Suprafețe și regiuni Steiner

Autori: **Ion Alexandru Gabriel, Teodor Dragomir, clasa a XI-a I**
Colegiul Național de Informatică "*Tudor Vianu*", București
coord. prof. **Carmen Mincă**

Abstract

Articolul prezintă o problemă celebră propusă și rezolvată de Jakob Steiner (1796 – 1863), un matematician elvețian care a adus importante contribuții în domeniul geometriei și al mecanicii. Problema a fost publicată pentru prima dată în Crelle's Journale (1826). La final, articolul conține implementările în limbajul C++ a soluției problemei lui Steiner varianta în plan, respectiv în spațiu.

Cuprins articol

1. Prezentarea problemei lui Jakob Steiner

*Definim o **regiune** ca fiind o porțiune tridimensională (din spațiu) delimitată de unul sau mai multe plane (de exemplu, un plan împarte spațiul în două regiuni).*

*Definim o **suprafață** ca fiind o porțiune bidimensională (din plan) delimitată de una sau mai multe drepte (de exemplu, o dreaptă împarte planul în două suprafețe).*

Enunțul problemei lui Steiner:

*Care este numărul maxim de regiuni din spațiu ce se pot obține prin intersecțiile a **n** plane?*

1.1. Rezolvarea problemei în plan

Pentru a rezolva problema în spațiul tridimensional și pentru a o înțelege mai bine, o vom reduce la un caz mai simplu, în planul bidimensional.

În plan, enunțul problemei devine: ***Care este numărul maxim de suprafețe din plan obținute prin intersecțiile a n drepte?***

De exemplu, pentru **n=3** drepte, se pot obține, prin intersecțiile celor trei drepte, cel mult **7** suprafețe (vezi Figura 1).

Observăm că, pentru a obține un număr maxim de suprafețe, dreptele vor fi poziționate astfel încât:

- oricare două drepte din cele **n** să nu fie paralele;
- oricare trei drepte din cele **n** să nu se intersecteze în același punct.

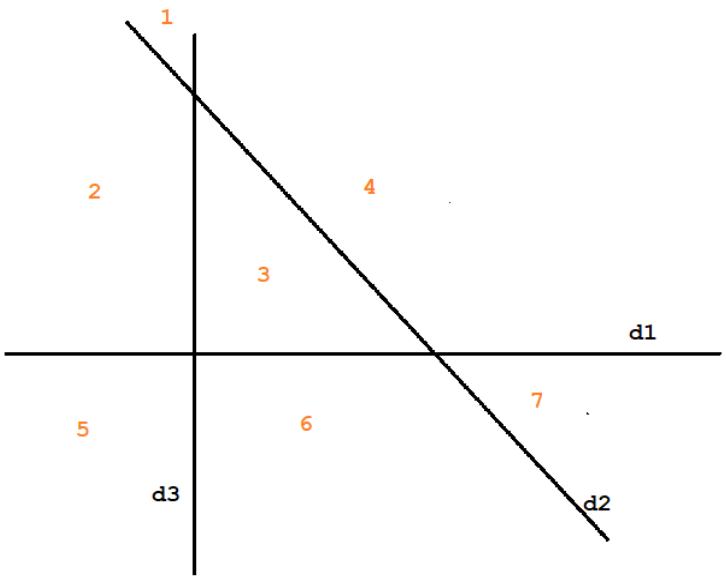

Figura 1

Fie șirul a_n = numărul maxim de suprafețe din plan separate de cele **n** drepte. Evident, $a_0 = \mathbf{1}, a_1 = \mathbf{2}$, $a_2 = \mathbf{4}$ și $a_3 = \mathbf{7}$.

