Pretraga sa vraćanjem - backtracking

Backtracking algoritmi traže rešenje:
1. metodima pokušaja i sistematskim ispitivanjem svih mogućnosti za konstrukciju rešenja i
2. vraćanjem u slučaju greške.

Backtracking je veoma opšta metoda koja se primenjuje za teške kombinatorne probleme.

Backtrackingom se mogu rešavati zadaci koji zahtevaju da se:
• nađu sva moguća rešenja
• nađe bar jedno rešenje problema;
• nađe optimalno rešenje prema zadatom kriterijumu.

Ovi problemi se često zahtevaju ispitivanje konačnog broja potproblema i traženje rešenja potproblema je često rekurzivno.

1. Dat je niz celih brojeva **a[0]...a[n-1]** i ceo broj **S**. Konstruisati algoritam koji postavlja operatore + - * / u izrazu (...(a[0]?a[1])?a[2])...)?a[n-1] tako da vrednost izraza bude S. Naci sva resenja.

Dat je pseudo-kod (bez f-je ispisa). Napisite je sami !!!

```
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 racunaj(int k, int ts)
{
//k= redni broj primenjen operacije
//ts = tekuca 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]); }
}
}

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");
}
```

2. Donatelo je jedan od nindža kornjača koji ima specijalni zadatak da upadne u mrežu tajnih neprijateljskih kanala, ali Donatelo još nije dovoljno sposoban **da se može neopaženo kretati po svetlosti**. Ali, Donatelo je dovoljno vešt da se može neopaženo kretati po mraku. Tokom zadatka, Donatelo je dospeo u prostoriju koja ima formu kvadratne ***NxN*** ($2 \leq N \leq 14$) matrice čije svako polje je jedna soba. Svaka soba ima sopstveno osvetljenje i ono je za svaku sobu ili uključeno ili isključeno. Štaviše svake **sekunde svetla menjanju svoj status** – uključena svetla se gase, isključena svetla se pale.

Na primer, neka su tokom prve sekunde sobe prikazane na slici levo, a tokom druge sekunde neka su sobe prikazane na slici desno (neobojene ćelije tabele predstavljaju osvetljene sobe, sive ćelije predstavljaju sobe sa isključenim svetlom).

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

Donatelo može prelaziti samo u one susedne sobe koje su povezane zajedničkim zidom sa njegovom tekućom sobom i te sobe moraju se nalaziti nadole ili nadesno u odnosu na njegovu tekuću sobu. Na primer, gledajući datu tabelu, ako Donatelo je u sobi (1,1) on se može kretati do sobe (1,2) i do sobe (2,1). Ulazak u narednu sobu se obavlja tačno u svakoj sledećoj sekundi. Dakle, ako u trenutku promene položaja, Donatelo uskače u sobu sa upaljenim svetlom, to nije problem, jer u trenutku ulaska, soba će imati isključeno svetlo. Donatelo može ostati u sobi dokle god želi **(0 sekundi ili više)**. Kada je Donatelo u sobi, svetlo u sobi, naravno ostaje isključeno sve dok on ne napusti sobu.

Tokom prve sekunde, Donatelo je u sobi koja se nalazi u gornjem levom uglu prostorije (0, 0) i on može da izabere da ostane u toj sobi tokom naredne sekunde ili da uskoči u susedne sobe (0, 1) ili (1, 0).

Napisati program koji pronalazi najkraće vreme (u sekundama) za koje će Donatelo iz sobe (0,0) dospeti do sobe (*N*-1, *N*-1) tako da ostane neopažen. Ulazak u prvu sobu takođe traje jednu sekundu.

Na ulazu se zadaje ceo broj *N,* a potom se iz svake od N narednih linija učitava po N brojeva 0 ili 1 koji označavaju da je u odgovarajućoj sobi svetlo isključeno ili uključeno. Kako se u narednoj sekundi status osvetljenja menja, smatrajte da soba (0, 0) će uvek imati ugašeno svetlo.

