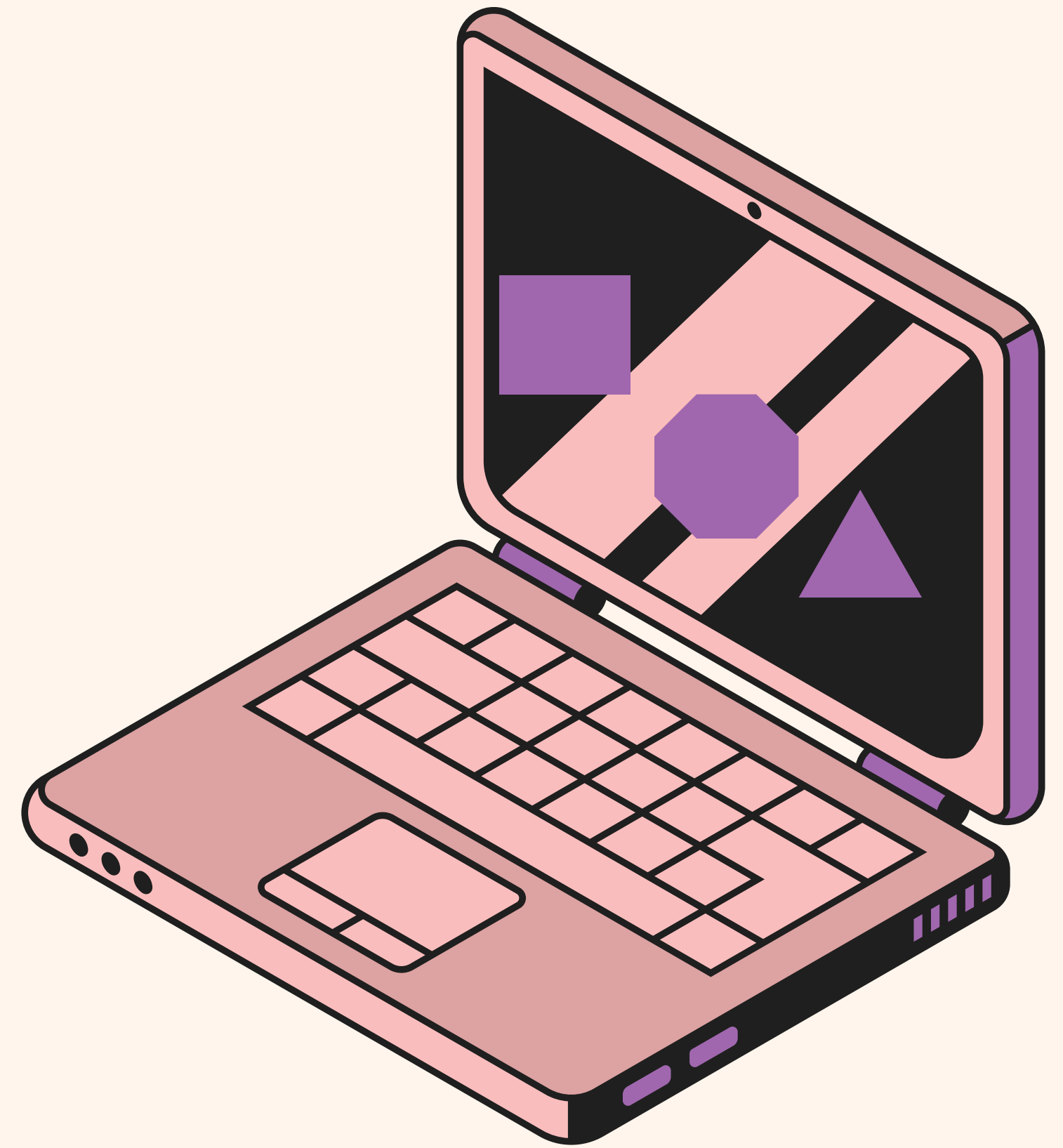


PROGETTO PCS



Andrea Tomatis, Matteo Villa, Cristian Fenoglio

SCOPO DEL PROGETTO

INPUT

Come input avremo delle tuple di 4 o 6 numeri interi.
(p, q, b, c, [id_vertice_1], [id_vertice_2])