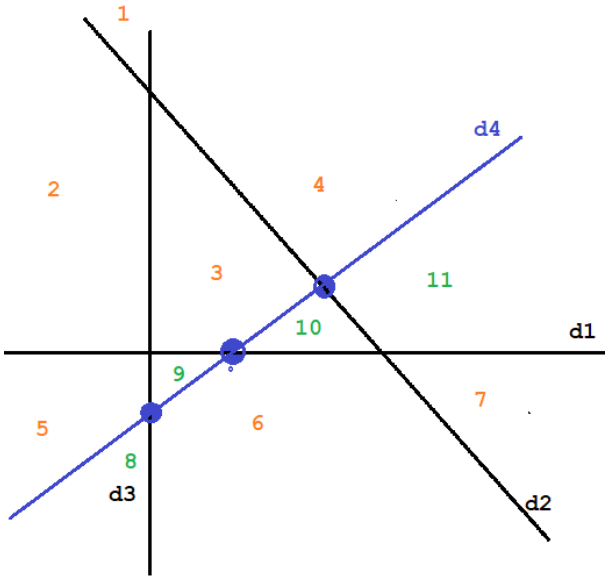

Figura 2

Fie **n-1** drepte așezate în plan astfel încât să determine numărul a_{n-1} maxim de suprafețe. Atunci, o a **n**-a dreaptă poate intersecta celelalte **n-1** drepte în cel mult **n-1** puncte, determinând **n** porțiuni din dreaptă care vor contribui cu câte o nouă suprafață.

De exemplu, **Figura 2** exemplifică intersecția celor **3** drepte cu o a **4**-a dreaptă astfel încât să rezulte un număr maxim de suprafețe. Suprafețele noi sunt numerotate cu **8**, **9**, **10** și **11**.

Șirul a_n este un șir recurent de ordinul **1**, având formula: $a_n = a_{n-1} + n$. Scriem în coloană primii *n* termeni ai șirului cu ajutorul formulei de recurență, și apoi adunăm valorile din fiecare coloană:

$a_0 = 1$
$a_1 = a_0 + 1 = 2$
$a_2 = a_1 + 2 = 4$
$a_3 = a_2 + 3 = 7$
…
$a_n = a_{n-1} + n$

$a_0 + a_1 + a_2 + a_3 + \ldots + a_{n-1} + a_n = 1 + a_0 + 1 + a_1 + 2 + a_2 + 3 + \ldots + a_{n-1} + n$

Efectuând simplificările, obținem formula generală a termenilor șirului $\mathbf{a_n}$ reprezentând soluția problemei în plan.

$$a_n = 1 + 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2} + 1 = \frac{n^2+n+2}{2} = \frac{n^2+n}{2} + 1$$

1.2. Rezolvare problemei în spațiu

Enunț: *Care este numărul maxim de regiuni din spațiu ce se pot obține prin intersecțiile a* ***n*** *plane?*

Pentru **n=2** plane, se pot obține cel mult **4** regiuni din spațiu (vezi Figura 3).

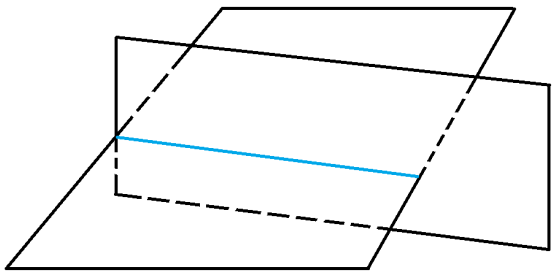

Figura 3

Pentru **n=3** plane, se pot obține cel mult **8** regiuni din spațiu (vezi Figura 4).