ULAZ

```
4
0 1 1 1
0 1 0 0
1 0 1 1
1 0 1 1
```

IZLAZ

```
10
```

Pojašnjenje

Tokom prve sekunde Donatelo je u sobi (0,0). U narednoj sekundi svetla se menjaju i Donatelo to zna, te mora odabrati ili da ostane u sobi (0,0) i čeka narednu promenu statusa osvetljenja ili da ode do sobe (0,1) (koja će u narednoj sekundi biti neosvetljena). Poštujući pravila, Donatelo ne može stići u sobu (3,3) pre 10. sekunde, na primer putanjom *(0,0), (0,1), (0,1), (1,1), (2,1), (2,2), (2,2), (2,3), (2,3), (3,3).*

```
#include <iostream>
#define MAXN 15
#define MAXTIME 2000000000
#define INIT_STATE_ON 1
#define INIT_STATE_OFF 0
using namespace std;

int N;
int matrix[MAXN][MAXN];
int bestAnswer = MAXTIME;
```

```
void bt(int time, int x, int y)
{
    // provera da li soba matrix[x][y] je osvetljena
    if (matrix[x][y] == INIT_STATE_ON && (time % 2) == 1) return; //time = 1, 3, 5...
    if (matrix[x][y] == INIT_STATE_OFF && (time % 2) == 0) return; //time = 2, 4, 6...

    // provera da li je Donatelo dospeo do sobe (N-1, N-1)
    if (x == N - 1 && y == N - 1)
    {
        if (time < bestAnswer) bestAnswer = time;
        return;
    }

    // Donatelo nece da ceka, vec uskace u susednu sobu,
    // razmislja metodom backrackinga bt(time + 1, x + 1, y); bt(time + 1, x, y+1); tj. ispistuje svaku
//susednu sobu i vraca se u slucaju greske
    if (x + 1 < N) bt(time + 1, x + 1, y);
    if (y + 1 < N) bt(time + 1, x, y + 1);

    // Donatelo hoce da saceka 1 sekundu i prelazi u sledece sosbe
//opet razmislja metodom backtracking-a
    if (x + 1 < N) bt(time + 2, x + 1, y);
    if (y + 1 < N) bt(time + 2, x, y + 1);
}

int main()
{
    // ucitavanje podataka o osvetljenosti soba
    cin >> N;
    for(int i = 0; i < N; i++)
    {
        for(int j = 0; j < N; j++)
        {
            cin >> matrix[i][j];
        }
    }

    // pozovi backtracking metod za 1. sekundu i sobu (0,0)
    bt(1, 0, 0);

    // ispis vremena
    cout << bestAnswer << endl;

    return 0;
}
```

3. Mnoge porodice u Sremskim Karlovcima se tradicionalno bave vinogradarstvom i proizvodnjom vina. Vinarija *SremskiBermet* je konačno dobila dugočekivanu narudžbinu od $S(0 < S \leq 1000000000)$ litara svog najboljeg crnog vina. Vino se mora transportovati u burićima kako bi očuvalo karakteristična svojstva. Vinarija raspolaže buradima sa n ($1 \leq n \leq 100$) različitih kapaciteta, a najmanji kapacitet je 1 litar. Broj burića svakog kapaciteta je neograničen. Vinarija mora da izabere što je moguće manje burića za transport, tako da je njihov broj bude najmanji mogući. Napišite program, koji, za date S, n i kapaciteta raspoloživih n burića (jedan od datih kapaciteta je jednak 1), određuje minimalan broj buradi, koje se koriste za prevoz vina.

