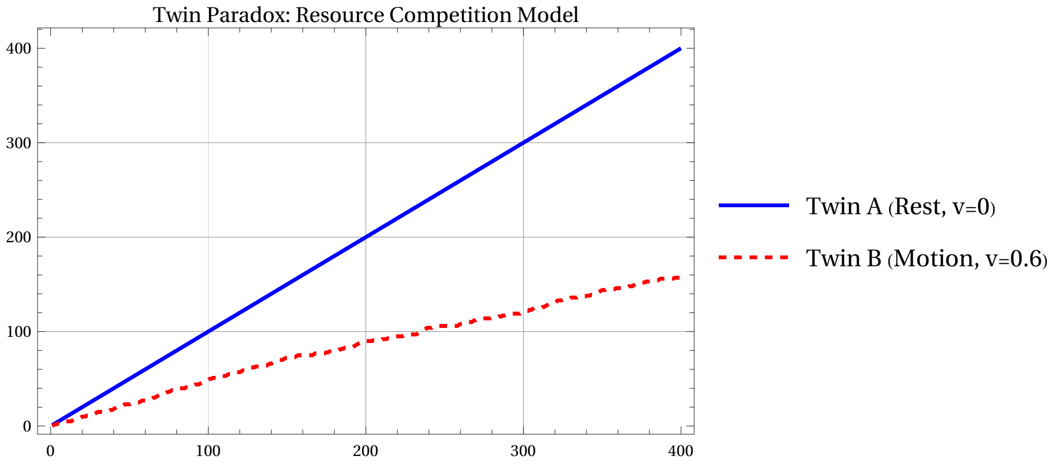
```
In[26]:= ClearAll["Global`*"];
SeedRandom[];

(* ========================================= *)
(* 1. 核心演化规则 (只负责，不负责) *)
(* ========================================= *)
(* 严格遵循你的 {{x,y},{y,z}} → {{x,z},{x,w},{w,z}} *)

ApplyGrowthRule[g_Graph] := Module[{
  edges, activeEdges, candidates, choice,
  e1, e2, x, y, z, w,
  newActive, newFrozen, newG
  },

 edges = EdgeList[g];
 activeEdges = Cases[edges, _UndirectedEdge]; (* 只有无向边是活性的 *)

 (* 寻找符合拓扑结构的边对 {{x,y}, {y,z}} *)
 candidates = {};
 (* 为了效率，随机采样寻找，找到一对立刻停止 *)
 Module[{shuffled, eA, eB},
  shuffled = RandomSample[activeEdges, Min[Length[activeEdges], 400]];
     (*To reduce the error, we can increase the number of
      samples to be the same as the number of iterations.*)
  Do[
   eA = shuffled⟦i⟧;
   Do[
    eB = shuffled⟦j⟧;
    If[Length[Intersection[List @@ eA, List @@ eB]] == 1,
     candidates = {eA, eB};
     Goto[];
     ];
    , {j, i + 1, Length[shuffled]}];
   , {i, 1, Length[shuffled] - 1}];
  ];

 Return[{g, 0}]; (* 没找到能反应的边，Tick不增加 *)

 Label[];
 {e1, e2} = candidates;
 y = Intersection[List @@ e1, List @@ e2]⟦1⟧;
```

```
  x = Complement[List @@ e1, {y}][[1]];
  z = Complement[List @@ e2, {y}][[1]];
  w = Max[VertexList[g]] + 1;

  (* 规则应用 *)
  newActive = {UndirectedEdge[x, z], UndirectedEdge[x, w], UndirectedEdge[w, z]};
  newFrozen = {DirectedEdge[x, y], DirectedEdge[y, z]}; (* 历史固化 *)

  newG = Graph[
   VertexList[g] ~Join~ {w},
   Union[Complement[edges, {e1, e2}], newActive, newFrozen]
  ];

  Return[{newG, 1}]; (* 返回新图，并标记 Tick + 1 *)
 ];


(* ======================================== *)
(* 2. 受控双生子实验 (Forced Causality)        *)
(* ======================================== *)

RunControlledTwin[totalSteps_, velocity_] := Module[{
  gA, gB,
  ageA = 0, ageB = 0,
  historyA = {}, historyB = {},
  res, isForcedMove
  },

 (* 初始化两个全等的宇宙泡 *)
 gA = Graph[{1 ↔ 2, 2 ↔ 3, 3 ↔ 1}];
 gB = Graph[{1 ↔ 2, 2 ↔ 3, 3 ↔ 1}];

 Print[];
 Print[];
 Print[, velocity*100, ];

 Monitor[Do[
  (* --- Twin A: 绝对静止 --- *)
  (* 它不需要处理位移，所有算力用于自身演化 *)
  res = ApplyGrowthRule[gA];
  gA = res[[1]];
  ageA += res⟦2⟧;

  (* --- Twin B: 强制运动 --- *)
  (* 掷骰子决定当前的时间片用于还是 *)
  (* 如果 Random < velocity，模拟该粒子正在穿过空间格点 *)
  (* 此时，计算资源被用于更新坐标（External），无法进行内部重写（Internal） *)

  isForcedMove = RandomReal[] < velocity;

  If[isForcedMove,
   (* 强制动：虽然消耗了全局Step，但内部没有变化 *)
   (* 实际上这里应该有坐标变换代码，但对于统计内部年龄，不操作=不长 *)
   Null,

   (* 没动：抓紧时间长一下 *)
   res = ApplyGrowthRule[gB];
   gB = res⟦1⟧;
   ageB += res⟦2⟧;
  ];

  (* 记录数据 *)
  AppendTo[historyA, ageA];
  AppendTo[historyB, ageB];

 , {step, 1, totalSteps}],
 Row[{, step, , ageA, , ageB}]
];

(* --- 3. 输出与可视化 --- *)
Print[, totalSteps, ];
Print[, ageA];
Print[, ageB];
Print[, N[ageB / ageA]];
Print[, 1.0 - velocity];

ListLinePlot[{historyA, historyB},
 PlotStyle →
   {Directive[Blue, Thickness[0.006]], Directive[Red, Dashed, Thickness[0.006]]},
 PlotLegends → {,  <> ToString[velocity] <>},
 AxesLabel → {, },
 PlotLabel → ,
  GridLines → Automatic,
  PlotTheme → ,
  ImageSize → Medium
 ]
]
```

```
(* 运行：设置总步数 2000, B的速度为 0.6c (即60%的时间在处理位移) *)
RunControlledTwin[400, 0.6]
```

```
启动强制对照实验...
Twin A (静止): 强制 100% 用于内部生长
Twin B (运动): 强制 60.% 用于位移 (资源被剥夺)

实验结束 (Global Steps = 400)
Twin A (Rest) Age: 400
Twin B (Move) Age: 157
Time Dilation Ratio (Observed): 0.3925
Theoretical Logic: 1 - v = 0.4
```

Out[30]=


Twin Paradox: Resource Competition Model
Twin A (Rest, v=0)
Twin B (Motion, v=0.6)