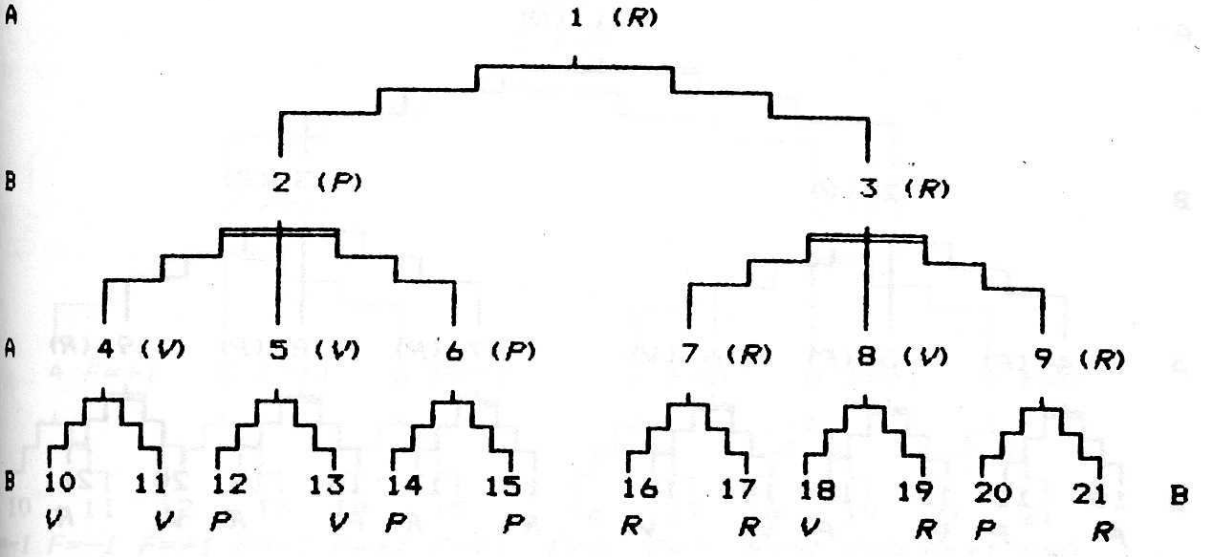
Algoritmus Nilssonovho typu tu neuspeje z dvoch dôvodov:

1. Nemá mechanizmus, ktorý by objavil, že "Turing je človek." a "Sokrates je Grék." nemôže byť riešením.
2. Ak by aj taký mechanizmus existoval, algoritmus nemá prostriedky pre zmenu riešenia, ktoré sa už našlo. Ak "Turing je človek." je prvý triviálny problém, ktorý sa našiel, potom "Nájdi niečo, čo je človek." a "Nájdi niečo, čo je omylné." sa označia ako vyriešené a preto "Sokrates je človek." sa odstráni zo zoznamu OPEN a "Nájdi niečo, čo je Grék." sa tým pádom, použitím predchádzajúcej hodnoty "niečo" (tá je naviazaná na "Turing") stane neriešiteľné.

Príklad druhého typu úloh: "Ukážte, že Tom vie zviesť herečku. Zvedenie herečky možno zredukovať na získanie auta a získanie jachty. Tom má 5000$ a auto stojí 5000$ a jachta tiež 5000$.". Nilssonov algoritmus by chybne odpovedal, že Tom môže zviesť herečku. Problémy tohto typu sa vyskytujú aj pri plánovaní činnosti robota. Pre riešenie úloh tohto typu boli navrhnuté zovšeobecnené AND/OR grafy (Levi, Sirovich, 1975, 1976) [1], v ktorých môžu mať redukčné operátory dva alebo viac vstupných uzlov.

