

Algoritmi i strukture podataka

vežbe 10

Mirko Stojadinović

16. decembar 2015

1 Algoritamske strategije - bektreking

Bektreking algoritmi tragaju za svim rešenjima nekog problema. Najpoznatiji problem ove vrste je problem osam kraljica (dama). Ovi algoritmi su posebno korisni kada se lako mogu odbaciti mnoge mogućnosti za rešenja. Neki problemi koji se rešavaju na ovaj način su rešavanje ukrštenica, slagalica, Sudokua. Ovi algoritmi nalaze se u osnovi logičkih programske jezika, npr. Prologa.

Problem: zadata je tablica za igru iks oks na kojoj su već odigrani neki potezi. Treba utvrditi koji igrač ima strategiju koja vodi do pobeđe, te koji potez je optimalan potez igrača koji je na redu - na koje polje treba staviti svoju oznaku da bi pobedio ako je to moguće ili odigrao nerešeno ako je moguće.

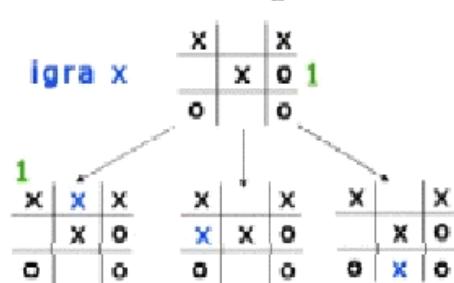
Neka je na potezu igrač x. Ako postoji način da on pobedi, toj situaciji se dodeli oznaka 1. Ako ne može pobediti, ali može igrati nerešeno, situacija se označava sa 0, a ako gubi kako god igrao situacija se označava sa -1. Slično, ako je na potezu o, situacija u kojoj on može pobediti se označava sa -1, kada može igrati najviše izjednačeno sa 0 i kada gubi sa 1.

Primer: zadata je ovakva situacija:

X		X
	X	O
O		O

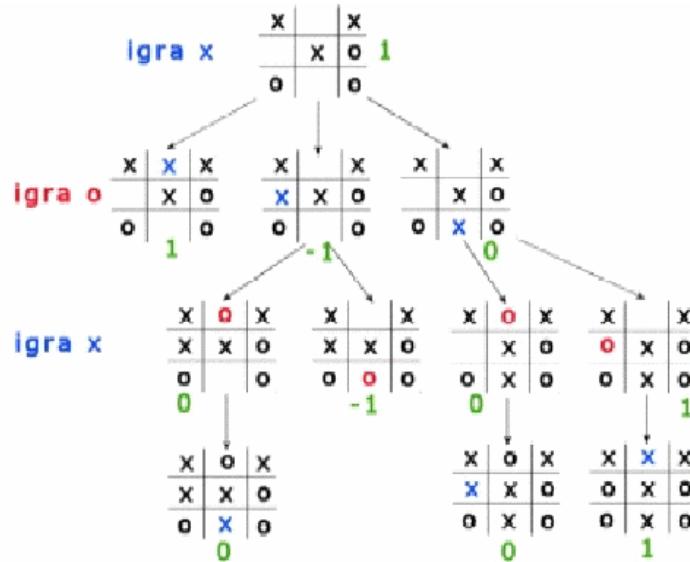
Na potezu je IKS, ima tri polja na koja može igrati. Dobije se stablo situacija u kome:

- * levo dete trivijalno ima oznaku 1, jer je pobeda za x.
- * za ostalu decu se još ne znaju oznake, ali se zna da koren ima oznaku 1, jer je x na potezu i može pobediti



Ako x ne odigra svoj pobednički potez, moguće su ove situacije:

1. Desno dete je dobilo oznaku 0, jer je na potezu o i ako on odigra prvi potez, rezultat je nerešen, ako odigra drugi, pobediće x
2. Srednje dete dobija oznaku -1, jer zavisno od toga šta igra o, onda x gubi ili igra nerešeno



Iz ovog je jasno da je oznaka svakog čvora u kojem je na potezu o jednaka minimumu svih oznaka dece tog čvora i analogno ako je na potezu x oznaka čvora je maksimum svih oznaka dece tog čvora.

Dakle, algoritam rešavanja je: da bi se odredilo ko u zadatoj situaciji pobeduje treba kreirati stablo svih situacija dostižnih iz zadate situacije; koren stabla je zadata situacija, a deca svakog čvora su situacije do kojih se dolazi u jednom potezu iz tog čvora. Težina utvrđivanja pobednika pada što je nivo čvora u stablu veći. Listovi stabla su trivijalne situacije završetak igre. Za utvrđivanje oznake u čvoru u kojem je na potezu x potrebno je naći maksimum svih oznaka dece tog čvora; za određivanje oznake kad je na potezu o potrebno je naći minimum svih oznaka dece tog čvora. Konstrukcija takvog stabla se može obaviti rekurzivnim algoritmom.

