

Geometria Divizibilității și Algoritmii Tabelului Parascan-Margoș

1. Arhitectura și Conceptele Fundamentale ale Tabelului

Tabelul Parascan-Margoș nu este o simplă grilă de reprezentare numerică, ci o matrice de reprezentare a operatorului modulo, configurată ca un instrument de cercetare vizuală pentru maparea relațiilor de divizibilitate. Importanța sa strategică rezidă în capacitatea de a vizualiza setul de perechi (c, r) care satisfac condiția de divizibilitate exactă, transformând proprietățile aritmetice abstracte în structuri geometrice rigide.

Sistemul permite ajustarea dimensiunii grilei (până la un ordin de mărime de 500), facilitare critică pentru analiza comportamentului asimptotic al densității divizorilor și identificarea regularităților la scări macroscopice. Interacțiunea dintre variabilele fundamentale este sintetizată în tabelul tehnic de mai jos:

Variabilă	Semnificație Tehnică și Matematică
Rând (r)	Reprezintă divizorul potențial în cadrul operației de calcul.
Coloană (c)	Reprezintă numărul examinat (deîmpărțitul); axa de referință.
Celulă Activă	Reprezentarea vizuală a mulțimii perechilor (c, r) pentru care $c \equiv 0 \pmod r$.
Celulă Goală	Absența relației de divizibilitate ($c \not\equiv 0 \pmod r$); spațiu neutru.

Această arhitectură carteziană stabilă constituie fundamentul dualității de vizualizare, permițând cercetătorului să penduleze între rigoarea calculului numeric și topologia pură a modelelor rezultate.

2. Dualitatea Reprezentării: Modul Divizibilitate vs. Modul Fractal

Schimbarea perspectivei de la calculul valoric la geometria repetitivă este esențială pentru detectarea structurilor de auto-similaritate care guvernează distribuția numerelor. Tabelul Parascan-Margoș operează prin două moduri distincte de procesare a datelor:

- **Modul Divizibilitate (Focalizare pe Calcul):** În acest mod, celulele active expun valoarea câtului (c/r). Primul rând ($r=1$) funcționează ca o „axă a identității”, reflectând valoarea însăși a coloanei, în timp ce diagonala ($r=c$) afișează invariabil unitatea (1). Acest mod transformă grila într-un calculator matricial de factori, evidențiind ponderea valorică a fiecărui divizor.
- **Modul Fractal (Focalizare pe Model Vizual):** Prin utilizarea constantei „1” pentru toate celulele active, sistemul elimină ponderea valorică a câtului, lăsând să apară topologia pură a divizibilității. Această uniformizare vizuală permite identificarea unor structuri de tip „Sierpinski-like” — pattern-uri triunghiulare repetitive și simetrice care reapar la diferite niveluri de magnitudine.

Această dualitate este crucială („So What?” layer) deoarece facilitează detectarea tiparelor ascunde: transformarea abstractului în vizual permite ochiului uman și algoritmilor de pattern-matching să identifice ordinea acolo unde calculul liniar ar vedea doar o succesiune de resturi. Elementul de legătură între cele două moduri este izolarea numerelor prime, care își păstrează poziția și semnificația indiferent de modul de calcul aplicat.

3. Analiza Algoritmilor de Ierarhizare și Sortare

Algoritmii de ierarhizare nu sunt simple instrumente de ordonare estetică, ci mecanisme de reorganizare a grilei bazate pe densitatea divizibilității, menite să expună proprietățile de grup ale numerelor compuse și prime.

- **Ierarhizarea Orizontală și Verticală:** Această sortare are la bază cardinalitatea setului de divizori, notată prin funcția $d(n)$. Ierarhizarea orizontală reorganizează coloanele în funcție de numărul total de divizori, grupând numerele „simple” (primele, cu $d(n)=2$) la originea axei. Ierarhizarea verticală aplică același principiu rândurilor, modificând ordinea divizorilor pentru a evidenția aglomerările de factori.
- **Ierarhizarea Ciclu 6:** Bazată pe logica Modulo 6, acest algoritm funcționează ca un filtru determinist care izolează numerele de forma $6k \pm 1$. Prin gruparea resturilor 1 și 5 și prioritizarea numerelor prime în cadrul fiecărui grup, sistemul reduce spațiul de căutare pentru primalitate cu aproximativ 66.6%, eliminând redundant multiplii de 2 și 3.

O limitare tehnică inerentă sistemului actual este imposibilitatea aplicării simultane a ierarhizării orizontale și verticale; activarea uneia resetează cealaltă axă la ordinea naturală. Totuși, impactul acestor sortări este major, permițând vizualizarea tranziției de la structuri „sparse” (numere prime) la structuri „dense” (numere compuse înalt divizibile).