ULAZ

```
10000 7
12 1 11 30 14 2 18
```

IZLAZ

```
335
```

Da biste izabrali odgovarajuće buriće, tako da je njihov broj minimalan, oni treba da budu od velikog kapaciteta, tako da najpre treba sortirate po opadajućem redosledu vrednosti raspoloživih kapaciteta.
Potom treba ispitati koliko burića svakog kapaciteta možemo iskoristiti da smestimo preostalu količinu vina. Ako se dobije tačna količina raspoloživog vina, proces se zaustavlja i to je minimalan broj buradi se traži. Ako nije dostignut tačan iznos, proces se nastavlja. Zadatak uvek ima rešenje, jer postoji bure kapaciteta 1.
Zadatak se može rešiti korišćenjem metoda "backtracking" ili dinamičkim programiranjem. Mi smo odabrali backtracking.

```
#include <iostream>
using namespace std;
 int  s,n,i,p,w,k=1000000000,j;
 int B[10000] ;

void minibure(int p,int T, int m)
 { //p je tekuci broj bureta, T je kapacitet nasutog vina u burice
   //m je broj upotrebljenih burica
  int br;
  br=(s-T)/B[p]; //max broj burica kapaciteta B[p]
  if (m+br<k) {
   if (T+br*B[p]==s)k=m+br; //proces se zaustavlja, k je min broj burica
            else if (p<n) //ako nismo upotrebili jos sve burice
            while (br>=0)
            {
            minibure(p+1, T+br*B[p],m+br);
            br--;
            }
   }
 }

 int main()
{
  cin >> s>>n;
  for(i=1;i<=n;i++) cin >>B[i]; // burici B[1], B[2],..., B[n]
  for (i=n; i>=2; i--){
       p=1;
       for (j=2; j<=i; j++)
              if (B[p]>B[j])p=j;
              w=B[p]; B[p]=B[i]; B[i]=w;
          }
  minibure(1,0,0);
  cout <<k<<endl;
}
```

4. Data je geografska karta simetričnom matricom susedstva A(nxn) gde je a[i,j]=1 ako je država *i* susedna državi *j*, a[i,j]=0 inače. Konstruisati algoritam kojom se boji karta sa 4 boje (1, 2, 3, 4) tako da su susedne države obojene različitim bojama. Naći jedno resenje, a ukoliko rešenja nema dati odgovarajuću poruku.

ULAZ

0	1	1	0	1	1
1	0	1	1	1	1
1	1	0	1	1	0
0	1	1	0	1	1
1	1	1	1	0	1
1	1	0	1	1	0

IZLAZ

```
Drzava - Boja
0 - 1
1 - 2
2 - 3
3 - 1
4 - 4
5 - 3
```

Rešenje:
U programu se koristi matrica A(nxn) za evidentiranja susedstva.
Matrica A je simetrična => dovoljno je učitavati vrednosti donjeg trougla matrice, a elementima gornjeg trougla dodeljivati vrednosti simeticne u odnosu na glavnu dijagonalu.
Rezultat bojenja se daje nizom $boja[i],\ i=1,\ldots,n$ ciji elementi uzimaju vrednosti: 0 (dok je drzava neobojena), 1, 2, 3 ili 4 (nakon bojenja nekom od 4 boje).

```
#include <iostream.h>
int a[20][20], boja[20];
int n;

int SusedOd(int k, int c)
{// proverava da li je neki od suseda drzave k obojen bojom c
for (int j=1; j<=n; j++)
if (a[k][j]==1 && boja[j]==c) return 1;
re turn 0;
}

void pisi(int n)
{
for (int i=1; i<=n; i++)
cout << "Drzava " << i << " je obojena bojom " << boja[i]<< endl;
cout << "_____" << endl;
}

void Boji(int k)
{
for (int c=1; c<=4; c++) // proba da drzavu K boji nekom od 4 boje
if (!SusedOd(k,c))// ako susedi od K nisu obojeni bojom c
{
  boja[k]=c; // oboji drzavu K bojom c
  if (k==n)pisi(n); // ako su sve drzave obojene, pisi resenje
  else Boji(k+1); // ako nisu, nastavi sa bojenjem
  boja[k]=0; // odustaje se od tekuceg izbora boje da bi se probalo
  // sa drugom bojom
}
}

void main()
{
cin >> n;

for (int i=2; i<=n; i++)
for (int j=1; j<i; j++)
 {
  cout << "Da li se drzava " << i << " granici sa " << j<< ":";
  cin >> a[i][j];
   a[j][i]=a[i][j];
  }

Boji(1);
}
```

5. Napisati C program koji će učitati cele brojeve S,n , potom niz celih brojeva v dimenzije n i za datu sumu S i dati niz v koji sadrzi vrednosti novcanica ispisati sva resenja za rasitnjavanja sume S. Pretpostaviti da svake novcanice ima proizvoljno mnogo.

