Clasa a XI-a B

Soluţii Teză Semestrul I

Se acordă 2 puncte din oficiu

Partea I. Backtracking - 3 puncte(6*0.5 puncte)

1. Folosind tehnica bactracking un elev a scris un program care generează toate numerele de câte n cifre ($0<n\leq 9$), cifrele fiind în ordine strict crescătoare. Dacă n este egal cu 5 , scrieți în ordine crescătoare toate numerele având cifra unităților 6 , care vor fi generate de program.
Răspuns: 12346, 12356, 12456, 13456, 23456

2.Utilizând metoda backtracking se generează permutările cuvântului INFO. Dacă primele 3 soluții generate sunt : FINO, FION, FNIO care este a 5-a soluție ?
Răspuns: FNOI -> a 5-a soluție

3. Se utilizează un algoritm pentru a genera în ordine lexicografică inversă toate permutările mulţimii { 1 , 2 , 3 , 4 , 5 }. Primele patru permutări generate sunt: 54321 , 54312 , 54231 , 54213. Care este a 5-a soluție ?
Răspuns: 54132

4. Utilizând metoda backtracking se generează toate permutările mulţimii {1,2,3,4} . Dacă primele trei permutări generate sunt, în această ordine: 1234, 1243, 1324 precizaţi care este permutarea generată imediat după 3412 .
Răspuns: 3421

5. Utilizând metoda backtracking se generează toate cuvintele de câte 3 litere din mulţimea {a,b,c} . Dacă primele patru cuvinte generate sunt, în acestă ordine: aaa, aab, aac, aba, care este cel de-al optulea cuvânt generat?
Răspuns: acb

6. Se utilizează metoda backtracking pentru a genera toate cuvintele de câte patru litere distincte din mulţimea {d,a,n,s} . Ştiind că al doilea cuvânt generat este dans , iar al treilea este dsan , care va fi ultimul cuvânt obţinut?
Răspuns: dasn

Partea II. Greedy - 2 puncte

1. Se introduce de la tastatură numărul natural k > 1. Se cere să se afişeze la ecran cel mai mic număr natural n având exact k divizori naturali proprii (diferiți de 1 si n).

```
#include <iostream>
using namespace std;

int n,i,k,gata,v;

int verif(int n,int k)
{
     int i=0,j;
     for(j=2;j<=n-1;j++)
          if(n%j==0) i++;
     if(i==k) v=1;
     else v=0;
     return v;
}

int main()
{
     cout<<"Dati numarul de divizori k > 1: ";
     cin>>k;
     cout<<endl<<"Cel mai mic numar care are exact
"<<k<<"divizori este "<<endl;
     n=k+2;
     gata =0;
     while(gata==0)
     {
          v=verif(n,k);
          if(v==1)
          {
               cout<<n;
               gata=1;
          }
          n++;
     }
     return 0;
}
```

Clasa a XI-a B

Teză Semestrul I

Partea III Grafuri neorientate 3 puncte (2*1,5 puncte)

1. Se dă lista muchiilor unui graf neorientat. Să se afișeze vârfurile izolate ale grafului.

Date de intrare

Fişierul de intrare ***grade.in*** conţine pe prima linie numărul n, reprezentând numărul de vârfuri ale grafului. Fiecare dintre următoarele linii conține câte o pereche de numere i j, cu semnificația că există muchie între i şi j.

Date de ieşire

Fişierul de ieşire ***grade.out*** va conţine pe prima linie n numere naturale, reprezentând gradele vârfurilor, în ordinea vârfurilor.

Restricţii şi precizări

$1 \leq n \leq 100$

$1 \leq i , j \leq n$

muchiile se pot repeta în fișierul de intrare dar se va considera una singură

Exemplu 1

grade.in

5
1 4
1 3
3 5
4 5
2 4
1 2
4 2
3 4

grade.out

Varfuri izolate: nu sunt

Exemplu 2

grade.in

6
1 4
1 3
3 5
4 5
2 4
1 2
4 2
3 4

grade.out

Varfuri izolate: 6

Solutia 1. Utilizam matricea de adiacenta

```
#include <iostream>
#include <fstream>

using namespace std;

ifstream fin("grade.in");
ofstream fout("grade.out");

int a[20][20], n;

void CreareMA()
{
    int i, j;
    fin>>n;
    while(fin>>i>>j)
        a[i][j]=a[j][i]=1;
}
```

```
void AfisareMA()
{
     int i, j;
     for(i=1;i<=n;i++)
     {
          cout<<endl;
          for(j=1;j<=n;j++)
               cout<<a[i][j]<<" ";
     }


}

int Izolat(int k)
{
     int i, g=0, este=0;
     for(i=1;i<=n;i++)
          g=g+a[k][i];
     if(g==0)
          este=1;
     return este;
}

int main()
{
     int i, nr=0;
     CreareMA();
     cout<<"Matricea de adiacenta"<<endl;
     AfisareMA();
     fout<<"Varfuri izolate: ";
     for(i=1;i<=n;i++)
          if(Izolat(i)==1)
          {
               fout<<i<<" ";
               nr++;
          }
     if(nr==0)
          fout<<"nu sunt!";

     return 0;
}
```

Solutia 2. Utilizam vectorul de grade

```
#include <iostream>
#include <fstream>

using namespace std;

ifstream fin("grade.in");
```

```
ofstream fout("grade.out");

int d[20], n;

void Grade()
{
    int i, j;
    fin>>n;
    while(fin>>i>>j)
    {
        d[i]++; d[j]++;
    }

}


int main()
{
    int i, nr=0;
    Grade();

    fout<<"Varfuri izolate: ";
    for(i=1;i<=n;i++)
        if(d[i]==0)
        {
            fout<<i<<" ";
            nr++;
        }
    if(nr==0)
        fout<<"nu sunt!";

    return 0;
}
```

2. Se dă lista muchiilor unui graf neorientat. Să se afișeze vârfurile de grad maxim.

Date de intrare

Fişierul de intrare ***gradmax.in*** conţine pe prima linie numărul n, reprezentând numărul de vârfuri ale grafului. Fiecare dintre următoarele linii conține câte o pereche de numere i j, cu semnificația că există muchie între i și j.

Date de ieşire

Fişierul de ieşire ***gradmax.out*** va conţine pe prima linie numărul m de vârfuri de grad maxim, urmat de cele m vârfuri de grad maxim ,în ordine crescătoare, separate prin exact un spațiu.

Restricţii şi precizări

$1 \leq n \leq 100$

$1 \leq i, j \leq n$

muchiile se pot repeta în fișierul de intrare dar se va considera una singură

Exemplu

gradmax.in

5

1 4

2 5

2 3
2 1
4 5
3 2
4 3

gradmax.out
2

O solutie.

```
#include <iostream>
#include <fstream>

using namespace std;

ifstream fin("gradmax.in");
ofstream fout("gradmax.out");

int d[20], n, maxim;

void Grade()
{
     int i, j;
     fin>>n;
     while(fin>>i>>j)
     {
          d[i]++; d[j]++;
     }

}

int main()
{
     int i;
     Grade();
     for(i=1;i<=n;i++)
          if(d[i]>maxim)
               maxim=d[i];
     cout<<"Gradele nodurilor= ";
     for(i=1;i<=n;i++)
          cout<<d[i]<<" ";
     cout<<"\nGrad maxim= "<<maxim;
     for(i=1;i<=n;i++)
          if(d[i]==maxim)
```

```
                fout<<i<<" ";

    return 0;
}
```