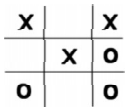
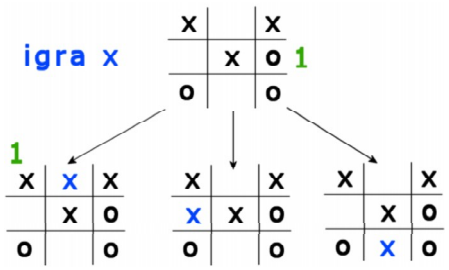
Backtracking na primjeru igre Križić-kružić

Dana je tablica za igru križić-kružić na kojoj su već odigrani neki potezi. Želimo utvrditi koji igrač ima strategiju koja vodi do pobjede, te koji potez je optimalni potez igrača koji je na redu – na koje polje treba staviti svoju oznaku da bi pobjedio ako je to moguće, ili izvukao nerješeno ako je to moguće.

Zamislimo da je u danoj situaciji na potezu igrač x. Ako postoji način da on pobjedi, toj situaciji ćemo dodjeliti oznaku 1. U protivnom, ako ne može pobijediti, ali može igrati nerješeno, situaciju označavamo sa 0, a ako gubi kako god igrao, situaciju ćemo označiti sa -1.
Slično, situacije u kojima je na potezu o označavamo sa -1 ako o može pobijediti, sa 0 ako može igrati najviše izjednačeno i sa 1 ako gubi.
Neka je npr. dana ova situacija:

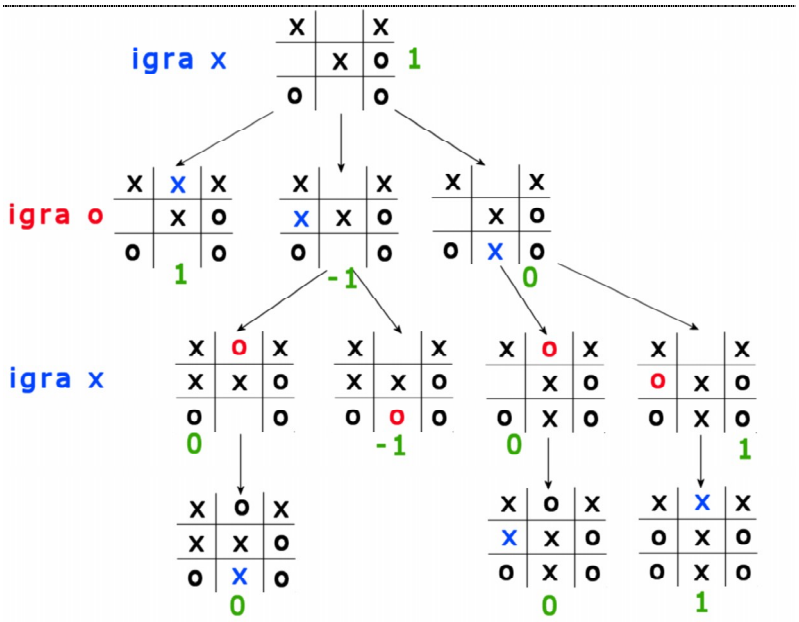


Na potezu je križić; on ima tri polja na koja može igrati:



Dobivamo stablo situacija. Situacija u lijevom djetetu korijena ovog stabla trivijalno ima oznaku 1. Iako za ostalu djecu još nemamo oznake, možemo utvrditi da će oznaka korijena biti također 1, jer x na potezu može pobjediti.

Pogledajmo ipak što bi se dogodilo da x ne odigra svoj pobjednički potez, tko dobiva u preostalim situacijama:



Pogledajmo npr. zašto je desno dijete korijena dobilo oznaku 0. Tu je na potezu o, vidimo da djeca od te situacije imaju oznake 0 i 1. Najbolje što o u tom trenutku može napraviti je otići u dijete označeno nulom (jer u njemu igra izjednačeno; da je odigrao potez koji vodi u polje označeno sa 1, izgubio bi). Da je postojalo dijete označeno sa -1, kružić bi mogao pobjediti i desno dijete korijena bi također imalo oznaku -1. Vidimo da će oznaka svakog čvora u kojem je na potezu o biti jednaka minimumu svih oznaka djece tog čvora. Analogno, lako se vidi da je oznaka svakog čvora u kojem je na potezu x jednaka maksimumu svih oznaka djece tog čvora.

Da rezimiramo: da bismo odredili tko u zadanoj situaciji pobjeđuje potrebno je konstruirati stablo svih situacija dostižljivih iz zadane. Korijen tog stabla je upravo zadana situacija, a djeca svakog čvora su situacije do kojih se u jednom potezu dolazi iz tog čvora.
Očito težina utvrđivanja pobjednika pada što je nivo čvora u stablu veći: listovi stabla su trivijalne situacije u kojima je neki od igrača pobijedio ili je tablica popunjena do kraja. Da utvrdimo tko je pobjednik (oznaka) u čvoru u kojem je na potezu x potrebno je odrediti maksimum svih oznaka djece toga čvora – dakle, riješiti sve jednostavnije situacije.
Da utvrdimo tko je pobjednik (oznaka) u čvoru u kojem je na potezu o potrebno je odrediti minimum svih oznaka djece toga čvora.

Konstrukcija ovog stabla, odnosno svih situacija koje moramo obići radi se rekurzivnim algoritmom, tj. tehnikom backtracking. Evo i pseudo-koda za naš problem:

```
int krizic_kruzic (tablica T, char na_potezu, potez *treba_igrati)
{
// ulaz: tablica=djelomicno popunjeno polje za igru
// ulaz: na_potezu=oznaka igraca koji je trenutno na potezu
// izlaz: treba_igrati=optimalan potez kojeg igrac na potezu moze
//                     odigrati u ovoj situaciji
// izlaz: funkcija vraca oznaku cvora u stablu koji odgovara tablici T
    tablica dijete;
    potez ptemp;
    int vrijednost_djeteta, temp;

    if (T je list)
        // ploca je puna - fja pobjednik vraca 1 ako pobjedjuje x, -1 ako o,
        //                 0 ako je nerjeseno
        return pobjednik (T);
    else
    {
        if (na_potezu=='x')
            vrijednost_djeteta=-beskonacno;
        else
            vrijednost_djeteta=+beskonacno;

        za svaki legalan potez move u tablici T radi
        {
            if (na_potezu=='x')
            {
                dijete=T + na mjestu move je stavljen 'x';
                temp=krizic_kruzic (dijete, 'o', &ptemp);
```

```
            if (temp > vrijednost_djeteta)
            {
                vrijednost_djeteta=temp;
                *treba_igrati=move;
            }
        }
        else
        {
            dijete=T + na mjestu move je stavljen 'o';
            temp=krizic_kruzic (dijete, 'x', &ptemp);
            if (temp < vrijednost_djeteta)
            {
                vrijednost_djeteta=temp;
                *treba_igrati=move;
            }
        }
    }

    return vrijednost_djeteta;
}
```

Ako pokrenemo ovaj algoritam na praznoj tablici, dobijemo da je njezina oznaka 0. To znači da niti jedan igrač nema pobjedničku strategiju u igri križić-kružić, tj. ako oba igrača igraju dovoljno pametno, igra uvijek završava nerješeno.