel nod
    - β - similar lui α, dar pentru jucătorul MIN

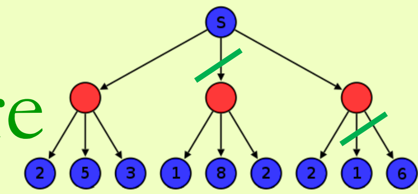
Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → MiniMax cu retezare α-β

- Aspecte teoretice
  - α
    - valoarea cele mai bune (adică cea mai mare) alegeri găsită până la momentul curent în orice punct de-a lungul unui drum pentru MAX
    - scorul minim pe care jucătorul MAX îl poate obţine în mod garantat
    - dacă un nod *v* este mai slab (mai mic) decât α, MAX îl va evita prin eliminarea acelei ramuri →
      - Dacă s-a ajuns cu căutarea într-un nod MIN a cărui valoare $\leq \alpha$ atunci descendenţii acelui nod nu mai merită exploraţi pentru că oricum MAX îi va ignora
  - β
    - cea mai mică valoare găsită la orice punct de-a lungul unui drum pentru MIN
    - scorul minim pe care jucătorul MAX speră să-l obţină
    - dacă un nod *v* este mai bun (mai mare) decât β, MIN îl va evita prin eliminarea acelei ramuri →
      - Dacă s-a ajuns cu căutarea într-un nod MAX a cărui valoare $\geq \beta$ atunci descendenţii acelui nod nu mai merită exploraţi pentru că oricum MIN îi va ignora

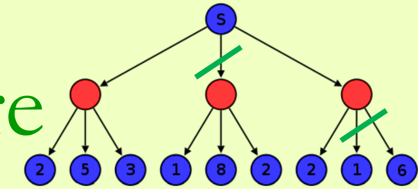
Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → MiniMax cu retezare α-β

- Aspecte teoretice
  - Valoarea α a unui nod
    - Iniţial, scorul acelui nod (dacă este frunză) sau -∞
    - Apoi,
      - Nod de tip MAX = cel mai mare scor al descendenţilor
      - Nod de tip MIN = valoarea α a predecesorului
  - Valoarea β a unui nod
    - Iniţial, scorul acelui nod (daca este frunză) sau +∞
    - Apoi,
      - Nod de tip MIN = cel mai mic scor al descendenţilor
      - Nod de tip MAX = valoarea β a predecesorului
  - Scorul unui nod
    - Nod de tip MAX valoarea α finală
    - Nod de tip MIN valoarea β finală

Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → MiniMax cu retezare α-β

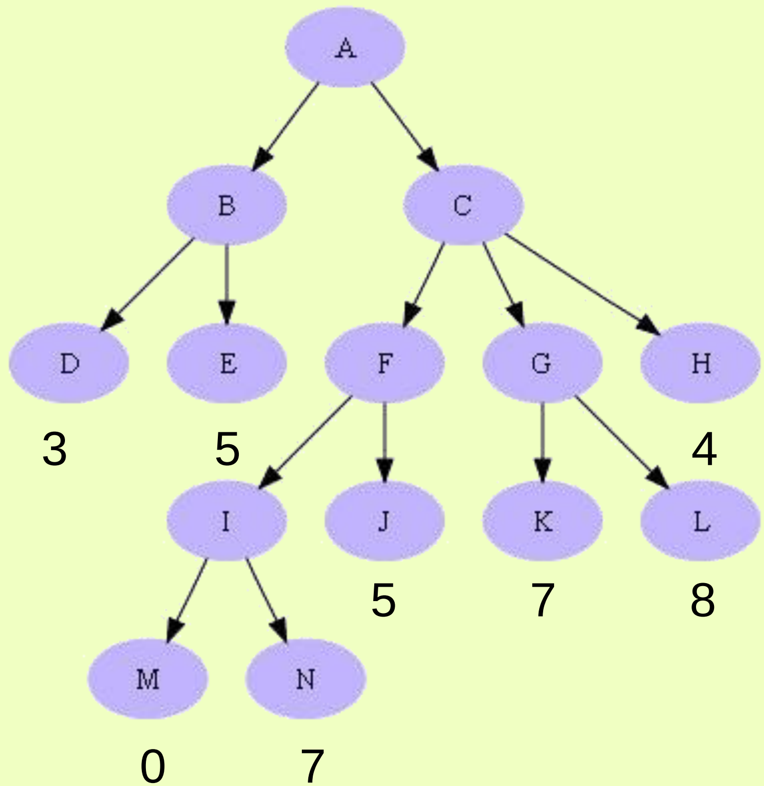
- Algoritm

```
int backMiniMaxAB(Node N, int A, int B){
    Set Alpha value of N to A and Beta value of N to B;
    if N is a leaf
            return the estimated score of this leaf
    else{
            if N is a Min node
                    for each child Ni of N {
                        Val = backMiniMaxAB(Ni, Alpha of N, Beta of N);
                        Beta value of N = minim(Beta value of N, Val);
                        if Beta value of N ≤ Alpha value of N then exit loop;
                    }
                    return Beta value of N;
            else // N is a Max node
                    for each child Ni of N do {
                        Val = MINIMAX-AB(Ni, Alpha of N, Beta of N);
                        Alpha value of N = Max{Alpha value of N, Val};
                        if Alpha value of N ≥ Beta value of N then exit loop;
                    }
                    Return Alpha value of N;
    }
}
```

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax cu retezare α-β

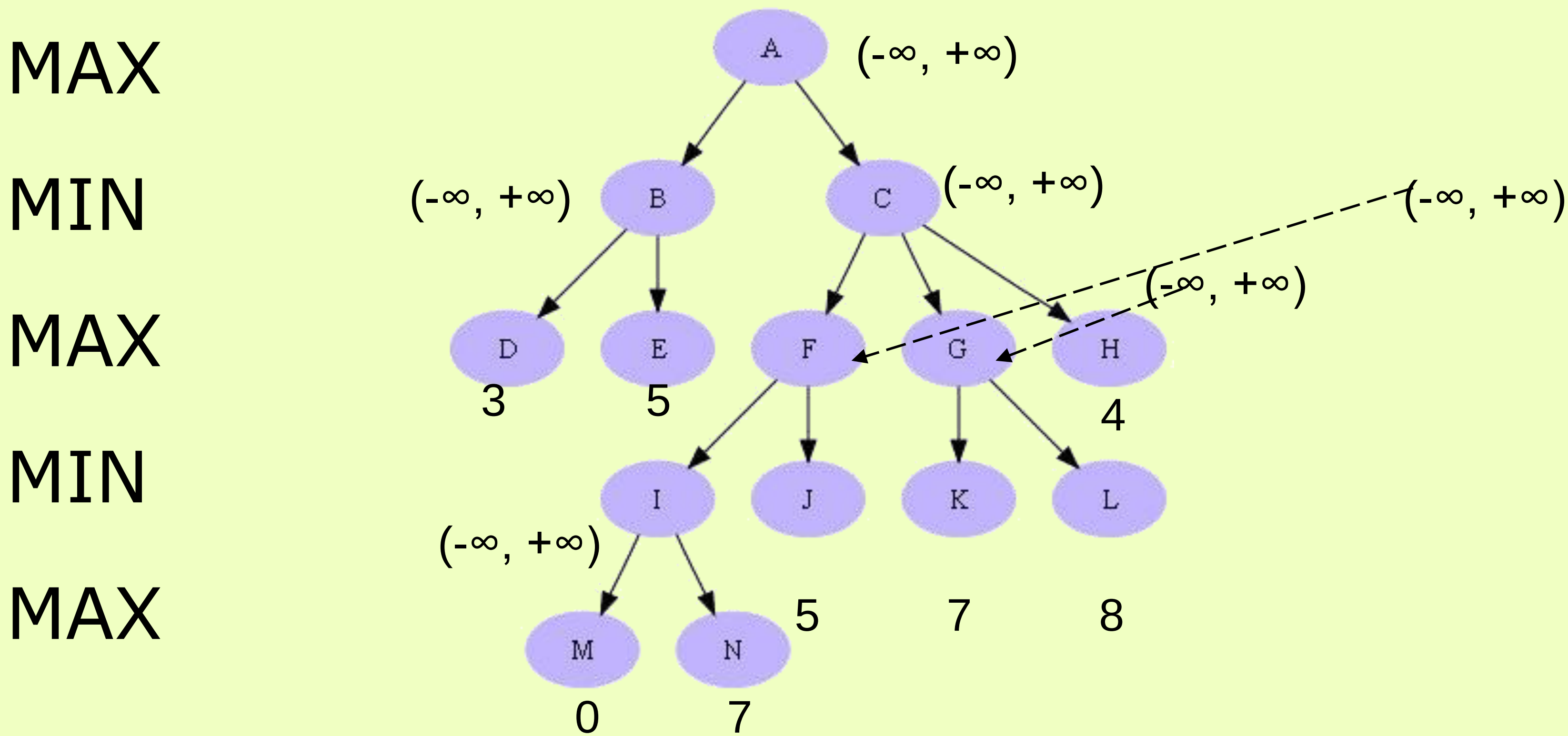
- Exemplu



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax cu retezare α-β

- Exemplu



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax cu retezare α-β

- Exemplu

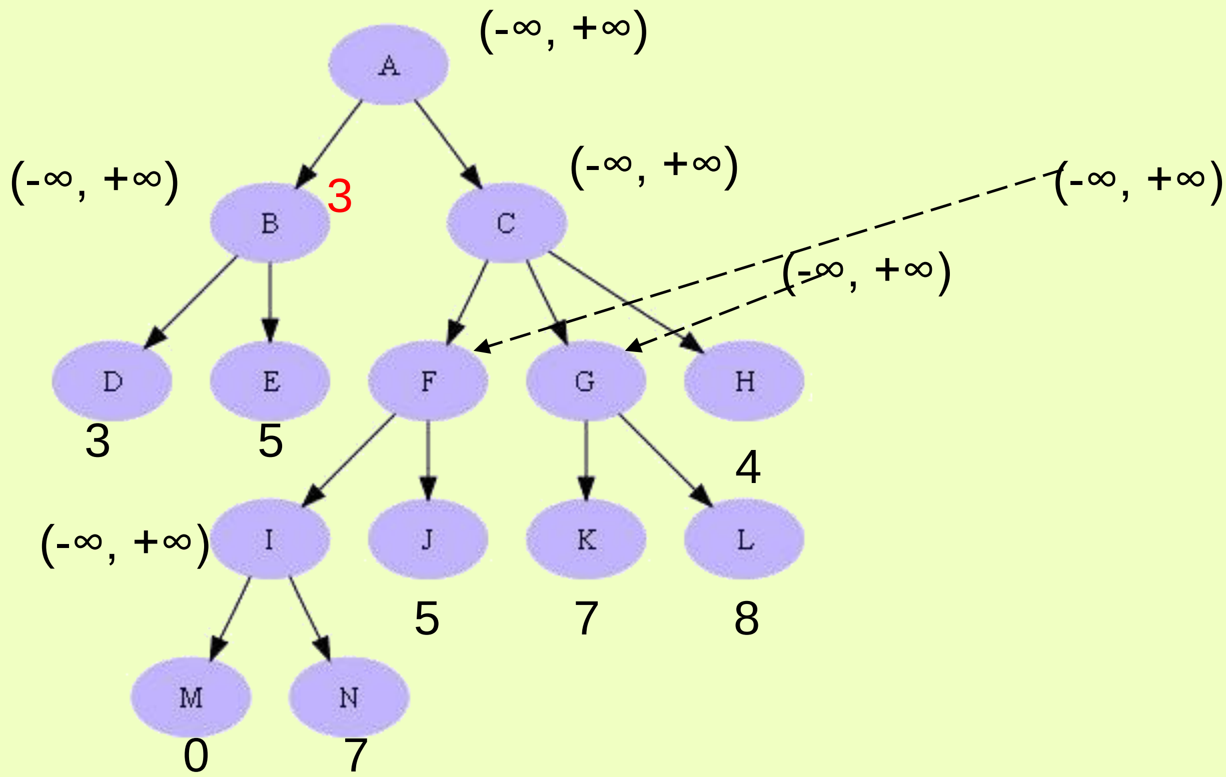
MAX(α)