4. Tehnica Izolării Numerelor Prime: „Starea Fundamentală”

Numerele prime reprezintă „cărămizile” elementare ale sistemului numeric, iar izolarea lor cromatică în portocaliu nu este o alegere estetică, ci una funcțională. În arhitectura tabelului, primalitatea este

confirmată vizual prin punctele de referință ale **Diagonalei** ($r=c$) și ale **Primului Rând** ($r=1$).

Un număr este identificat în „Starea Fundamentală” dacă prezintă celule active exclusiv în aceste două puncte, cu absența totală a divizorilor intermediari în intervalul deschis $(1, c)$. Această configurație reprezintă o formă de invarianță topologică: numerele prime sunt singurele care păstrează o amprentă binară (două puncte active) indiferent de permutările rândurilor sau coloanelor aplicate prin algoritmi de ierarhizare.

Această izolare permite cercetătorului să elimine „zgomotul” computațional — definit ca densitatea divizorilor proprii d în $(1, c)$ caracteristici numerelor compuse. Stabilitatea vizuală a portocaliului în ambele moduri de lucru (Divizibilitate și Fractal) transformă tabelul într-un instrument de înaltă precizie pentru observarea modului în care aceste „ancore” numerice modelează geometria fractală adiacentă.

5. Implicații în Securitatea Datelor și Analiza Simetriei

Maparea densității divizibilității prin Tabelul Parascan-Margoș oferă perspective valoroase în analiza vulnerabilităților de factorizare, pilonul central al criptografiei moderne (ex: algoritmul RSA). Vizualizarea aglomerărilor de divizori prin ierarhizarea orizontală permite detectarea directă a numerelor „smooth” (numere compuse cu factori mici), care constituie primele ținte în atacurile de factorizare.

Potențialul sistemului se extinde către:

- **Dezvoltarea funcțiilor hash:** Utilizarea pattern-urilor fractale complexe și a auto-similarității pentru a crea amprente digitale greu de inversat.
- **Criptografie geometrică:** Generarea de chei bazate pe configurații simetrice ale divizibilității, mai degrabă decât pe simple secvențe numerice.

- **Optimizarea Ciclu 6:** Studiul distribuției numerelor prime $6k \pm 1$ pentru eficientizarea proceselor de generare a cheilor criptografice robuste.

În concluzie, Tabelul Parascan-Margoș funcționează ca o „hartă interactivă” a sistemului numeric, transformând complexitatea abstractă a divizibilității într-o structură geometrică inteligibilă, predictibilă și vizual stabilă, oferind un avantaj analitic decisiv în studiul teoriei numerelor și al securității informației.

```
<!DOCTYPE html>
<html lang="ro">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Tabel Fractal Parascan-Margoș</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/html2canvas/1.4.1/html2can
vas.min.js"></script>
  <style>
    .cell {
      width: 20px;
      height: 20px;
      display: flex;
      align-items: center;
      justify-content: center;
      font-size: 8px;
      border: 0.1px solid #f3f4f6;
    }
    .active {
      background-color: #1f2937;
      color: white;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <div class="grid grid-cols-10 gap-1">
    <div class="cell active">1</div>
    <div class="cell">2</div>
    <div class="cell">3</div>
    <div class="cell">4</div>
    <div class="cell">5</div>
    <div class="cell">6</div>
    <div class="cell">7</div>
    <div class="cell">8</div>
    <div class="cell">9</div>
    <div class="cell">10</div>
  </div>
</body>
</html>
```

```

    .prime-highlight {
prime */      background-color: #f97316; /* Portocaliu pentru numere
              color: white;
              font-weight: bold;
            }
    .empty {
              background-color: #ffffff;
              color: #e5e7eb;
            }
  </style>
</head>
<body class="bg-gray-100 p-4">
  <div class="max-w-7xl mx-auto">
    <h1 class="text-2xl font-bold mb-4 text-gray-800">Tabel Fractal
Parascan-Margoș</h1>

    <div class="mb-4 flex flex-wrap items-center gap-2">
      <input type="number" id="sizeInput" value="25" min="1"
max="500" class="w-20 px-2 py-1 border rounded text-sm">
      <button onclick="applyCustomSize()" class="px-3 py-1 bg-
gray-600 text-white rounded text-sm">Aplică Dimensiune</button>
      <button onclick="resetTable()" class="px-3 py-1 bg-gray-800
text-white rounded text-sm">Reset</button>
      <button onclick="renderTable('divisibility')" class="px-3 py-1
bg-blue-600 text-white rounded text-sm">Divizibilitate</button>
      <button onclick="renderTable('fractal')" class="px-3 py-1 bg-
green-600 text-white rounded text-sm">Fractal</button>
      <button onclick="renderSortedTable('horizontal')" class="px-3
py-1 bg-purple-600 text-white rounded text-sm">Ierarhizare
Orizontală</button>
      <button onclick="renderSortedTable('vertical')" class="px-3
py-1 bg-indigo-600 text-white rounded text-sm">Ierarhizare
Verticală</button>
      <button onclick="renderCycle6()" class="px-3 py-1 bg-
yellow-600 text-white rounded text-sm">Ierarhizare Ciclu 6</button>