6.2.6. Prehľadávanie stromov hier

Procedúra minimax – uvažujme strom na obr. 6.20, každý uzol reprezentuje pozíciu hry, neterminálne uzly sú označené menom hráča, ktorý je na ťahu. Celý strom je konštruovaný z pohľadu hráča A a úlohou je nájsť jeho najlepší ťah v pozícii 1. Terminálne uzly sú označené hodnotou pre hráča A – *V*, *P*, *R* (výhra, prehra, remíza).



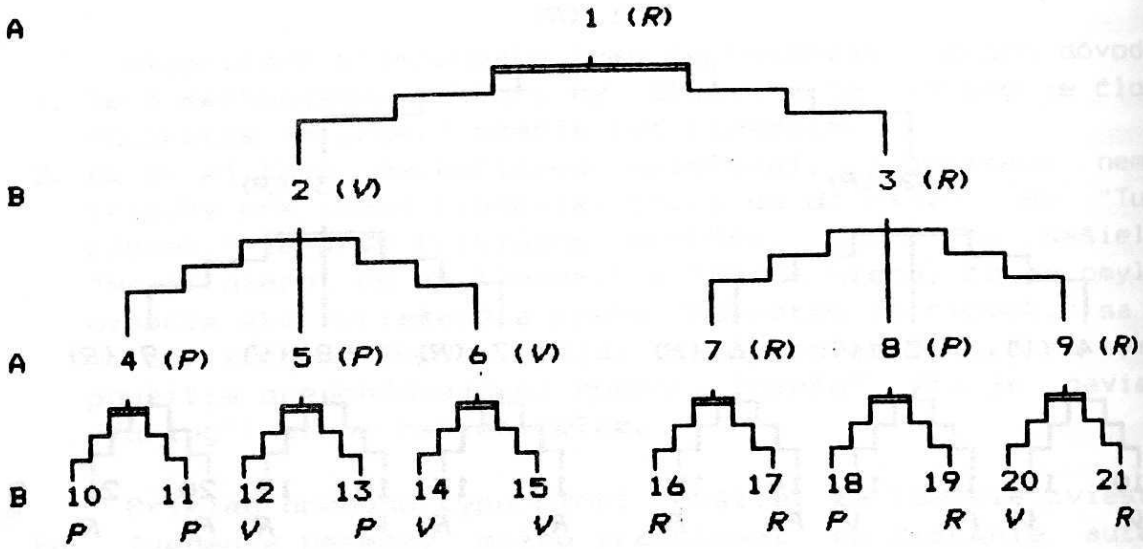
Obr. 6.20 Strom hry z pohľadu hráča A; V-výhra,P-prehra,R-remíza

Podľa procedúry minimaxu by mal hráč A ťahať na pozíciu 2 alebo 3, podľa toho, ktorá má väčšiu hodnotu (vyberá sa maximum). Hodnoty neterminálnych uzlov sa vypočítavajú z hodnôt terminálnych uzlov takto:

1. Hodnota (pre hráča A) uzla s OR potomkami (A ťahá v danej pozícii) je maximum hodnôt jeho potomkov.
2. Hodnota uzla s AND potomkami (B je na ťahu) je minimum hodnôt jeho potomkov.

Možný výsledok hry na obr. 6.20 možno číselne ohodnotiť takto: V=+1, P=-1, R=0; ohodnotenia neterminálnych uzlov sú na obr. 6.20 uvedené pri každom neterminálnom uzle v zátvorkách. Uzol 2 sa vyhodnotí na P (najlepšie, čo môže hráč B v pozícii 2 urobiť, je ťahať na pozíciu 6) a uzol 3 sa vyhodnotí na R (B najlepšie urobí, ak ťahá na 7 alebo na 9). Pri odhade súperovho konania sa totiž predpokladá, že tiež využíva procedúru minimax. Pri ohodnotení uzla s AND potomkami musí A predpokladať, že B urobí svoj najlepší možný ťah. Táto technika ignoruje možnosť, že B prehliadne istú možnosť výhry, ak A potiahne na 2 a takisto neuvažuje možnosť, že B uprednostní ťah na pozíciu 9 pred ťahom na 7 (z hľadiska ohodnotenia sú tieto pozície rovnaké).