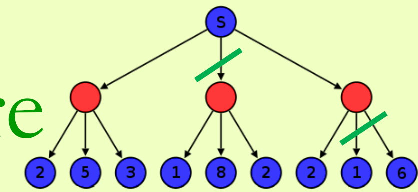
MIN(β)

MAX(α)

MIN(β)

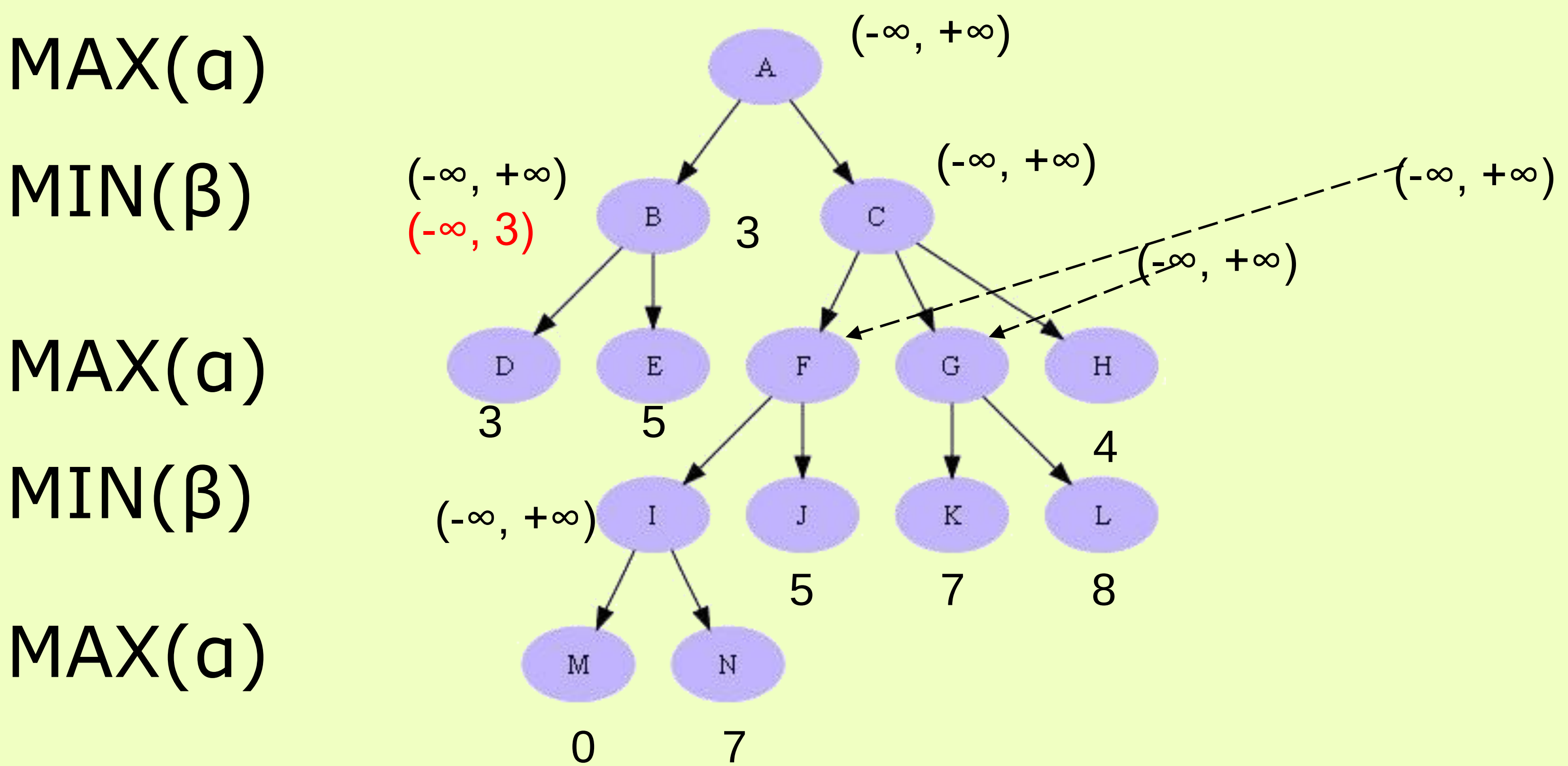
MAX(α)

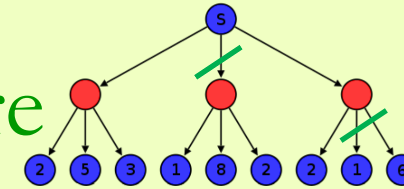




Strategii de joc → MiniMax cu retezare α-β

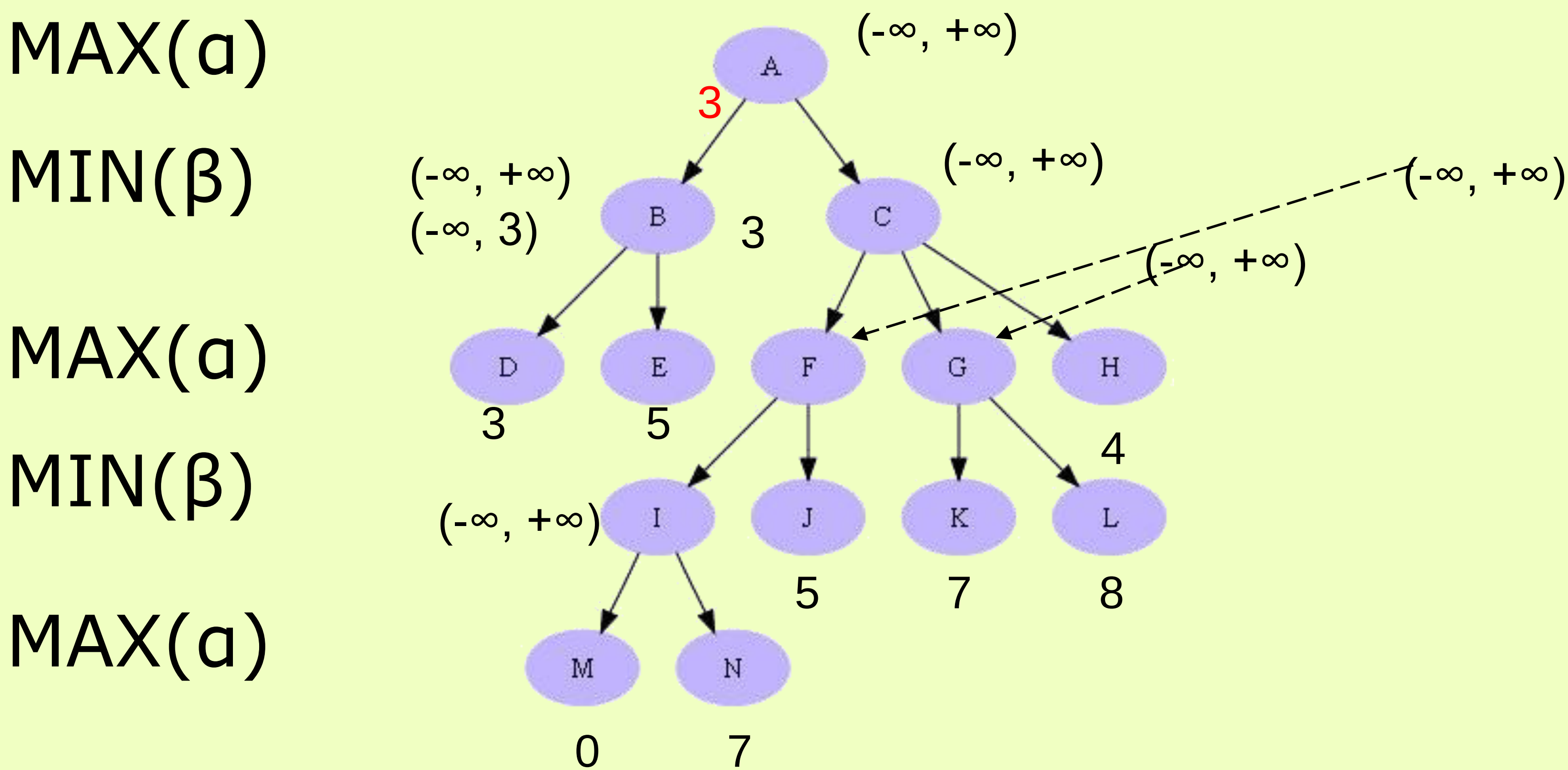
- Exemplu





Strategii de joc → MiniMax cu retezare α-β

- Exemplu



Strategii de joc → MiniMax cu retezare α-β

- Exemplu

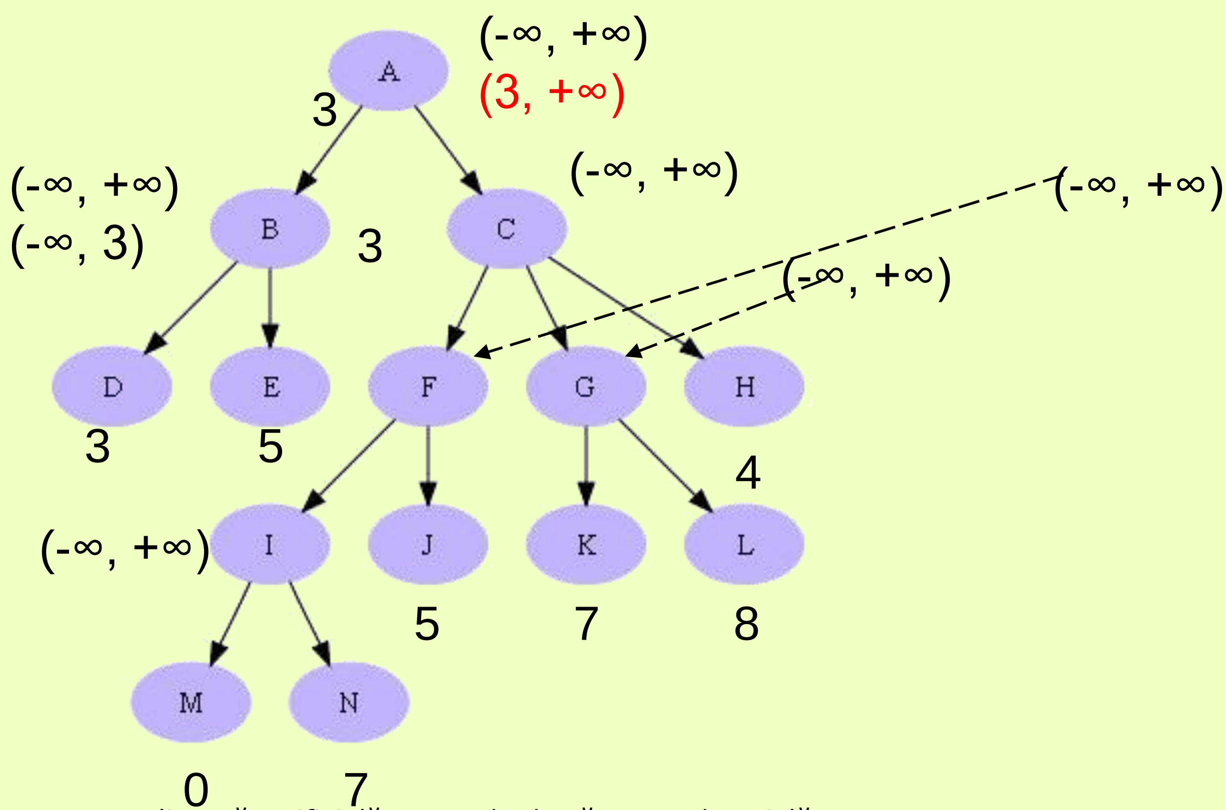
MAX(α)