```

```
    <button onclick="saveAsJPG()" class="px-3 py-1 bg-red-600
text-white rounded text-sm">Salvare JPG</button>
  </div>
```

```
  <div id="container" class="bg-white p-2 rounded shadow
overflow-auto" style="max-height: 80vh;">
    <div id="table-grid" class="inline-grid">
      <!-- Celulele generate dinamic -->
    </div>
  </div>
</div>
```

```
<script>
  let currentSize = 25;

  function resetTable() {
    currentSize = 25;
    document.getElementById('sizeInput').value = 25;
    document.getElementById('table-
grid').style.gridTemplateColumns = `repeat(25, 20px)`;
    renderTable('divisibility');
  }

  function applyCustomSize() {
    const input = document.getElementById('sizeInput');
    let val = parseInt(input.value);
    if (isNaN(val) || val < 1) val = 1;
    if (val > 500) val = 500;
    input.value = val;
    currentSize = val;
    document.getElementById('table-
grid').style.gridTemplateColumns = `repeat(${val}, 20px)`;
    renderTable('divisibility');
  }

  function isPrime(n) {
    if (n < 2) return false;
```

```
    for (let i = 2; i <= Math.sqrt(n); i++) if (n % i === 0) return
false;
    return true;
}
```

```
function renderCycle6() {
  const range = Array.from({length: currentSize}, (_, i) => i +
1);
  const sorted = range.sort((a, b) => {
    const restA = a % 6;
    const restB = b % 6;
    if (restA !== restB) return restA - restB;
    return (isPrime(a) ? 0 : 1) - (isPrime(b) ? 0 : 1);
  });
  renderTable('divisibility', sorted, null);
}
```

```
function renderTable(type, colOrder = null, rowOrder = null) {
  const grid = document.getElementById('table-grid');
  grid.innerHTML = "";

  const cols = colOrder || Array.from({length: currentSize}, (_,
i) => i + 1);
  const rows = rowOrder || Array.from({length: currentSize}, (_,
i) => i + 1);
```

```
  const fragment = document.createDocumentFragment();
```

```
  for (let i = 0; i < currentSize; i++) {
    for (let j = 0; j < currentSize; j++) {
      const r = rows[i];
      const c = cols[j];
      const cell = document.createElement('div');
      cell.className = 'cell';
```

```
      let value = 0;
      if (type === 'divisibility') {
```

```

    if (r === c) value = 1;
    else if (r === 1) value = c;
    else if (c % r === 0) value = c / r;
    else value = 0;
  } else {
    if (r === 1) value = 1;
    else if (r === c) value = 1;
    else if (c > r && c % r === 0) value = 1;
    else value = 0;
  }
}

```

// Logica de marcare: primele primesc portocaliu (starea fundamentală)

```

const isCellPrime = isPrime(c);
if (isCellPrime && (r === 1 || r === c)) {
  cell.classList.add('prime-highlight');
  if (currentSize <= 40) cell.textContent = value;
} else if (value !== 0) {
  cell.classList.add('active');
  if (currentSize <= 40) cell.textContent = value;
} else {
  cell.classList.add('empty');
}
}
fragment.appendChild(cell);
}
}
grid.appendChild(fragment);
}

```

```

function renderSortedTable(axis) {
  const range = Array.from({length: currentSize}, (_, i) => i +
1);

  function countDivisors(n) {
    let count = 0;
    for (let i = 1; i <= n; i++) if (n % i === 0) count++;
    return count;
  }
}

```

```

    const sorted = range.sort((a, b) => countDivisors(a) -
countDivisors(b));

    if (axis === 'horizontal') renderTable('divisibility', sorted,
null);
    else renderTable('divisibility', null, sorted);
  }

function saveAsJPG() {
  const container = document.getElementById('table-grid');
  html2canvas(container).then(canvas => {
    const link = document.createElement('a');
    link.download = 'tabel-fractal.jpg';
    link.href = canvas.toDataURL('image/jpeg');
    link.click();
  });
}

document.getElementById('table-
grid').style.gridTemplateColumns = `repeat(${currentSize}, 20px)`;
renderTable('divisibility');
</script>
</body>
</html>

```

