Kvôli spôsobu ohodnocovania uzlov sa hráč A nazýva MAX (jeho pohľad reprezentuje strom) a hráč B sa nazýva MIN (niekedy sa používa aj označenie PLUS a MÍNUS). Strom z obr. 6.20 z pohľadu hráča MIN je na obr. 6.21 - AND a OR uzly sa vymenia a symetricky sa zmenia aj hodnoty uzlov.

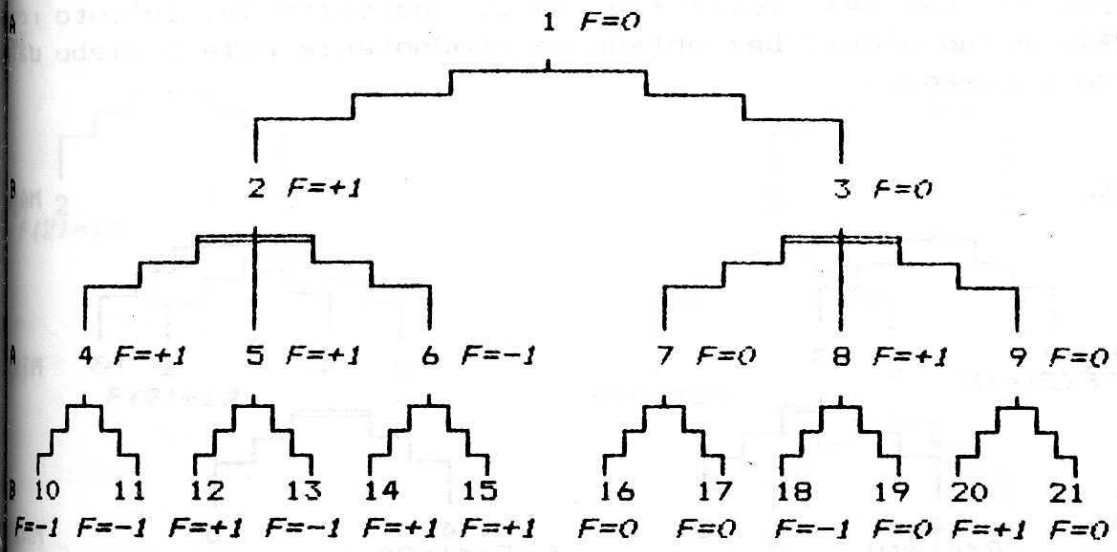


Obr. 6.21 Strom hry z obr. 6.20 z pohľadu hráča B

Negmax formalizmus - Knuth a Moore (1975) [1] navrhli reprezentáciu, ktorá unifikuje obr. 6.20 a 6.21 a umožňuje jednoduchým spôsobom realizovať výpočet optimálneho ťahu pre hráča A aj B. Ak n je terminálny uzol, jeho hodnota je f(n). Hodnota n pre druhého hráča je -f(n). Hodnota každého uzla sa potom vypočítava funkciou F takto:

1. F(n) = f(n), ak n nemá potomkov
2. F(N) = max{-F(n_1), ..., -F(n_k)}, ak n má potomkov n_1, ..., n_k

Najlepší ťah (pre obidvoch hráčov) je potom do uzla, pre ktorý nastalo maximum podľa bodu 2, t.j. hráč, ktorý je na ťahu v n, by mal ťahať z n do n_i, pre ktoré -F(n_i) = F(n). Táto formulácia sa nazýva negmax. Príslušný strom (pre hry z obr. 6.20, 6.21) je na obr. 6.22.