MIN(β)

MAX(α)

MIN(β)

MAX(α)



Strategii de joc → MiniMax cu retezare α-β

- Exemplu

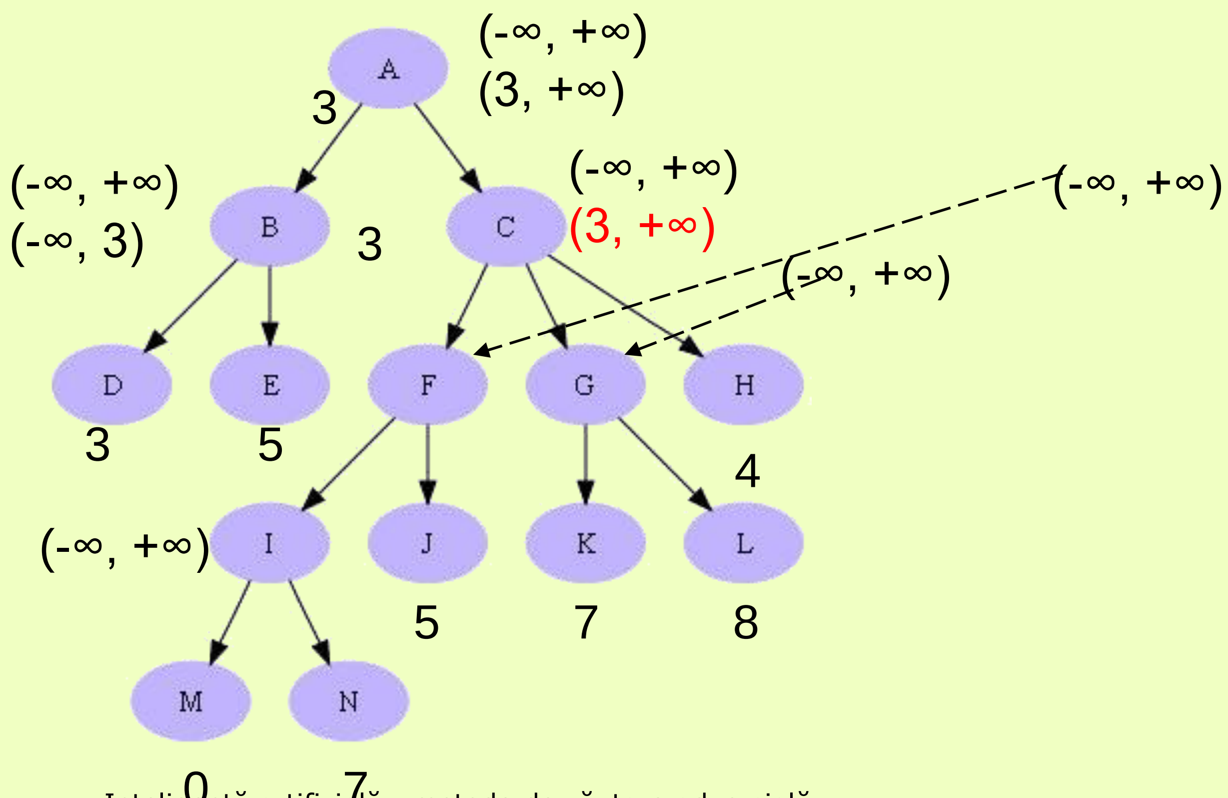
MAX(α)

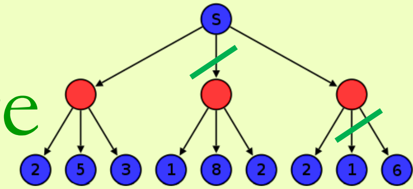
MIN(β)

MAX(α)

MIN(β)

MAX(α)





Strategii de joc → MiniMax cu retezare α-β

- Exemplu

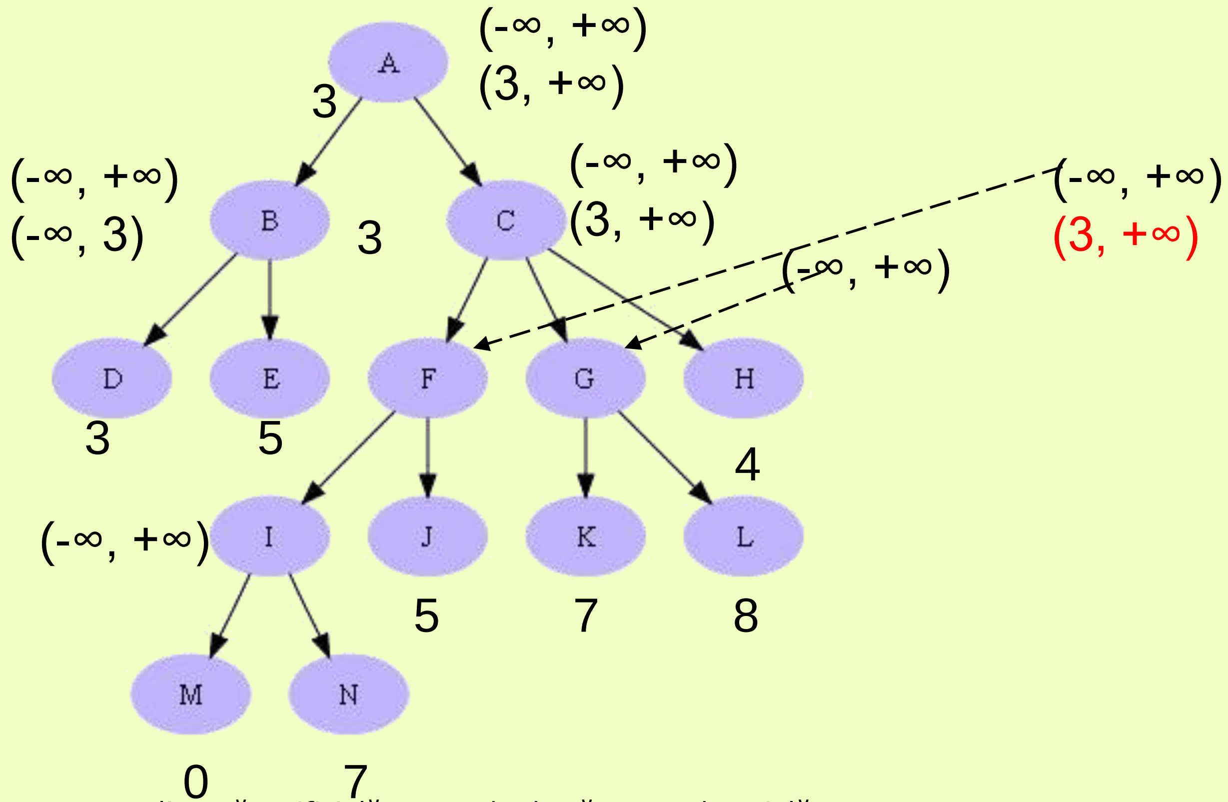
MAX(α)

MIN(β)

MAX(α)

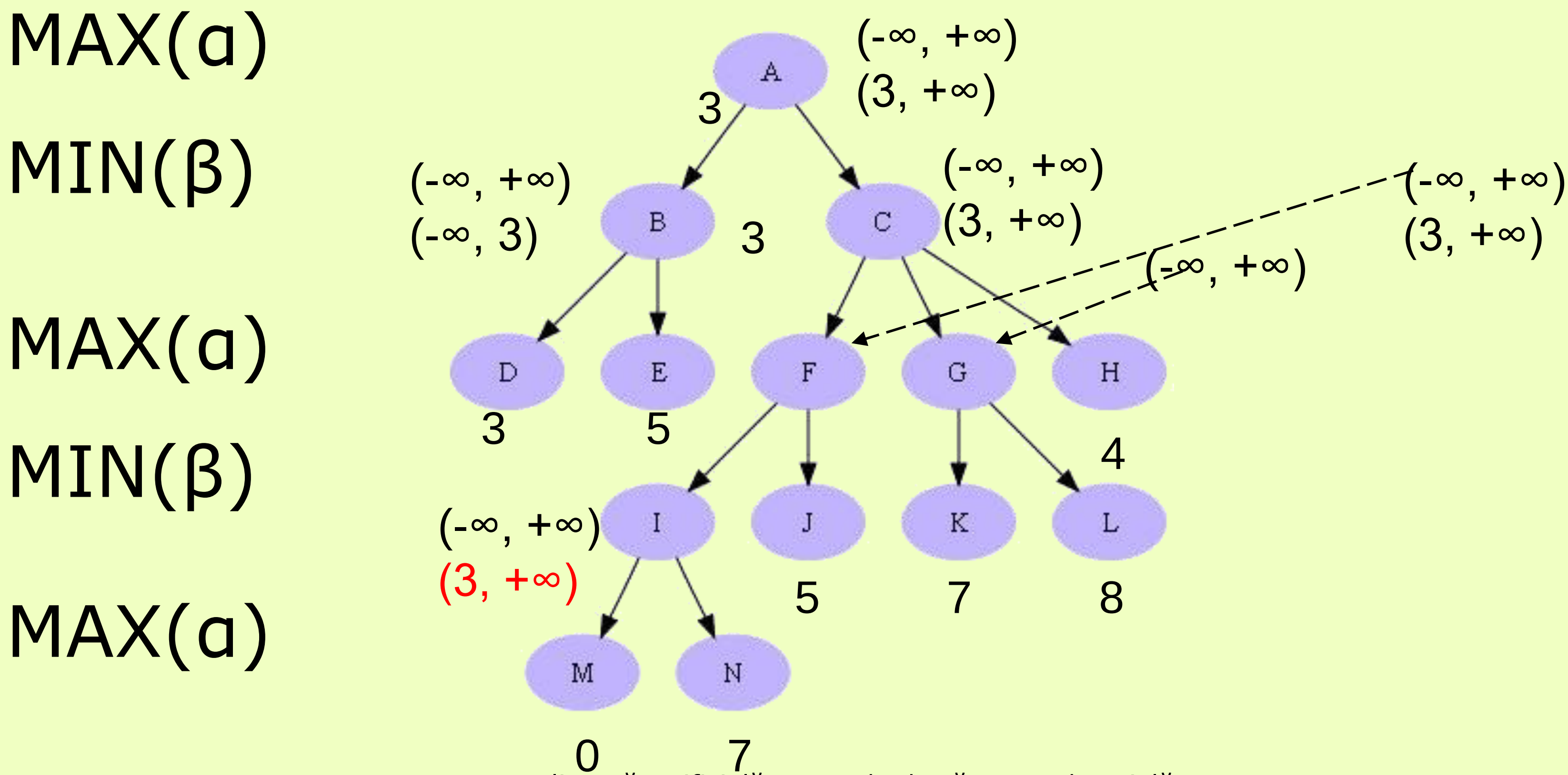
MIN(β)

MAX(α)