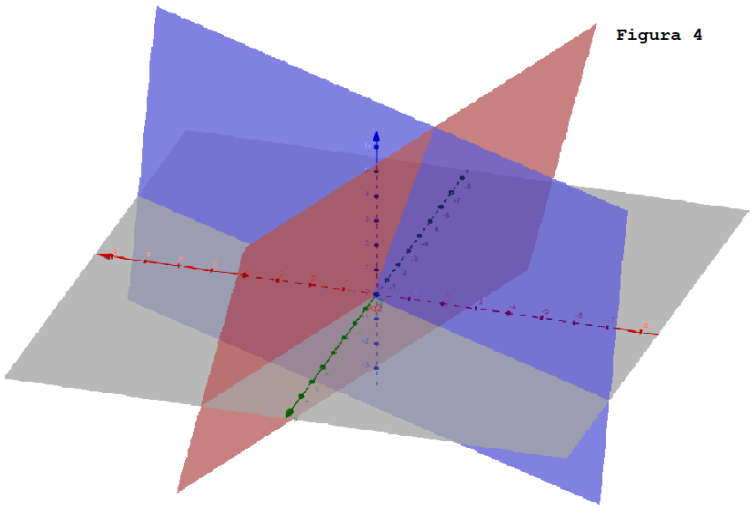

Figura 4

Fie r_n numărul maxim de regiuni din spațiu care se pot forma prin intersecțiile a **n** plane. Pentru a obține un număr maxim de regiuni, planele vor fi poziționate astfel încât:

- oricare două plane nu pot fi paralele;
- cel mult trei plane se pot întâlni în același punct (se evită cazul în care trei plane se intersectează formând împreună o structură asemănătoare colțului unei camere);
- dreptele de intersecție a oricăror două perechi de plane nu pot fi paralele.

Relația de recurență caracteristică acestei problemei este: $r_n = r_{n-1} + a_{n-1}$, unde a_n este șirul de la problema în plan anterioară. Raționamentul care a condus la această recurență este legat de raționamentul problemei în plan: planul **n** va intersecta celelalte **n-1** plane astfel încât acesta va conține **n-1** drepte de intersecție neparalele două câte două si neconcurente oricare trei dintre ele. Fiecare dintre cele a_{n-1} suprafețe determinate indică formarea unui spațiu suplimentar.

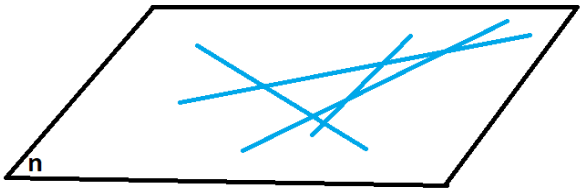

Figura 5

Pornind de la formula de recurență a șirului r_n, se obține formula termenului general al șirului: $r_n = 2 + a_1 + a_2 + a_3 + \ldots + a_{n-1}$, unde $a_k = \frac{k^2+k}{2} + 1$ =>

$$r_n = 2 + \sum_{k=1}^{n-1}\left(\frac{k^2+k}{2}+1\right) = n + 1 + \frac{1}{2}\cdot\left(\sum_{k=1}^{n-1} k^2 + \sum_{k=1}^{n-1} k\right) = n + 1 + \frac{1}{2}\cdot\left(\frac{(n-1)\cdot n\cdot(2n-1)}{6} + \frac{(n-1)\cdot n}{2}\right)$$

Obținem:

$$r_n = n + 1 + \frac{(n-1)\cdot n\cdot(n+1)}{6} = \frac{n^3+5\cdot n+6}{6}$$

2. Aplicare în informatică – implementare soluții în limbajul C++

O soluție poate fi un algoritm care să construiască un tablou unidimensional ale cărui elemente memorează termenii șirului a_n, respectiv r_n, pe baza formulelor de recurență descrise în acest articol.

2.1. Implementarea soluției problemei din plan

Pentru problema în plan, propunem următorul algoritm care utilizează formula de recurență $a_n = a_{n-1} + n$, construiește un tablou unidimensional cu n elemente și are complexitatea de timp și spațiu **O(n),** $1 \le n \le 100000$. (**Figura 6**).

```cpp
#include <iostream>

using namespace std;

long long Steiner[100005];
int n;

int main()
{
cin>>n;

Steiner [1]=2, Steiner[2]=4;
// O dreapta poate imparti planul in 2 semiplane, iar 2 in 4

for(int i=3; i<=n; i++)
    Steiner[i]=Steiner[i-1]+i;
// Steiner[i] este numarul de regiuni ce pot fi
//  delimitate prin intersectia a i drepte

cout<<Steiner[n];
return 0;
}
```