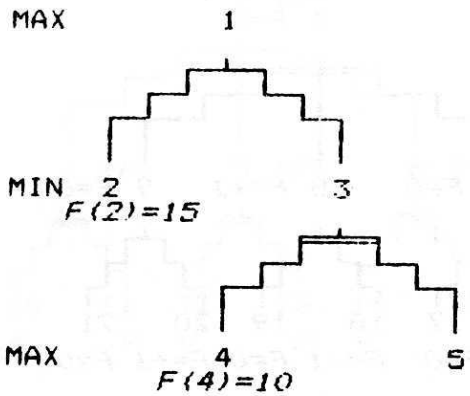


Obr. 6.22 Strom hry z obr.6.20 v NEGMAX notácii

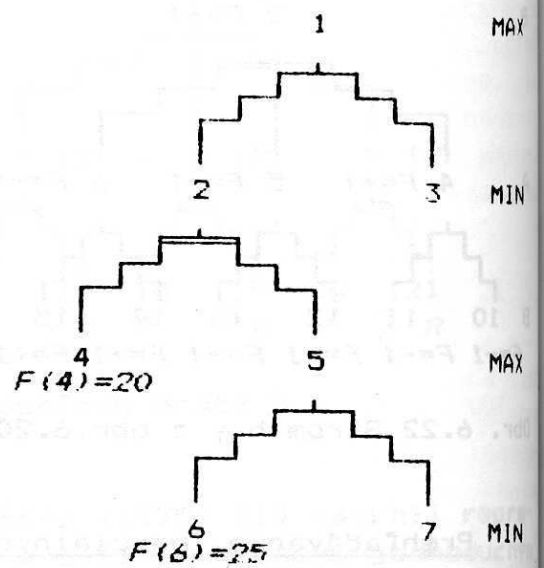
Prehľadávanie parciálnych stromov hier - doteraz sa predpokladalo, že existuje celý strom hry (až do koncových pozícií), čo pri praktickej aplikácii nie je možné. Preto sa väčšinou generuje iba určitá "rozumná" časť stromu, ktorá uvažuje všetky legálne ťahy v rámci určitého obmedzenia do hĺbky stromu (počet ťahov uvažovaných dopredu). V tom prípade je však potrebné ohodnotiť koncové uzly (tip nodes) - t.j. uzly parciálneho stromu bez potomkov. Na to sa používa statická hodnotiaca funkcia (static evaluation function) - obdoba heuristickej funkcie h* v Nilssonovom algoritme. Ak parciálny strom obsahuje terminálne uzly celého stromu hry (t.j. koncové uzly, ktoré reprezentujú koncové stavy hry - t.j. výhra, prehra, remíza), statická hodnotiaca funkcia vráti plus_nekonečno pre výhru, mínus_nekonečno pre prehru a 0 pre remízu; inak vráti konečnú hodnotu - kladnú v pozícii výhodnej pre hráča MAX (jej veľkosť určuje mieru výhodnosti pre hráča MAX) a zápornú v pozícii výhodnej pre hráča MIN. Minimaxná procedúra potom priradí spätné hodnoty (backed-up values) prechodcom koncových uzlov. Predpokladá sa, že tieto spätné ohodnotenia (backed-up evaluations) dávajú presnejší odhad skutočných hodnôt, ako by bolo možné získať statickou hodnotiacou funkciou priamo z týchto pozícií bez pohľadu niekoľko krokov dopredu.

Algoritmus alfa-beta (α-β) - úplné prehľadávanie stromu hry je často zbytočné, čo je ilustrované aj na obr. 6.23, 6.24. Uzly 2 a 4 na obr. 6.23 boli ohodnotené statickou hodnotiacou funkciou alebo spätným ohodnotením podľa potomkov. Nech MAX ťahá na pozíciu 2, odhad hodnoty tejto pozície je 15. Ak by ťahal na 3, hodnota uzla 3 je nanajvýš 10 (pretože hodnota jeho prvého potomka - uzla 6 je 10 a vyberá minimum hodnôt všetkých potomkov).