Strategii de joc → MiniMax cu retezare α-β

- Exemplu



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax cu retezare α-β

- Exemplu

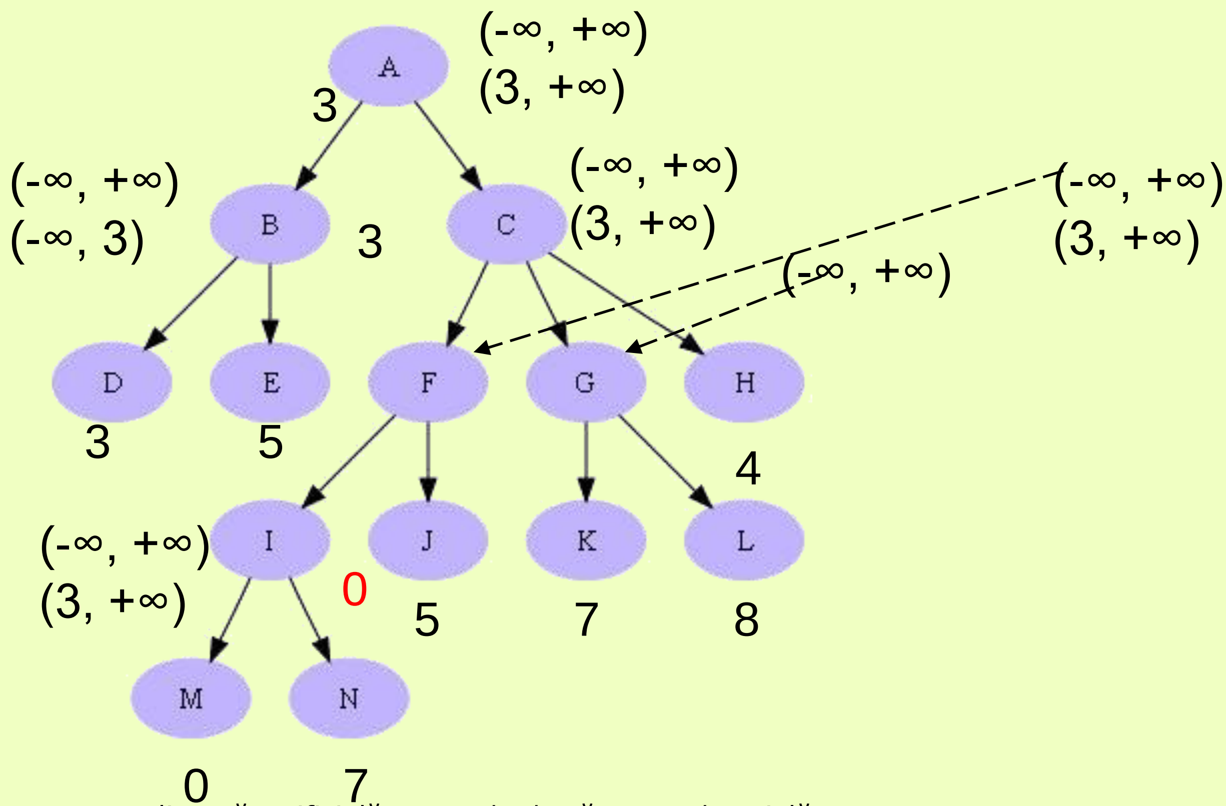
MAX(α)

MIN(β)

MAX(α)

MIN(β)

MAX(α)



Strategii de joc → MiniMax cu retezare α-β

- Exemplu

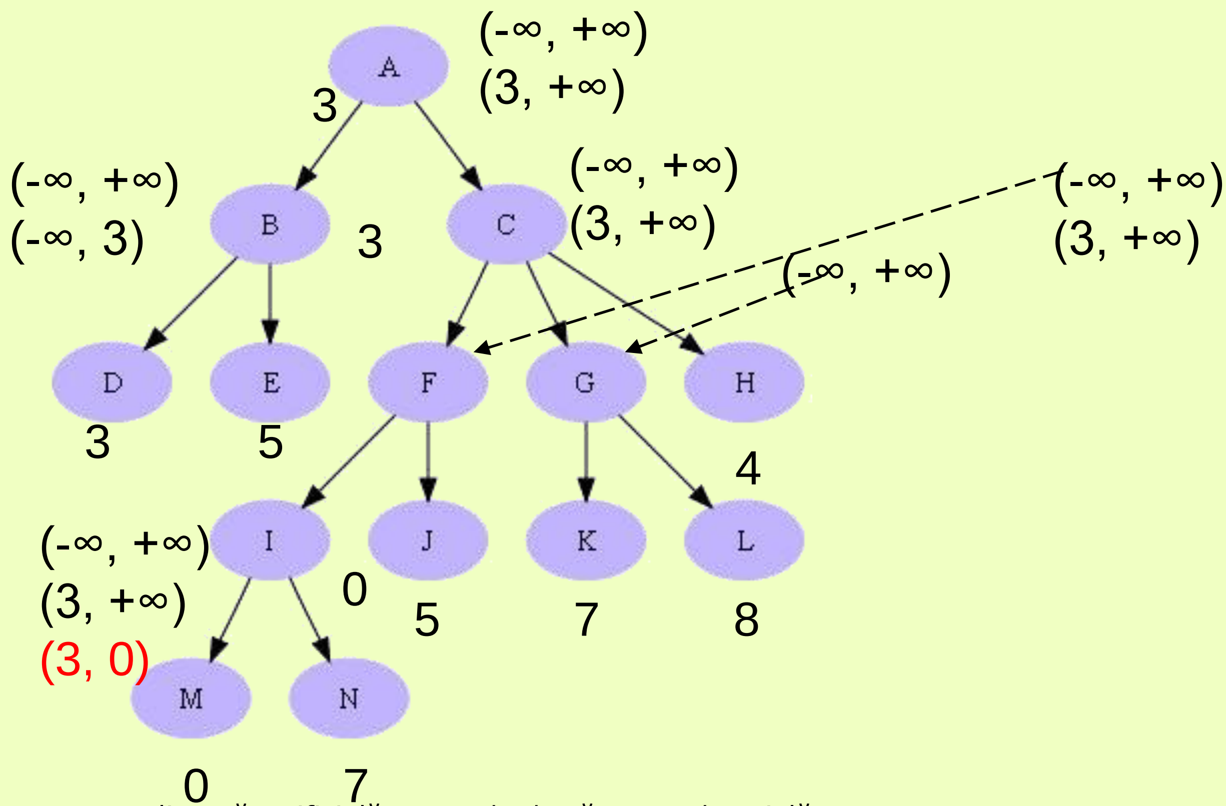
MAX(α)

MIN(β)

MAX(α)

MIN(β)

MAX(α)



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax cu retezare α-β

- Exemplu

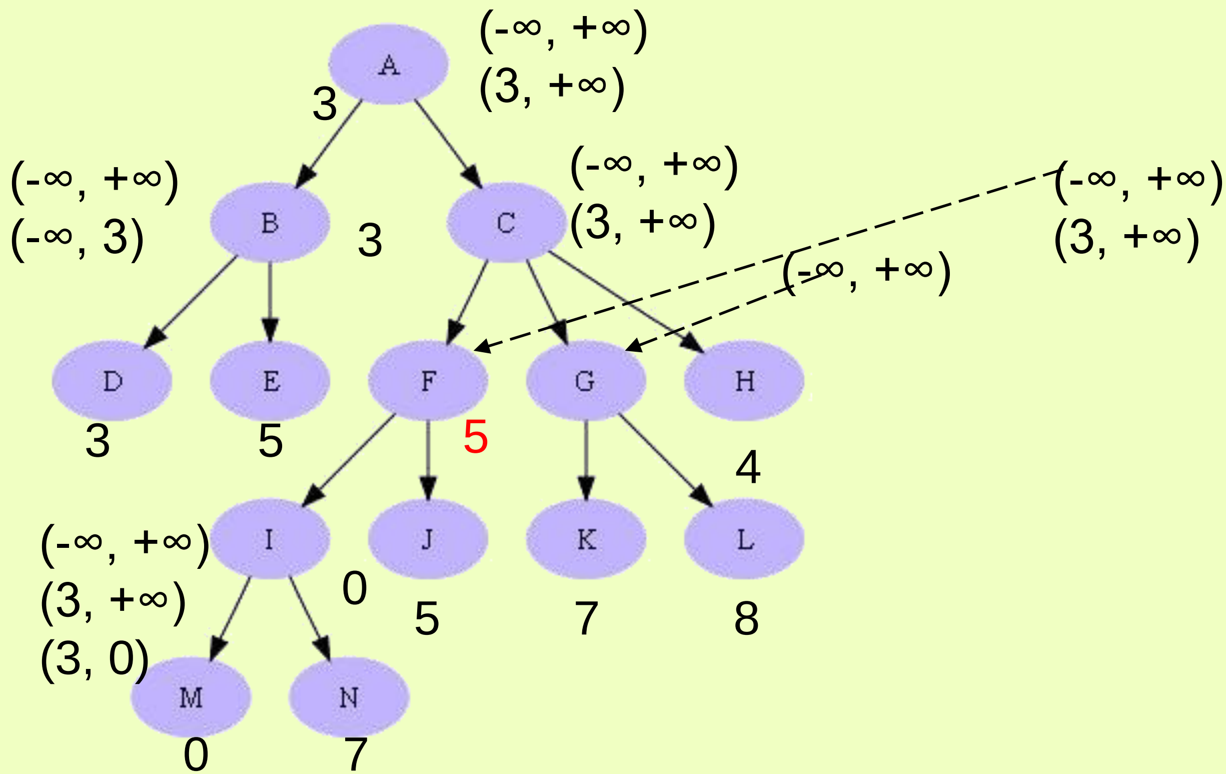
MAX(α)

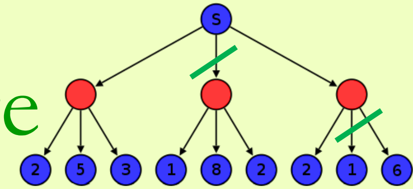
MIN(β)

MAX(α)

MIN(β)

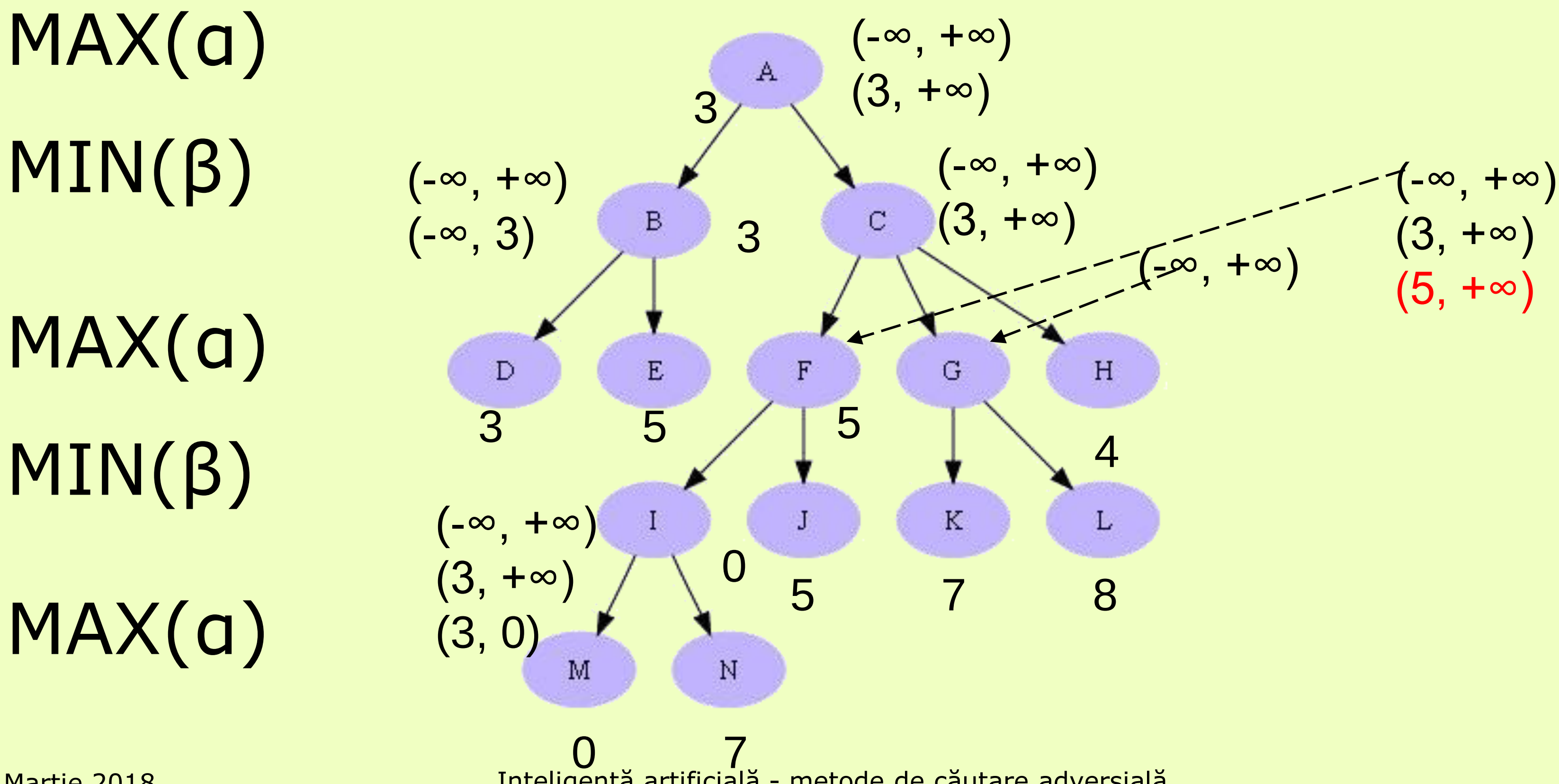
MAX(α)