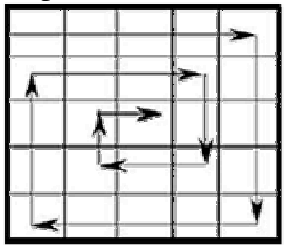
```
#include <stdio.h>
int s; /* novcani iznos koji treba razbiti */
int n; /* broj razlicitih novcanica */
int v[10]; /* vrednosti novcanica */
int x[10]; /* kolicina pojedinih novcanica */
void pisi()
{ int i;
```

```
printf("\nRazmena: \n"); for(i=0; i<n; i++) printf("%d dinara:%d puta ", v[i],x[i]);
printf("\n");
}
void razmeni(int k, int s)
{//k= redni broj monete //s = tekuca suma za rasitnjavanje
   int i;
   if(k>=n) { if(s==0) pisi();}
   else
    for(i=0; i<=s/v[k];i++)
       { x[k]=i;
         razmeni(k+1, s -i*v[k]);
        }
}

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);
}
```

6. Neka je data specijalna kvadratna matrica puž sa sledećim karakteristikama:

- elementi matrice pripadaju skupu {0, 1, 2, 3};
- svaki red i kolona sadrže svaku od vrednosti 1, 2, 3, tačno jednom, dok su ostale vrednosti u matrici jednake 0;
- počev od gornjeg levog ugla, kretanjem po spirali, nenulte vrednosti se pojavljuju u redosledu 1, 2, 3, 1, 2, 3, ... , 1, 2, 3.

Na primer kvadratna matrica 5x5 će poštujući sledeće karakteristike, izgledati:



0	1	0	2	3
0	2	3	0	1
1	3	0	0	2
3	0	2	1	0
2	0	1	3	0

Za dati prirodan broj n, potrebno je generisati nxn puž matricu.

Ulaz

U jedinoj liniji standardnog ulaza nalazi se jedan prirodan broj n (5 <= n <= 200).

Izlaz

Ako postoji rešenje, onda se na standardnom izlazu mora štampati n linija, a u svakoj liniji po n brojeva razdvojenih jednim blanko karakterom, koji predstavljaju zahtevanu matricu. Ako postoji više rešenja, ispisati ma koje.
Ako ne postoji rešenje, ispisati -1.

Napomena

Vremensko ograničenje: 1 s
Memorijsko ograničenje: 64 MB

Primer

ulaz	izlaz
5	0 1 0 2 3
	0 2 3 0 1
	1 3 0 0 2

3 0 2 1 0
2 0 1 3 0

Drugo korektno rešenje:
0 1 2 3 0
2 3 0 0 1
0 0 3 1 2
1 0 0 2 3
3 2 1 0 0

Rešenje

Upotrebom backtracking matrica će biti kompletirana u spirali, tako da se u svakom koraku prate 2 parametra: broji koji sledi iz 1-2-3 niza (zavisno šta se uradilo poslednji put sa brojem) i 0. Ako se krećemo dijagonalom, onda menjamo smer i u svakoj promeni smera, vrsta ili kolona su već kompletirani, tako da možemo da radimo delimične provere kako bi poboljšali strategiju. Čim se nađe prvo rešenje, štampa se i prekida rad programa.