Preto by mal MAX voliť ťah na 2. Dôležité je, že toto rozhodnutie možno urobiť bez ohľadu na ohodnotenie uzla 5 alebo ďalších jeho potomkov.



Obr. 6.23 Alfa-odseknutie

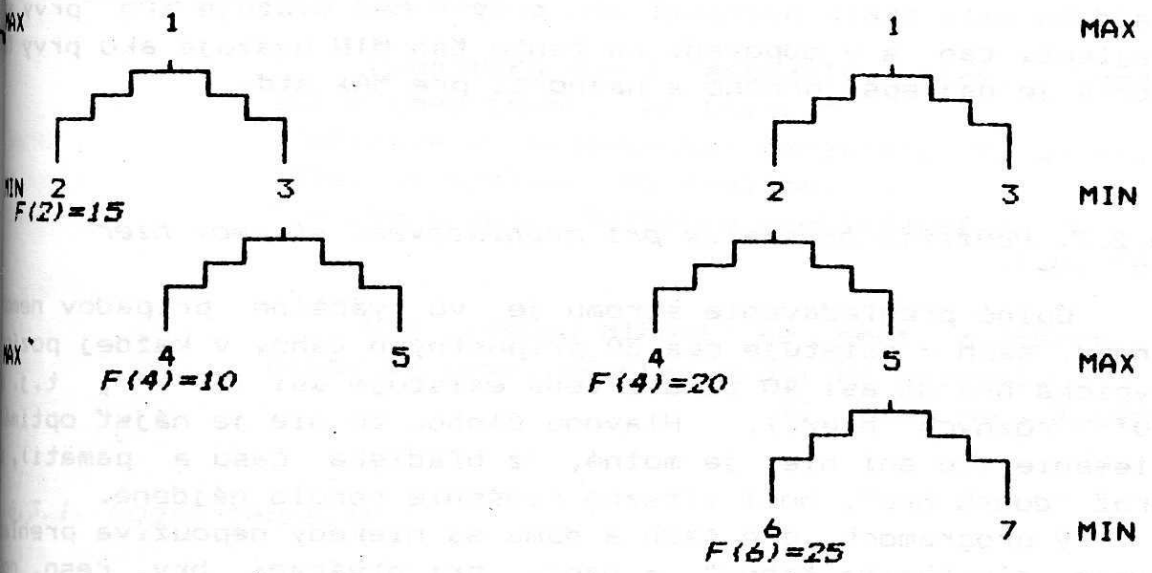


Obr. 6.24 Beta-odseknutie

Na obr. 6.24 je uzol 4 ohodnotený na 20; keď sa uzol 6 ohodnotí na 25, je jasné, že MIN sa vyhne ťahu na pozíciu 5 (v 5 ťahá MAX, volí maximum hodnôt a je isté, že toto maximum bude väčšie ako nanajvýš rovné 25, čo je hodnota prvého potomka uzla 5). Uzlu 4 môže byť teda priradená hodnota 20, bez nutnosti vyhodnotiť uzol 7 alebo ďalších potomkov uzla 7.

Algoritmus α-ß odstráni tieto nepotrebné vyhodnocovania. Ak sa generovanie uzlov strieda s ich vyhodnocovaním, potom uzly ako napr. potomkovia uzla 5 na obr. 6.23 alebo uzla 7 na obr. 6.24, nemusia byť ani generované. Algoritmus používa dva parametre - alfa a beta. Na obr. 6.23 sa α nastaví na 15 v uzle 2, eliminácia uzla 5 je α-odseknutie. Na obr. 6.24 sa ß nastaví na 20 v uzle 4 a reprezentuje hornú hranicu ohodnotenia uzla 2, eliminácia uzla 7 je ß-odseknutie. Procedúra garantuje, že výsledné ohodnotenie bude rovnaké, ako pri úplnom prehľadávaní stromu.