Strategii de joc → MiniMax cu retezare α-β

- Exemplu



Strategii de joc → MiniMax cu retezare α-β

- Exemplu

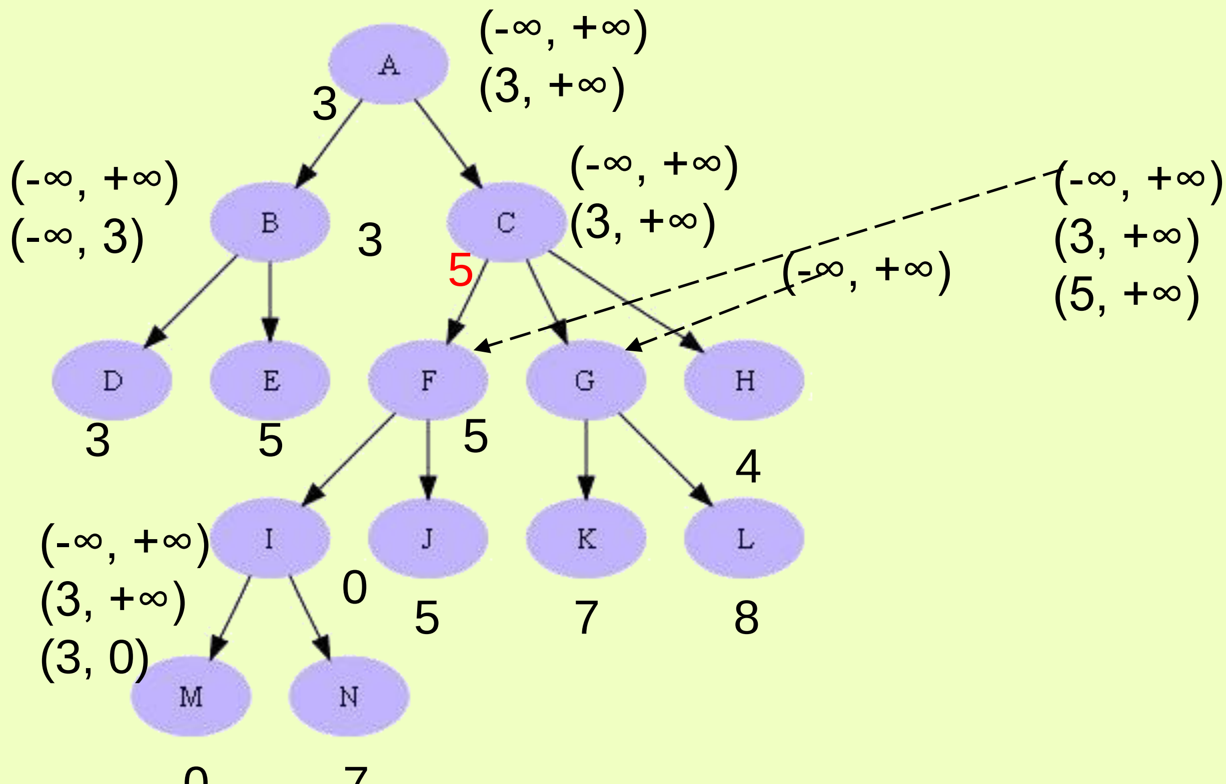
MAX(α)

MIN(β)

MAX(α)

MIN(β)

MAX(α)



Strategii de joc → MiniMax cu retezare α-β

- Exemplu

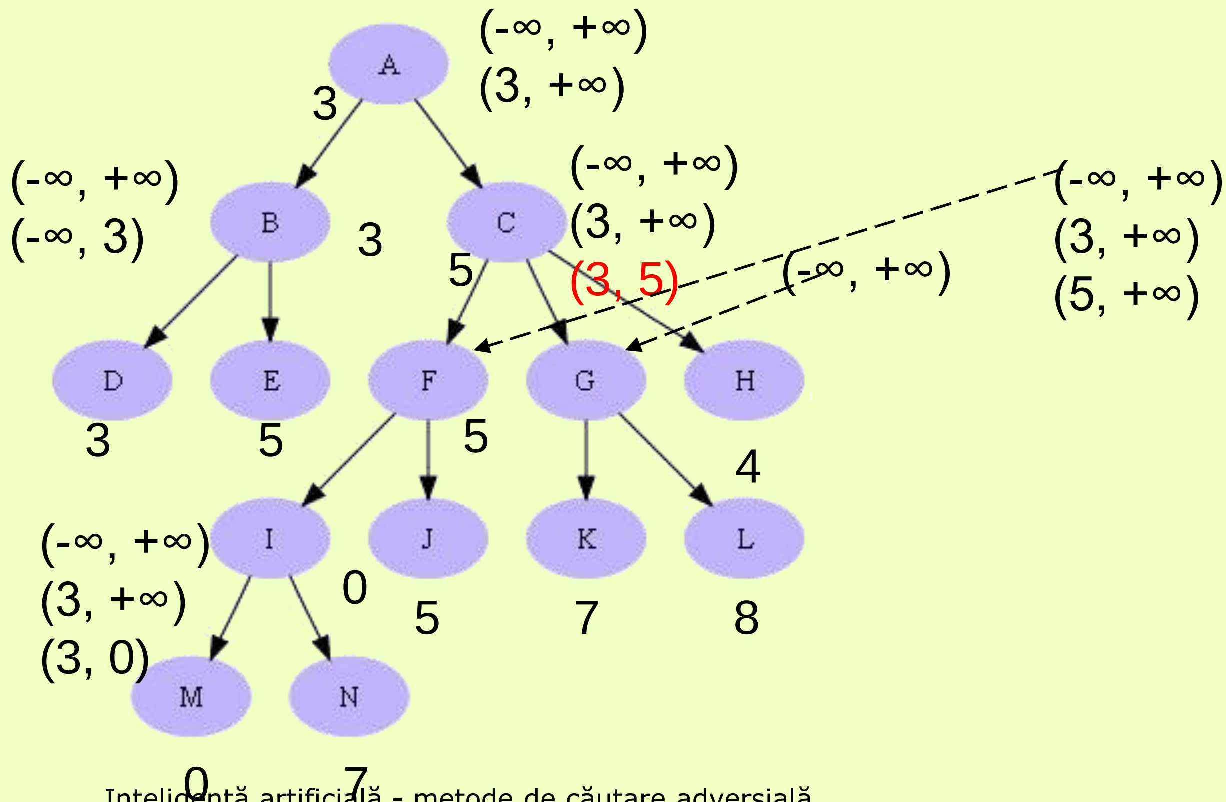
MAX(α)

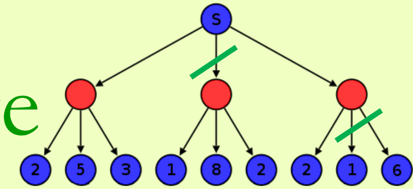
MIN(β)

MAX(α)

MIN(β)

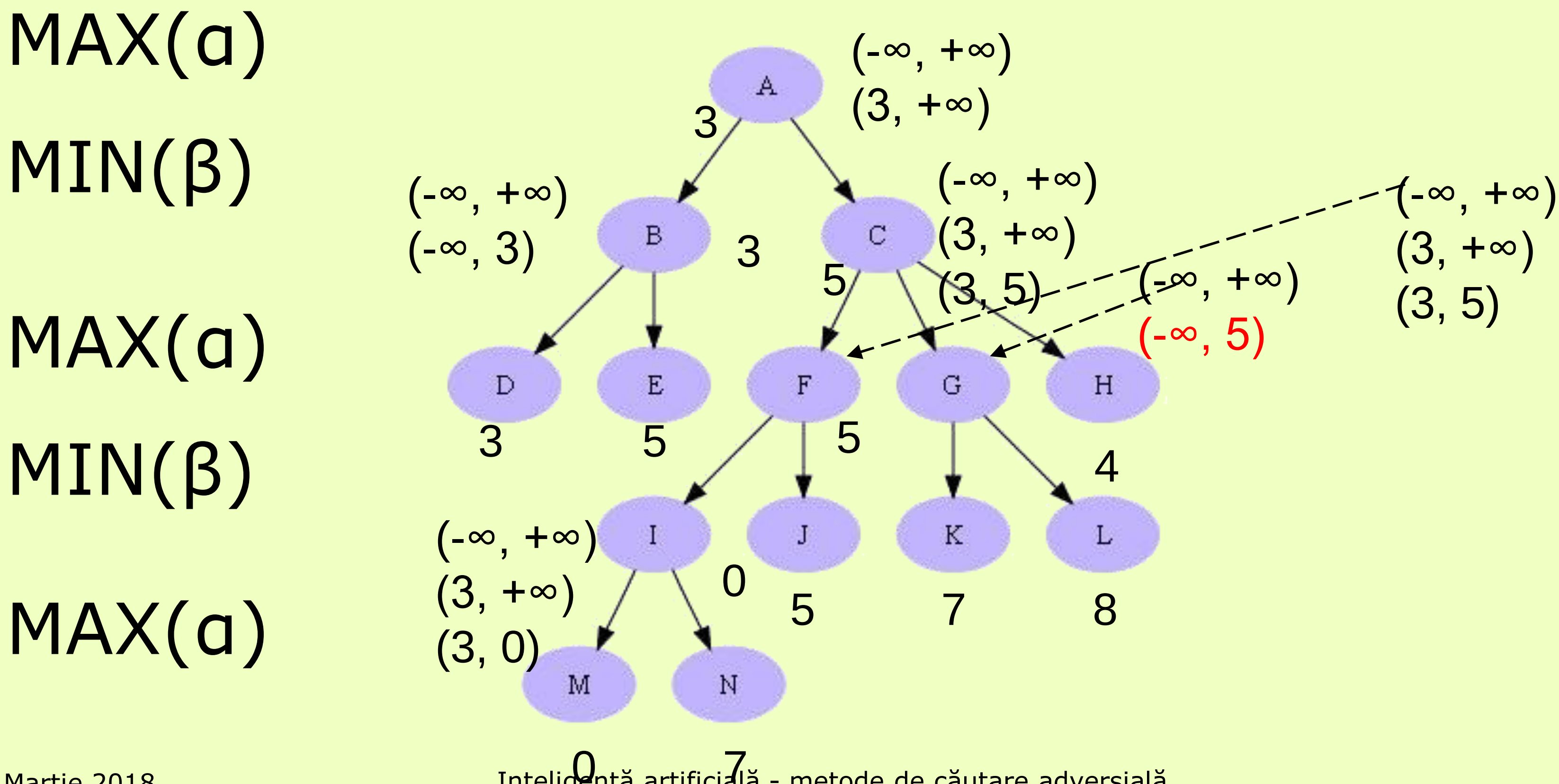
MAX(α)