1. Napisati funkciju u programskom jeziku C koja koristeći backtracking ispisuje sve binarne brojeve od n cifara.

Rešenje:

Početni poziv funkcije bi bio `ispisi_binarne (a, 0, n)`.

```
void ispisi_binarne (int a[], int i, int n){
    if (i == n){
        int j;
        for (j=0; j<n; j++)
            printf ("%d", a[j]);
        printf ("\n");
    }
    else {
        a[i] = 0;
        ispisi_binarne (a, i+1, n);
        a[i] = 1;
        ispisi_binarne (a, i+1, n);
    }
}
```

Koje bi modifikacije bile neophodne kada bi umesto binarnih trebalo ispisati sve oktalne brojeve?

2. Napisati funkciju u C-u koja na ulazu prima dva celobrojna niza A i B, oba dimenzije n , i na izlaz štampa sve moguće nizove dimenzije n koji na i-toj poziciji imaju ili broj $A[i]$ ili broj $B[i]$, i čija se svaka dva uzastopna člana razlikuju najviše za 5.

Npr. za nizove

```
A: 3 1 7
B: 6 0 4
```

na ekran treba da se odštampa

```
3 1 4
3 0 4
6 1 4
```

REŠENJE 1 - gruba sila, LOŠE (neefikasno) rešenje:

```
void mesavina (int a[], int b[], int c[], int i, int n){
    int j;
    if (i == n){
        for (j=0; j<n-1; j++)
            if (abs(c[j] - c[j+1]) > 5)
                break;

        if (j == n-1){
            for (j=0; j<n; j++)
                printf("%d ",c[j]);
            printf("\n");
        }
    }
    else {
        c[i] = a[i];
        mesavina (a, b, c, i+1, n);
        c[i] = b[i];
        mesavina (a, b, c, i+1, n);
    }
}
```

Bolje rešenje je da se kad god je to moguće uradi odsecanje, tj. da se ne izvršavaju oni rekursivni pozivi koji ne mogu da dovedu do rešenja.

REŠENJE 2 - korišćenje odsecanja, DOBRO (efikasnije) rešenje:

```
void mesavina (int a[], int b[], int c[], int i, int n){
    int j;

    if (i == n){
        for (j=0; j<n; j++)
            printf("%d ",c[j]);
        printf("\n");
    }
    else {
        if (i==0 || abs (a[i] - c[i-1]) <= 5){ // odsecanje
            c[i] = a[i];
            mesavina (a, b, c, i+1, n);
        }
        if (i==0 || abs (b[i] - c[i-1]) <= 5){ // odsecanje
            c[i] = b[i];
            mesavina (a, b, c, i+1, n);
        }
    }
}
```

Prethodni algoritam ima maksimalno 2^n rešenja. Kolika je složenost tog algoritma?

3. Napisati C program koji za datu sumu S i dati niz v koji sadrži vrednosti novčanica pronalazi sva rešenja za rasitnjavanja sume S. Prepostaviti da svake novčanice ima proizvoljno mnogo.

U čemu je razlika ovog zadatka i zadatka rešavanog tehnikom pohlepnog algoritma?

Rešenje:Naredni kod u .c fajlu

```
#include <stdio.h>
```

```

int s; /* novcani iznos koji treba razbiti */
int n; /* broj razlicitih novcanica */
int v[10]; /* vrednosti novcanica/pona */
int x[10]; /* kolicina pojedinih ponaa */

void pisi();
void razmeni(int k, int s);

int main(){
    int i;
    printf("Unesite sumu i broj novcanica\n");
    scanf("%d%d",&s,&n);

    printf("\nUnesite vrednost za %d novcanica\n",n);
    for(i=0; i<n; i++)
        scanf("%d",&v[i]);
    razmeni(0,s);

    return 0;
}

void pisi(){
    int i;
    printf("\nRazmena: \n");
    for(i=0; i<n; i++)
        printf("%d dinara: %d puta\t\t", v[i],x[i]);
    printf("\n");
}

/* k = redni broj ponaa, s = tekuca suma za rasitnjavanje */
void razmeni(int k, int s){

    int i;

    if(k>=n) {
        if(s==0)
            pisi();
    }
    else
        // i oznaava broj ponaa k koje koristimo u sumi
        // odsecanje je i<=s/v[k] (ne nastavlja se u slucaju kada suma prevaziđe s)
        for(i=0; i<=s/v[k]; i++){
            x[k]=i;
            razmeni(k+1, s-i*v[k]);
        }
}

```

4. Za n studenata i n studentkinja jednog fakulteta dati su podaci o ukupnom broju osvojenih poena na kvalifikacionom testu za kviz. Odrediti što je moguće više parova koji se mogu formirati unutar fakulteta i prijaviti za kviz parova ako svaki par čine po jedna studentkinja i jedan student i ako razlika u broju poena kod svakog para ne sme da predje dati broj p.