Pri formulácii algoritmu použijeme negmax reprezentáciu, obaja hráči sa snažia o maximalizáciu ohodnotení. Na obr. 6.23 sú obr. 6.23 a obr. 6.24 v negmax reprezentácii. Pre ohodnotenie uzla 1 sa volá funkcia VALUE (vracia číslo typu integer) s parametrami POSITION = uzol_1, ALPHA = mínus_nekonečno, BETA = plus_nekonečno. Statická hodnotiaca funkcia je f.



Obr. 6.25 NEGMAX reprezentácia obr. 6.23, 6.24

Výpočet ohodnotenia uzla (pozície p) použitím α-β algoritmu:

```
VALUE(p, α, β)
  position p;
  int α, β;
  {
  urči potomkov p1, p2, ..., pd pozície p;
  if(d == 0)
      {
      return(f(p));                          /* p je terminalny uzol */
      }
  else
      {
      m = α;
      for(i=1; i<=d; i++)                    /* cyklus pre synov uzla p */
           {
           t = -VALUE(p1, -β, -m);
           if(t > m) m = t;
           if(m >= β) return(m);                        /* odseknutie */
           }
      return(m);                             /* nedoslo k odseknutiu */
      }
  }
```

Zlepšenie efektívnosti prehľadavania použitím α-β algoritmu závisí od usporiadania potomkov, napr. ak by sa na obr. 6.23 najprv uvažoval uzol 3 a až potom uzol 2, nedošlo by k žiadnemu

odseknutiu. Vo všeobecnosti je výhodné, aby sa najlepší potomok každého uzla stále uvažoval ako prvý - MAX uvažuje ako prvý svoj najlepší ťah a v odpovedi na tento ťah MIN uvažuje ako prvý ťah, ktorý je najlepší preňho a najhorší pre MAX atď.

6.2.7. Použitie heuristík pri prehľadávaní stromov hier

Úplné prehľadávanie stromu je vo väčšine prípadov nemožné (napr. šach - existuje cca 30 prípustných ťahov v každej pozícii, typická hra má asi 40 ťahov, teda existuje asi $(30^2)^{40}$, t.j. asi 10^{120} rôznych hier!). Hlavnou úlohou tu nie je nájsť optimálne riešenie (to ani nie je možné, z hľadiska času a pamäti), ale hrať "dobrú hru", hoci víťazné riešenie nebolo nájdené.

V programoch pre šach a dámu sa niekedy nepoužíva prehľadávanie, ale "kniha ťahov" - napr. pri otváraní hry, resp. počas hry sa rozpozná nejaká typická situácia a podľa toho sa zvolí metóda hry. Shannon [1] navrhol dve vylepšenia oproti základnému algoritmu prehľadávania. Typ A:

1. strom hry sa generuje do fixnej hĺbky
2. uzly na tejto hĺbke (v skutočnosti sú neterminálne) sa ohodnotia statickou hodnotiacou funkciou. To sa opakuje z každej novej pozície.

Typ B:

1. hĺbka stromu nie je fixne daná
2. selektívnosť pri prehľadávaní, viac pozornosti sa venuje sľubným ťahom

Statická hodnotiaca funkcia - väčšinou sa nepokúša priamo odhadovať vzdialenosť danej pozície od víťaznej. Zvyčajne je to lineárna funkcia, ktorej členy reprezentujú materiál (figúrky), bezpečnosť kráľa, mobilnosť, ovládanie stredu poľa, rozloženie pešiakov apod. Samuel [1] považuje za optimálny počet takýchto príznakov pre dámu 20 až 30.

Hĺbka prehľadávania - nemožno prehľadávať celý strom do hĺbky (už pre šesť ťahov - tri pre každého hráča - by existovalo $(30^2)^3 = 10^9$ koncových uzlov), ale šachoví majstri občas odhadujú 15 až 20 ťahov dopredu (pozdĺž jednej línie hry).