Strategii de joc → MiniMax cu retezare α-β

- Exemplu



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → MiniMax cu retezare α-β

- Exemplu

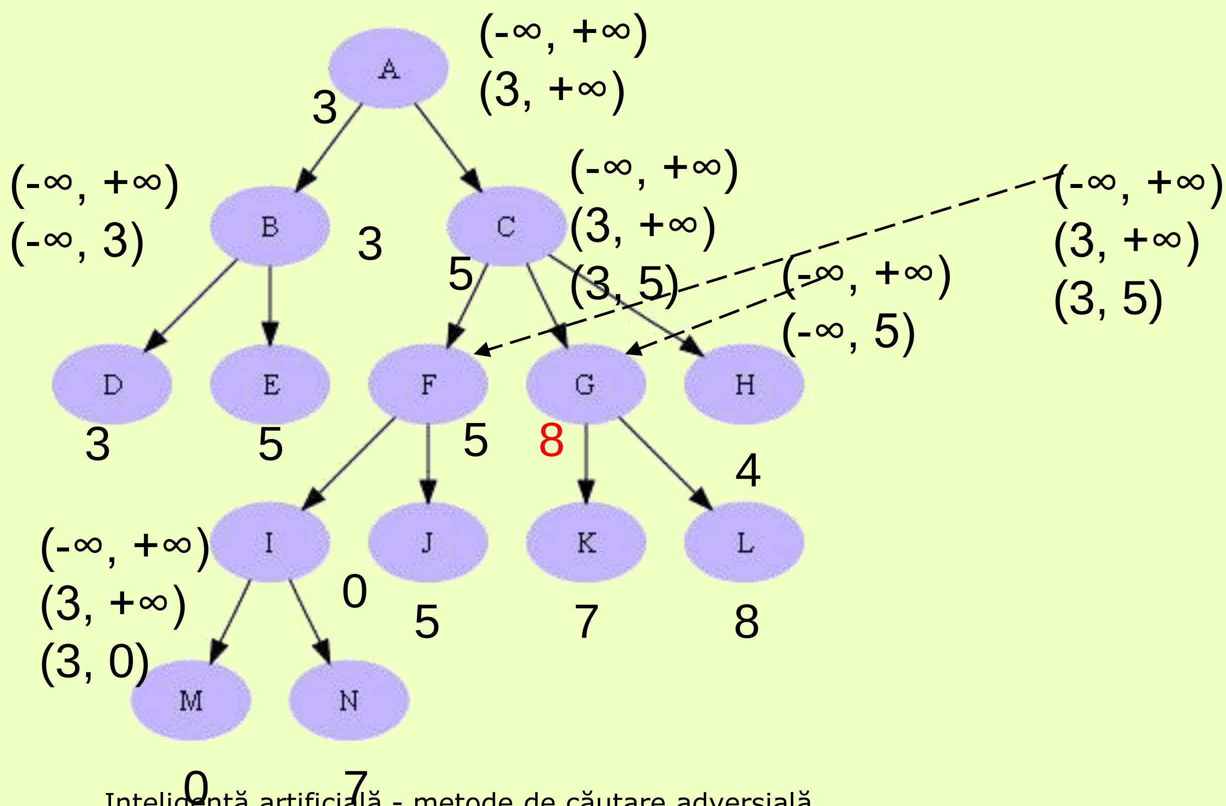
MAX(α)

MIN(β)

MAX(α)

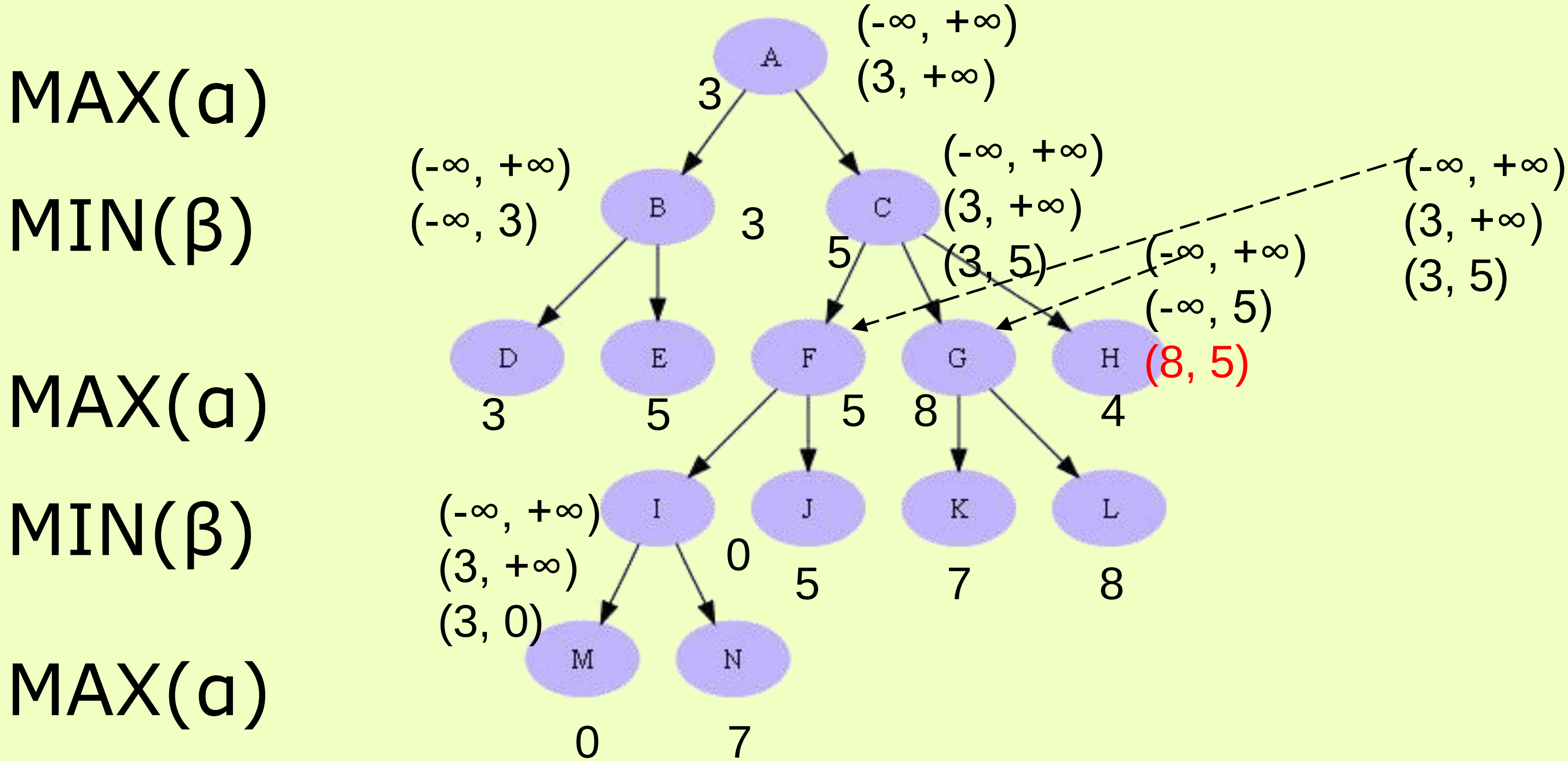
MIN(β)

MAX(α)

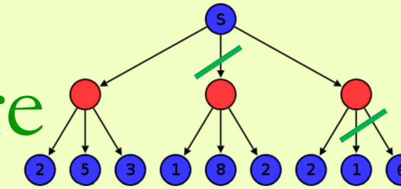


Strategii de joc → MiniMax cu retezare α-β

- Exemplu



Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → MiniMax cu retezare α-β

- Exemplu

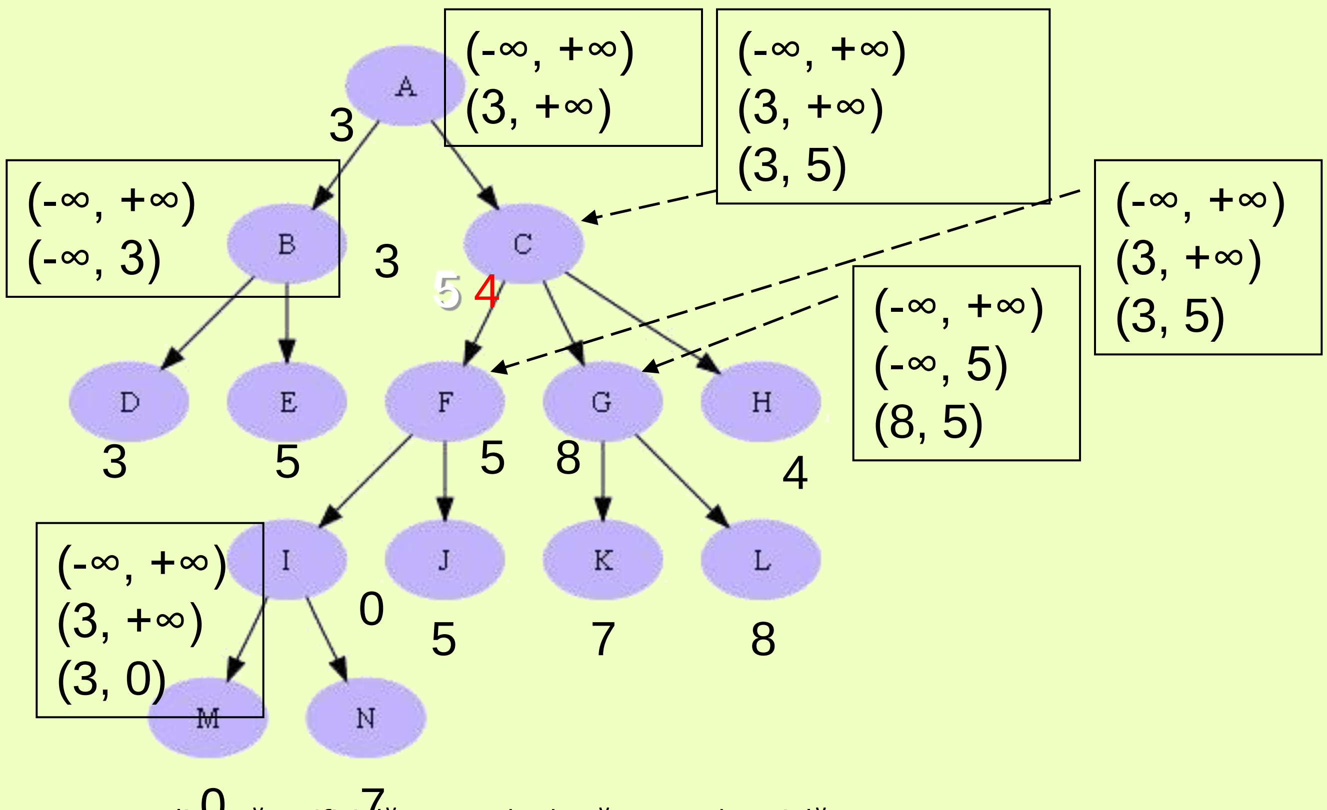
MAX(α)

