

```

ClearAll["Global`*"];
SeedRandom[2026]; (* 设定随机种子Or you can leave this seed blank. *)

(* ===== *)
(* Part 1: 物理定义 (手性与极性) *)
(* ===== *)

nodePolarity = <||>;
SetPolarity[n_, p_] := (nodePolarity[n] = p);
GetPolarity[n_] := Lookup[nodePolarity, n, 0];

(* ===== *)
(* Part 2: 核心演化引擎 (修复自环崩溃版) *)
(* ===== *)

NaturalEvolutionStep[g_Graph] := Module[{
  edges, activeEdges, candidates,
  e1, e2, x, y, z, w,
  pX, pY, pZ, isHetero,
  neighbors, energyRelease,
  newActive, newFrozen, newG,
  reactionType
},

edges = EdgeList[g];

(* --- FIX: 严格过滤自环 (u≠v), 防止 Complement 返回空集 --- *)
activeEdges = Cases[edges, UndirectedEdge[u_, v_] /; u != v];

(* --- A. 随机寻找反应对 --- *)
candidates = {};
Module[{shuffled = RandomSample[activeEdges, Min[Length[activeEdges], 100]], eA, eB},
Label[];
Do[
eA = shuffled[[i]];
Do[
eB = shuffled[[j]];
If[Length[Intersection[List @@ eA, List @@ eB]] == 1,
candidates = {eA, eB};
Break[]];
],

```

```

    {j, i + 1, Length[shuffled]}
  ];
  If[candidates ≠ {}, Break[]];
  {i, 1, Length[shuffled] - 1}
];
];

If[candidates == {}, Return[{g, }]];

{e1, e2} = candidates;
y = Intersection[List @@ e1, List @@ e2][[1]];
x = Complement[List @@ e1, {y}][[1]];
z = Complement[List @@ e2, {y}][[1]];

(* --- B. 物理判定 --- *)
pX = GetPolarity[x];
pY = GetPolarity[y];
pZ = GetPolarity[z];

isHetero = (Abs[pX + pY + pZ] < 3) && (pX ≠ 0 && pY ≠ 0 && pZ ≠ 0);

(* 使用括号分组, 确保语法安全 *)
If[isHetero,
(
  (* 分支 1: 湮灭 *)
  neighbors = AdjacencyList[g, y];

  (* 伽马射线暴优化: 尽量生成非自环的边 *)
  (* 尝试 3 倍于邻居数量的连接, 使用 Select 过滤掉自连 *)
  energyRelease = Select[
    Table[UndirectedEdge[RandomChoice[neighbors],
      RandomChoice[neighbors], {Length[neighbors] * 3}],
      #[[1]] != #[[2]] &
  ];

  newG = VertexDelete[g, y];
  newG = EdgeAdd[newG, energyRelease];
  reactionType = ;
),
(

```

```

(* 分支 2: 生长 *)
w = Max[VertexList[g]] + 1;
SetPolarity[w, pY];

newActive = {UndirectedEdge[x, z], UndirectedEdge[x, w], UndirectedEdge[w, z]};
newFrozen = {DirectedEdge[x, y], DirectedEdge[y, z]};

newG = EdgeAdd[
  EdgeDelete[g, {e1, e2}],
  Join[newActive, newFrozen]
];
reactionType = ;
)
];

Return[{newG, reactionType}];
];

(* ===== *)
(* Part 3: 创世与统计 (Genesis & Statistics) *)
(* ===== *)

RunMatterAntimatterSimulation[steps_] := Module[{
  g, nodesM, nodesA, edgesM, edgesA,
  activeCurve = {}, massCurve = {},
  res, status, currentMass
},

(* --- 1. 创世 --- *)
Print[];

(* 物质 +1 *)
nodesM = Range[1, 15];
Scan[SetPolarity[#, 1] &, nodesM];
edgesM = Join[
  UndirectedEdge @@@ Partition[nodesM, 2, 1, 1],
  {UndirectedEdge[1, 5], UndirectedEdge[5, 10], UndirectedEdge[10, 15]}
];

(* 反物质 -1 *)

```

```

nodesA = Range[20, 35];
Scan[SetPolarity[#, -1] &, nodesA];
edgesA = Join[
  UndirectedEdge @@@ Partition[nodesA, 2, 1, 1],
  {UndirectedEdge[20, 25], UndirectedEdge[25, 30], UndirectedEdge[30, 35]}
];

g = Graph[Join[nodesM, nodesA], Join[edgesM, edgesA]];

(* --- 2. 接触 --- *)
g = EdgeAdd[g, UndirectedEdge[15, 20]];

Print[, steps,];

(* --- 3. 循环 --- *)
Monitor[Do[
  res = NaturalEvolutionStep[g];
  g = res[[1]];
  status = res[[2]];

(* 统计 *)
AppendTo[activeCurve, Count[EdgeList[g], _UndirectedEdge]];
currentMass = Total[VertexDegree[g]];
AppendTo[massCurve, currentMass];

, {i, 1, steps}],
Row[{, i, , status, , Last[activeCurve], , Last[massCurve]}]
];

(* --- 4. 可视化 --- *)
Print[];

GraphicsGrid[{
  {
    ListLinePlot[activeCurve,
      PlotLabel → Style[, Red, 13],
      PlotStyle → {Red, Thickness[0.004]},
      Frame → True, GridLines → Automatic, ImageSize → 400],

    ListLinePlot[massCurve,

```

```

PlotLabel → Style[, Blue, 13],
PlotStyle → {Blue, Thickness[0.004]},
Frame → True, GridLines → Automatic, ImageSize → 400]
}
}, ImageSize → 800]
]

```

(\* 运行 400 步 \*)

RunMatterAntimatterSimulation[200]

正在初始化正反物质团块...

开始自然演化 (Steps: 200)...

演化完成。生成图表...

Out[64]=