5. Dat je niz celih brojeva $a[0]...a[n-1]$ i ceo broj S. Napisati program koji postavlja operatore + - * / u izrazu $(...a[0]?a[1])?a[2]?...?a[n-1]$ tako da vrednost izraza bude S. Naći sva rešenja.

Rešenje:Naredni kod u .c fajlu
Dat je C program (bez f-je ispisa).

```

#include <stdio.h>

int s; /* vrednost izraza */
int n; /* broj operanada */
int jeste; /*indikator 0/1 da li vrednost izraza je s*/
int a[50]; /* vrednosti operanada izraza */
int z[50]; /* niz operatora izraza */

void pisi (){
    int i;

    for (i=0; i<n; i++)
        printf ("%d %c ",a[i], z[i]);
    printf ("\n");
}

void racunaj(int k, int ts){
    //k = redni broj primenjen operacije
    //ts = tekuća suma izraza
    if(k==n-1)
    {
        if(ts==s)
        {
            pisi();
            jeste=1;
        }
    }
    else
    { /*formiranje operatora unutar izraza */
        z[k]='+'; racunaj(k+1, ts+a[k+1]);
        z[k]='-'; racunaj(k+1, ts-a[k+1]);
        z[k]='*'; racunaj(k+1, ts*a[k+1]);
        if(a[k+1]!=0)
        {
            z[k]='/';
            racunaj(k+1, ts/a[k+1]);
        }
    }
}
}

int main(){
    int i;
    printf("Unesite vrednost izraza i broj operanada izraza\n");
    scanf("%d%d",&s,&n);
    printf("\nUnesite vrednost za %d operanada\n",n);
    for(i=0; i<n;i++)
        scanf("%d",&a[i]);
    jeste=0;
    racunaj(0,a[0]);
    if (!jeste)
        printf("\nNema resenja\n");

    return 0;
}

```

6. Napisati program koji rešava igru Moj broj.

Rešenje: Program koji u glavnom nalazi dobro rešenje problema.

7. Otvoren skakačev hod je niz od $n^2 - 1$ skakačevih poteza na šahovskoj tabli dimenzija $n \times n$, takvih da je svako polje posećeno tačno jednom. Napisati pseudokod algoritma složenošći $O(8^{n^2})$ za nalaženje broja svih otvorenih

skakačevih hodova počev od unapred zadate pozicije.

REŠENJE:

br_resenja = 0

Algoritam Skakacev_obilazak (M, a, b, br_popunjeneh_polja)

Ulaz Matrica posećenih polja M, mesto na kome se nalazi skakač (a,b)

Izlaz

```
{  
    if (br_popunjeneh_polja == n*n)  
        br_resenja++;  
    else  
        za svako polje (c, d) na koje postoji legalan skok sa polja (a, b)  
            if M[c][d] == 0 {  
                M[c][d] = 1;  
                Skakacev_obilazak (M, c, d, br_popunjeneh_polja + 1);  
                M[c][d] = 0;  
            }  
    }  
}
```

Algoritam Pokreni_obilazak (M, a, b)

Ulaz Matrica posećenih polja M, mesto na kome se nalazi skakač (a,b)

Izlaz Broj otvorenih skakačevih hodova

```
{  
    M[a][b] = 1;  
    Skakacev_obilazak (M, a, b, 1);  
    stampaj (br_resenja);  
}
```

Zadatak: Napisati modifikaciju prethodnog algoritma u programskom jeziku C koja umesto brojanja na izlaz štampa sve otvorene skakačeve hodove. Program testirati na dimenzijama tabli 6×6 i manje (tabele 7×7 imaju ogroman broj kombinacija). Napisati modifikaciju prethodne funkcije koja ispisuje prvi otvoreni skakačev kod koji pronađe. Ovu funkciju testirati i na većim dimenzijama.

8. Napisati funkciju u programskom jeziku C koja ispisuje sve binarne brojeve od n cifara koji nemaju dve uzastopne jedinice.