MIN(β)

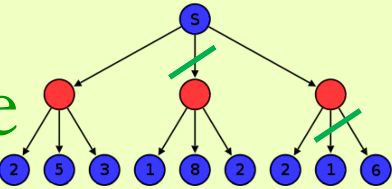
MAX(α)

MIN(β)

MAX(α)



Jocurile şi căutarea – spaţiul de căutare



Strategii de joc → MiniMax cu retezare α-β

- Exemplu

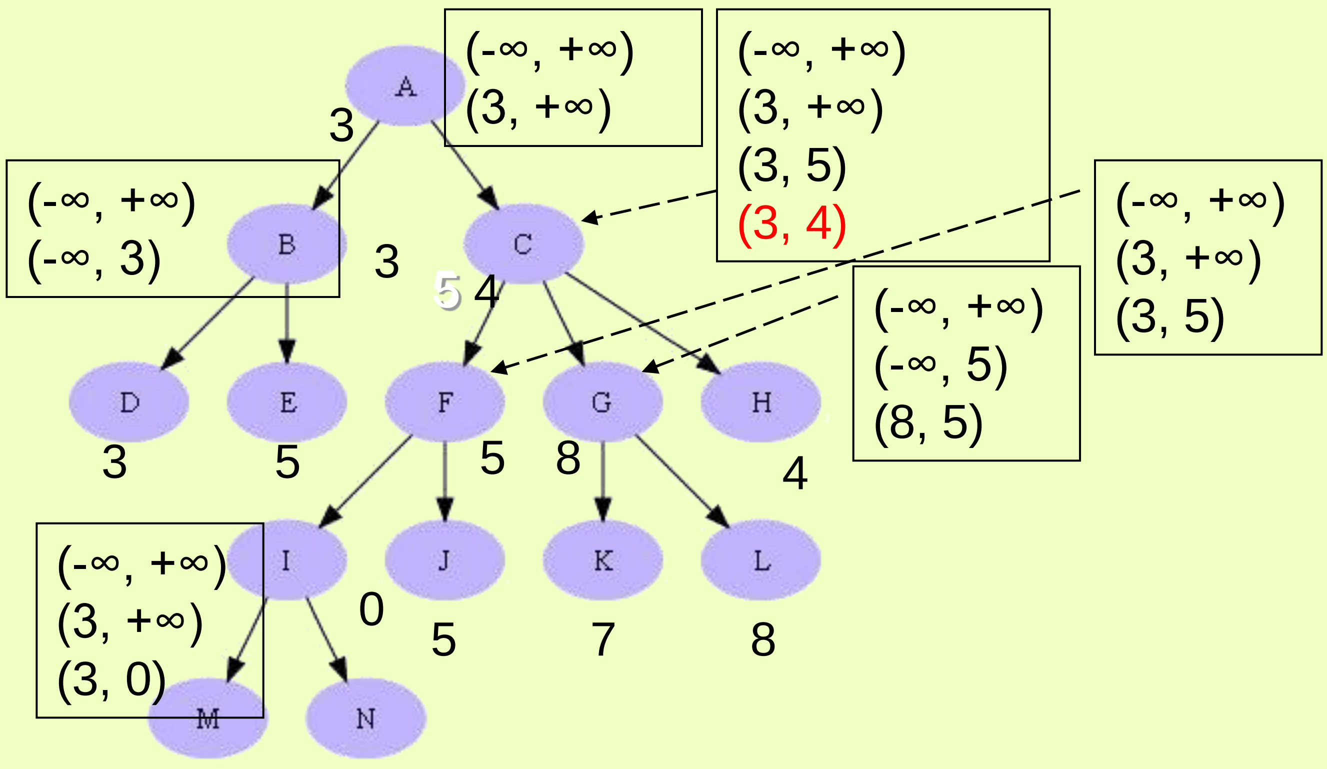
MAX(α)

MIN(β)

MAX(α)

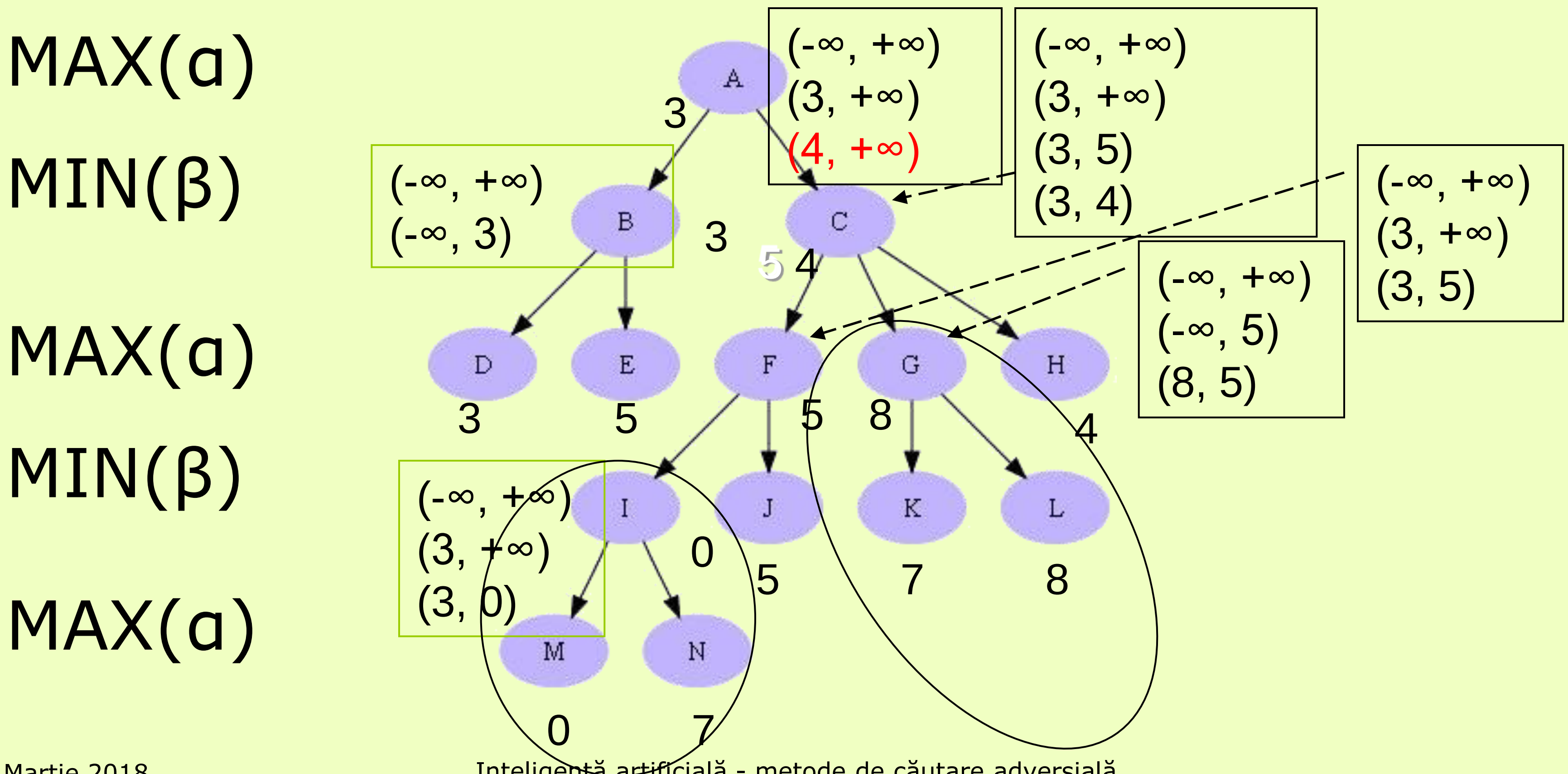
MIN(β)

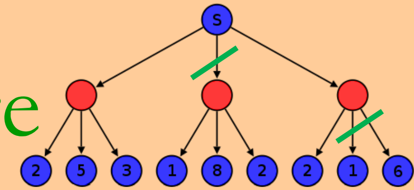
MAX(α)



Strategii de joc → MiniMax cu retezare α-β

- Exemplu

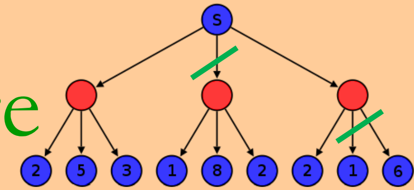




Strategii de joc → MiniMax cu retezare α-β

- Proprietăţi
  - Reducerea nu afectează rezultatul final
  - O bună ordonare a mutărilor îmbunătăţeşte algoritmul de reducere
  - Dacă succesorii sunt puşi perfect în ordine (cei mai buni se află primii), atunci complexitatea temporara ar fi = $O(b^{d/2})$, în loc de $O(b^d)$ cat are minimax
  - Se poate transforma un program de la nivelul începător la nivelul expert

Jocurile şi căutarea – spaţiul de căutare

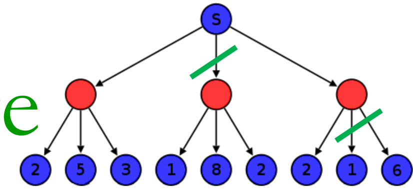


Strategii de joc → MiniMax cu retezare α-β

- MinMax *vs.* MinMax cu retezare α-β

	MinMax	**MinMax α-β**
Complexitate temporală	$O(b^m)$	$O(b^{m/2})$ cu ordonare perfectă a nodurilor
Complexitate spaţială	$O(b^m)$	$O(2b^{d/2})$ – cel mai bun caz (când unui nod Max îi este generat ca prim copil cel cu valoarea cea mai mare şi când lui Min îi este generat ca prim copil cel cu valoarea cea mai mică) $O(b^d)$ – cel mai rău caz (fără retezare)
Completitudine	Da	Da
Optimalitate	Da	Da

Jocurile şi căutarea – spaţiul de căutare



- Arbori AndOR, MiniMax, MiniMax cu retezare α-β
  - Complexitatea mărită → necesitatea unor tehnici eficiente de căutare în rezolvarea jocurilor
    - Jocul de şah
      - $b \sim 35$
      - $d \sim 100$
      - $b^d \sim 35^{100} \sim 10^{154}$ noduri
    - Jocul Tic-Tac-Toe
      - ~ 5 mutări legale dintr-un total de 9 mutări
      - $5^9 = 1\ 953\ 125$
      - $9! = 362\ 880$ (dacă computerul mută primul)
      - $8! = 40\ 320$ (dacă computerul mută al doilea)
    - Jocul Go
      - $b > 361$ (pentru o tablă de 19x19)

Jocurile şi căutarea – spaţiul de căutare

- În practică
  - Jocul de dame → Chinok
    - Chinok îl invinge pe Marion Tinsley în 1994 (după 40 de ani de confruntări)
    - Se foloseşte o bază cu finaluri de joc calculate pentru toate poziţiile unui joc perfect jucat cu cel mult 8 piese (444 bilioane de poziţii posibile)
  - Jocul de şah → Deep Blue
    - Deep Blue l-a învins pe Garry Kasparov în 1997
    - Se verifică 200 mil poziţii/sec
    - Factorul de ramificare se reduce la 6 cu retezarea α-β (faţă de 35-40)
  - Jocul Othello
    - Jucătorii umani refuză să joace cu computerele (pt că sunt prea bune) – Moor (1980), Logistello (1997)
  - Jocul Go
    - Jucătorii umani refuză să joace cu computerele (pt că sunt prea slabe)
    - Factorul de ramificare > 300 ➔ necesitatea unor algoritmi bazaţi pe pattern-uri
  - Jocul de table
    - BKG (logica fuzzy), TD-Gammon (reţele neuronale artificiale)
    - Computerul l-a învins pe campionul lumii, dar pt că a fost norocos