Figura 6

O soluție eficientă (complexitate de timp și spațiu **O(1)**), care nu necesită utilizarea tipurilor de date structurate, constă în implementarea formulei termenului general al șirului:

$$a_n = \frac{n^2+n}{2} + 1,\ 1 \le n \le 2^{36}$$

(**Figura 7**).

```cpp
#include <iostream>

using namespace std;

int n;

int main()
{
cin >> n;

cout << n*(n+1)/2 + 1;
// Folosim formula calculata anterior

return 0;
}
```

Figura 7

2.2. Implementarea soluției problemei în spațiu

Pentru problema în spațiu, propunem următoarele două modalități de rezolvare:

(1) pornind de la formula de recurență $r_n = r_{n-1} + a_{n-1}$, complexitate timp de executare și spațiu de memorare **O(n)** (se utilizează un tablou unidimensional ale cărui componente memoreaza primii **n** termeni ai șirului) **- Figura 8;**

Observație: Dacă nu dorim să memorăm valorile primilor **n** termeni ai șirului, putem rezolva problema folosind două variabile simple în locul tablourilor unidimensionale (algoritmul rezultat va avea o complexitate de timp **O(n)** și o complexitate de spațiu **O(1)**).

```cpp
#include <iostream>

using namespace std;

long long Steiner[100005];
long long Steiner_spatiu[100005];
int n;

int main()
{
cin>>n;

Steiner[1] = Steiner_spatiu[1] = 2;
// O dreapta poate imparti planul in 2 semiplane, iar 2 in 4
Steiner[2] = Steiner_spatiu[2] = 4;
// Un plan poate imparti spatiul in 2 semispatii, iar 2 in 4

for(int i=3; i<=n; i++)
    Steiner[i]=Steiner[i-1]+i;
// Calculam vectorul pentru problema in plan

for(int i=3; i<=n; i++)
    Steiner_spatiu[i] = Steiner_spatiu[i-1] + Steiner[i-1];
// Steiner_spatiu[i] este numarul de regiuni de spatiu
//   ce pot fi obtinute prin intersectia a i plane

cout<<Steiner_spatiu[n];
return 0;
}
```

Figura 8

(2) pornind de la formula termenului general $r_n = \frac{(n-1)\cdot n\cdot(n+1)}{6} + n + 1$, complexitate de timp și spațiu **O(1) (Figura 9)**

```cpp
#include <iostream>

using namespace std;

int n;

int main()
{
cin >> n;

cout << (n-1)*n*(n+1)/6 + n + 1;
// Folosim formula calculata anterior

return 0;
}
```

Figura 9

3. Concluzie

Soluțiile celor două variante (în plan și în spațiu) ale problemei matematicianului Jakob Steiner pot fi implementate în limbajul C++ în trei variante de complexități diferite:

(a) complexitate ca timp de executare **O(n)** și complexitate ca spațiu de memorare **O(n)** prin utilizarea formulelor de recurență și a tablourilor unidimensionale care să memoreze primii **n** termeni ai șirurilor $\mathbf{a_n}$ și $\mathbf{r_n}$ – **Figura 6/Figura 8**;

(b) complexitate ca timp de executare **O(n)** și complexitate ca spațiu de memorare **O(1)** prin utilizarea formulelor de recurență și a variabilelor simple, în locul tablourilor, care să memoreze succesiv termenii șirurilor $\mathbf{a_n}$ și $\mathbf{r_n}$;

(c) complexitate ca timp de executare **O(1)** și complexitate ca spațiu de memorare **O(1)** prin utilizarea formulelor termenilor generali și a unui număr redus de variabile simple) – **Figura 7/Figura 9**

Implementările în limbajul C++ (a) și (b) pot fi utilizate pentru un număr de regiuni mai mic sau cel mult egal cu $\mathbf{10^6}$, iar implementerea (c) poate fi utilizată pentru valori ale lui **n** care nu depășesc, în aplicarea formulelor termenilor generali, limitele tipului de date **long int** specific limbajului C++.

4. Bibliografie

1. Steiner's regions of space problem | Famous Math Problems 11 | NJ Wildberger – YouTube

2. https://ro.wikipedia.org/wiki/Jakob_Steiner

3. https://faculty.evansville.edu/ck6/bstud/steiner.html (mai multe despre Steiner)