No, da bi mogao da se uklopi u vremensko ograničenje 1s, moralo je drukčije da se razmišlja. Potrebno je ručno ili backtrackingom obraditi početne kvadrate za male slučajeve za n = 5, 6, 7, 8, 9, 10. Kad je n>10 onda ćemo za konačno rešenje koristiti (n-6)x(n-6) puž matricu 123 i 6 x 6 puž matricu 123 izdeljene na četiri 3x3 podmatrice kao na slici.

1	2	0	3	0	0
3	0	1	0	2	0
0	0	0	2	3	1
0	1	3	0	0	2
0	3	2	1	0	0
2	0	0	0	1	3

Sa ove 4 podmatrice mogu da se kompletiraju cetiri ugaona fragmenta u (n-6)x(n-6) puz matrici 123 i dobijamo ono sto nam treba, traena puz matrica 123.

Na primer, trazena matrica za n=11 se dobija postupkom kao na slici

1	2	0						3	0	0
3	0	1						0	2	0
0	0	0						2	3	1
			0	1	0	2	3			
			0	2	3	0	1			
			1	3	0	0	2			
			3	0	2	1	0			
			2	0	1	3	0			
0	1	3						0	0	2
0	3	2						1	0	0
2	0	0						0	1	3

7. Svaki uspešan radnik u uspešnoj programerskoj firmi će biti nagrađen za predstojeći Božić. Naime, šef Vlada će svakom nagrađenom radniku pokloniti zlatnik u specijalno obojenoj kutiji oblika kocke.

Međutim, nakon što je šef Vlada primio kutije iz lokalne prodavnice, otkrio je da nisu sve jednako obojene. Tako da, Vaš zadatak je da pomognete prodavcu da izračuna minimalni broj strana kocki koje se moraju obojiti iznova kako bi sve kutije za nagradu jednako izgledale (ili šef Vlada neće platiti kutije). Dve kutije izgledaju jednako, ako nakon što su rotirane na odgovarajući način, imaju iste boje na odgovarajućim stranama. Drugim rečima, dve kutije su jednake *ako i samo ako* možete rotirati dve kutije tako da boja prednje strane prve kutije je jednaka boji prednje strane druge kutije, boja leve strane prve kutije je jednaka boji leve strane druge kutije, i tako dalje.

Napisati C program koji sa standardnog ulaza učitava iz prve linije ulaza ceo broj N ($2 \leq N \leq 100$) - broj kutija za medalje. Potom iz svake od sledećih N linija učitati 6 celih brojeva F_{i1}, F_{i2}, F_{i3}, F_{i4}, F_{i5} i F_{i6} ($0 \leq F_{i1}, F_{i2}, F_{i3}, F_{i4}, F_{i5}, F_{i6} \leq 6$) – originalne boje za svih 6 strana i-te kutije za medalju. Redosled ovih 6 celih brojeva koji opisuju boje kutije je zadat kao: prednja strana (F_{i1}), gornja strana (F_{i2}), zadnja strana (F_{i3}), donja strana (F_{i4}), leva strana (F_{i5}) i desna strana (F_{i6}).

Vaš program treba da ispiše minimalni broj strana koje se moraju obojiti iznova kako bi sve kutije za medalju izgledale isto.

Proceniti vremensku i prostornu složenost rešenja.

Napomene:

1. U nekim test primerime, optimalno rešenje će biti da se oboje sve strane svih kutija istom bojom.

2. Voditi računa da Vaš program ne premaši vremensko ograničenje od 1 sekunde i memorijsko ograničenje od 64 megabajta

Primer 1

Ulaz	izlaz
2 3 1 4 2 5 6 5 1 6 2 4 3	0

Primer 2

Ulaz	Izlaz
7 1 5 2 5 1 0 5 1 6 0 4 4 5 1 2 5 5 4 4 3 6 1 3 4 4 5 0 0 5 1 5 3 4 3 4 1 3 4 5 5 1 4	16

Objašnjenje za primer 1: Dve kutije za medalju izgledaju isto (druga kutija izgleda isto kao prva, ali je rotirana jednom udesno).

Rešenje