Jocurile şi căutarea – spaţiul de căutare

- Strategia de joc
  - Pas cu pas
    - Ex.: XO, Dame, Şah
    - Algoritmi – pot lucra cu structuri:
      - Liniare
        - Strategia simetriei
        - Strategia perechilor
        - Strategia parităţii
        - Programare dinamică
        - Alte strategii
      - Arborescente
        - Arbori AndOr
        - MiniMax (cu tăieturi Alpha-Beta)
  - Completă
    - Ex.: ieşirea dintr-un labirint, deplasarea pe o hartă între 2 locaţii date
    - Algoritmi pot să identifice
      - **Un drum optim de la o locaţie la alta (pathfinding)**
      - O succesiune de acţiuni care să deplaseze jucătorul de la o locaţie la alta

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Pathfinding

- Aspecte teoretice
  - Problema
    - Identificarea pe o hartă (posibil cu obstacole) a unui drum optim de la o locaţie la alta
    - Unde?
      - pe o hartă
        - harta ca o matrice
        - harta ca un graf (simplu, *mesh*, *quad-tree*)
      - într-un mediu (cunoscut/necunoscut, static/dinamic)
    - De către cine?
      - un agent
      - 2 agenţi
      - mai mulţi agenţi
  - Soluţia
    - Utilizarea unei metode de căutare (optimizare)

Jocurile şi căutarea – spaţiul de căutare

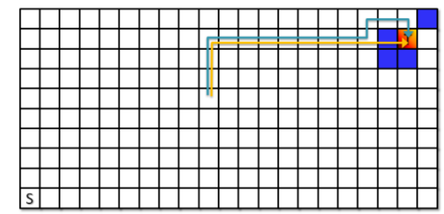
Strategii de joc → Pathfinding

- Aspecte teoretice
  - Soluţia
    - Cum?
      - Utilizarea unei metode de căutare (optimizare)
    - Care soluţie este cea mai bună?
      - viteza de calcul,
      - lungimea drumului,
      - calitatea drumului,
      - informaţia disponibilă
    - Dificultăţi
      - soluţie oferită în timp real
      - resurse limitate
      - spaţiu de căutare imens
      - modelarea situaţiilor reale implică incertitudini şi elemente (teren, unităţi, agenţi) eterogene

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Pathfinding

- Algoritmi de căutare pentru hărţi reprezentate ca "matrici de dale" → labirinturi
  - De ce?
    - Simplitate
  - Caracteristici
    - Spaţiu de căutare discret
    - Dalele sunt pătrate
    - Dalelele pot fi traversabile sau blocate
    - Mişcări în linie dreaptă de pe o dală pe alta (orizontală, verticală, diagonală)
  - Metode
    - A*
      - Algoritmi de tip *Best-first search*
      - Reprezintă un standard în industria jocurilor
    - Euristici → Manhattan (gen. Minkowski)
      - Estimarea distanţei de la punctul curent la destinaţie
      - Se ignoră obstacolele



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Pathfinding

- Algoritmi de căutare pentru hărţi reprezentate ca un graf
  - De ce?
    - Reprezentarea matriceală → zone mari din hartă care au acelaşi cost
  - Caracteristici
    - Spaţiu de căutare discret
    - “Dalele” pot avea orice formă
    - Dalelele pot fi traversabile sau blocate
    - Mişcări oarecare de pe o dală pe alta
  - Metode
    - Deplasare pe muchii
    - Deplasare pe noduri
      - Puncte
      - Marginile poligonului
      - Mijlocul poligonului
    - Ex. Dijkstra, Floyd-Warshall, Kruskal, etc

Jocurile şi căutarea – spaţiul de căutare

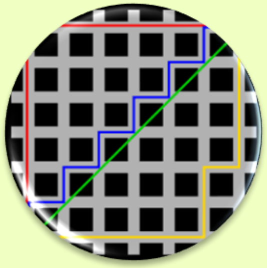
Strategii de joc → Pathfinding

- Îmbunătăţiri ale algoritmilor de căutare
  - Euristici mai bune
  - Abstractizări ierarhice
  - Abordări descentralizate

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Pathfinding

- Îmbunătăţiri ale algoritmilor de căutare
  - Euristici mai bune
    - Euristici fără memorie
      - Ex. Manhattan
      - Foarte rapid de calculat
      - Nu necesită memorie suplimentară
      - Calitate bună uneori
    - Euristici bazate pe memorie
      - Ex. euristici bazate pe repere
      - Destul de rapide de calculat
      - Necesită memorie
      - Calitate bună (identificarea drumurilor moarte)
    - Informaţie perfectă
      - Foarte costisitoare la calculare şi stocare
      - Foarte rapidă la utilizat (după ce au fost calculate)
      - Ne-practicabile



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Pathfinding

- Îmbunătăţiri ale algoritmilor de căutare
  - Abstractizări ierarhice
    - Grafuri în care
      - sunt adnotate şi fluxurile de mişcare (deplasări în ambele sensuri)
      - sunt asociate relaţii de dominanţă între muchii
      - Sunt identificate nivele (ex. cameră, casă, oraş, ţară)
  - Abordări descentralizate
    - Ideea de bază:
      - Descompunerea problemei în sub-probleme care
        - pot fi rezolvate repede
        - pot fi sub-optimale
        - pot fi incomplete
    - Scop
      - toate elementele (identificare colaborativă) să ajungă la destinaţie
    - Dificultate
      - se măreşte spaţiul de căutare: $O(n) \rightarrow O(n^m)$
      - factorul de ramificare explodează (de la 4 sau 8 la 5^m sau 9^m)

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Planning

- Aspecte teoretice
  - Problema
    - Planificarea (ordonarea) unor acţiuni
      - mutări, deplasări, alegeri, răsuciri, etc
    - Se dau
      - O stare iniţială

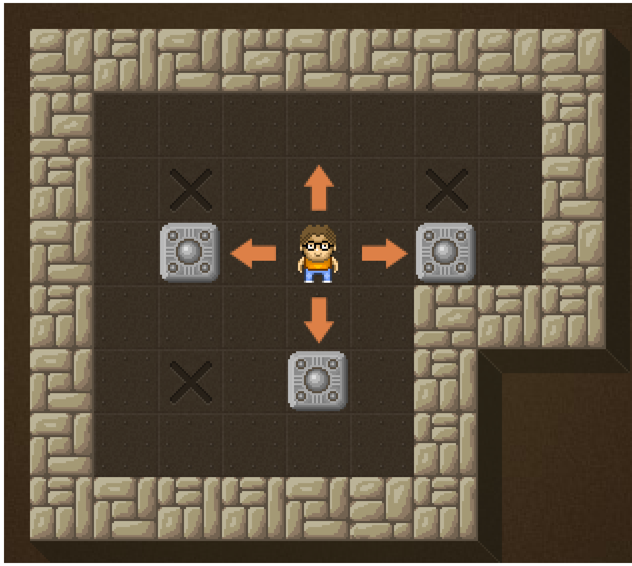
Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Planning

- Aspecte teoretice
  - Problema
    - Planificarea (ordonarea) unor acţiuni
      - mutări, deplasări, alegeri, răsuciri, etc
    - Se dau
      - O stare iniţială
      - O stare obiectiv (finală)

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Planning



- Aspecte teoretice
  - Problema
    - Planificarea (ordonarea) unor acţiuni
      - mutări, deplasări, alegeri, răsuciri, etc
    - Se dau
      - O stare iniţială
      - O stare obiectiv (finală)
      - Un set de acţiuni posibile

Jocurile şi căutarea – spaţiul de căutare

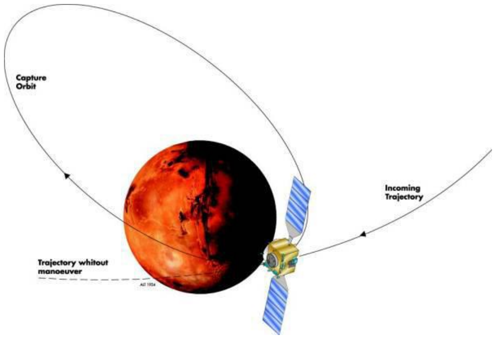
Strategii de joc → Planning

- Aspecte teoretice
  - Problema
    - Planificarea (ordonarea) unor acţiuni
      - mutări, deplasări, alegeri, răsuciri, etc
    - Se dau
      - O stare iniţială
      - O stare obiectiv (finală)
      - Un set de acţiuni posibile
    - Se cere
      - Să se identifice o secvenţă de acţiuni care transformă starea iniţială în starea finală

Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Planning

- De ce?
  - jocuri cu NPC *(First-person shooter, Role-playing, Real-time strategy)*
  - Probleme reale de planificare
    - Telescop în spaţiu
    - Roboţei
    - UAV-uri



Jocurile şi căutarea – spaţiul de căutare

Strategii de joc → Planning

- Cum?
  - Algoritmi
    - Pentru alegerea unei acţiuni (shooting)
    - Pentru evaluarea mediului
  - Abordări
    - STRIPS → http://www.dis.uniroma1.it/~degiacom/didattica/dottorato-stavros-vassos/
    - HTN
  - Simularea comportamentului
    - Maşini cu stări finite (FSM)
    - Arbori de comportament (Halo 2)
    - GOAP (FEAR)

Recapitulare

- Definirea unui joc
  - Starea iniţială (cum sunt aşezate iniţial elementele în joc)
  - Acţiunile posibile (care sunt mutările permise)
  - Test terminal (care indică terminarea jocului)
  - Funcţie utilitate (care spune cine şi cât a câştigat)

- Strategii de rezolvare a jocurilor
  - Bazate pe explorarea (aproape) completă a arborelui de joc
    - AndOR ➔ cine câştigă
    - MiniMax ➔ cine şi cât câştigă
    - MiniMax cu retezare α-β ➔ cine şi cât câştigă şi ce mutări nu merită să fie efectuate
  - Bazate pe strategii inteligente
    - Strategia simetriei
    - Strategia perechilor
    - Strategia parităţii
    - Programare dinamică
    - A* (Pathfinding, Planning)

Cursul următor

A. Scurtă introducere în Inteligenţa Artificială (IA)

B. Rezolvarea problemelor prin căutare
- Definirea problemelor de căutare
- Strategii de căutare
  - Strategii de căutare neinformate
  - Strategii de căutare informate
  - Strategii de căutare locale (Hill Climbing, Simulated Annealing, Tabu Search, Algoritmi evolutivi, PSO, ACO)
  - Strategii de căutare adversială

C. Sisteme inteligente
- Sisteme care învaţă singure
  - Arbori de decizie
  - Reţele neuronale artificiale
  - Maşini cu suport vectorial
  - Algoritmi evolutivi
- Sisteme bazate pe reguli
- Sisteme hibride

Cursul următor – Materiale de citit şi legături utile

- capitolul III din *S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995*
- capitolul 4 şi 5 din *H.F. Pop, G. Şerban, Inteligenţă artificială, Cluj Napoca, 2004*
- capitolul 2 din *Adrian A. Hopgood, Intelligent Systems for Engineers and Scientists, CRC Press, 2001*
- capitolul 6 şi 7 din *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*

- Informaţiile prezentate au fost colectate din diferite surse de pe internet, precum şi din cursurile de inteligenţă artificială ţinute în anii anteriori de către:
  - Conf. Dr. Mihai Oltean – www.cs.ubbcluj.ro/~moltean
  - Lect. Dr. Crina Groşan - www.cs.ubbcluj.ro/~cgrosan
  - Prof. Dr. Horia F. Pop - www.cs.ubbcluj.ro/~hfpop