OUTPUT

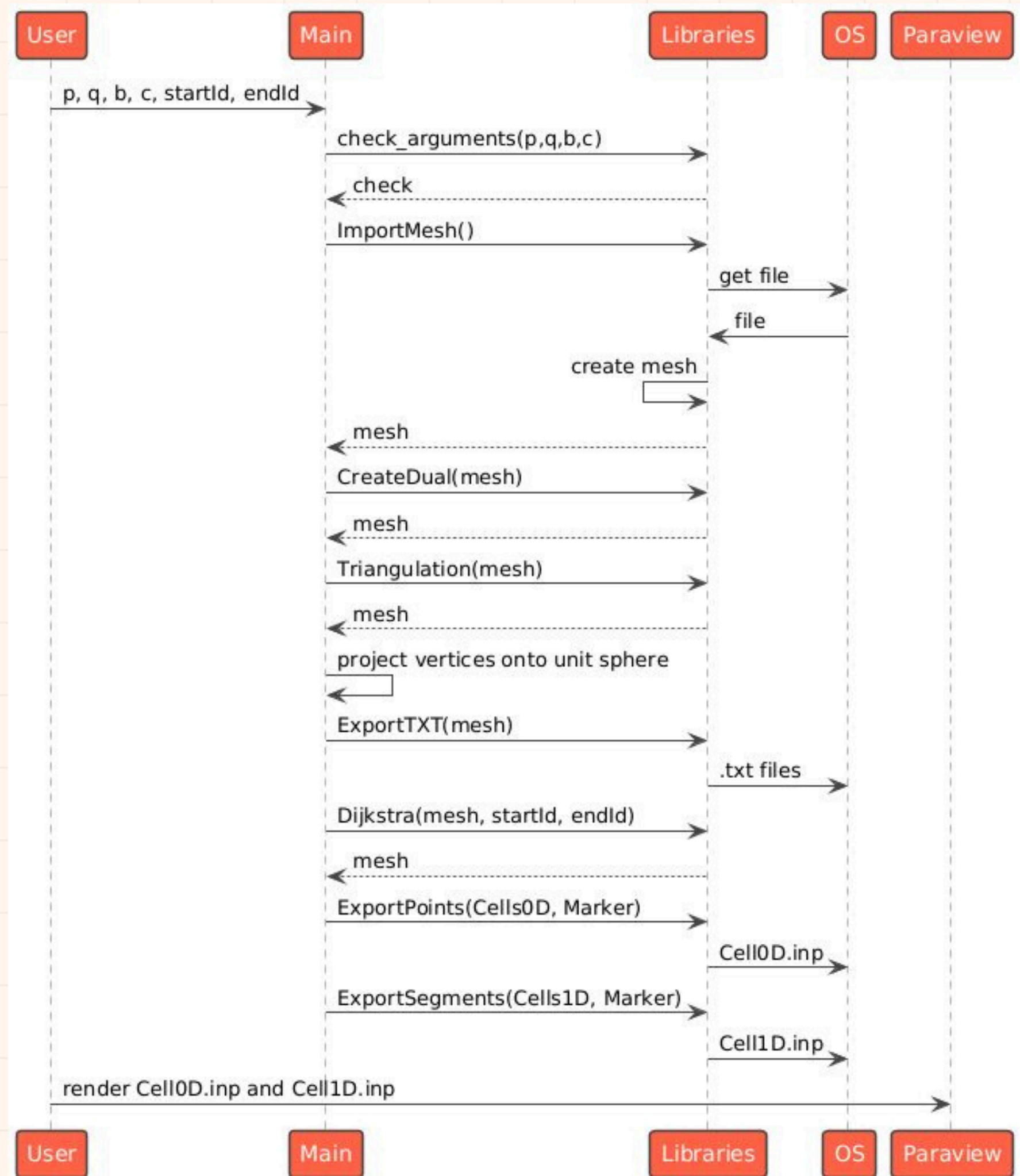
I file riguardanti vertici, lati, facce e solido, oltre alla stampa dei vertici su Paraview.

Il poliedro di **Goldberg** di classe I o II (se $b=c$) coi vertici che giacciono sulla sfera di raggio 1 centrata nell'origine degli assi.

Se l'input è costituito da una tupla di 6 elementi, verrà restituito il **cammino minimo** (evidenziato su Paraview), il numero di lati che lo compongono e la somma delle loro rispettive lunghezze

COME

Per spiegare il funzionamento del progetto ad alto livello, osserviamo questo UML.



COME



COME

PolygonalLibrary

C Polygon

- o VerticesCoordinates : MatrixXd
- o listVertices : std::vector<std::vector<unsigned int>>
- o p,q : unsigned int
- o NumCell0Ds : unsigned int
- o NumCell1Ds : unsigned int
- o NumCell2Ds : unsigned int
- o NumCell3Ds : unsigned int
- o Cell0DsId : vector<unsigned int>
- o Cell1DsId : vector<unsigned int>
- o Cell2DsId : vector<unsigned int>
- o Cell3DsId : vector<unsigned int>
- o Cell0DsCoordinates : Eigen::MatrixXd
- o Cell1DsExtrema : Eigen::MatrixXi
- o Cell2DsVertices : vector<vector<unsigned int>>
- o Cell2DsEdges : vector<vector<unsigned int>>
- o Cell3DsVertices : vector<vector<unsigned int>>
- o Cell3DsEdges : vector<vector<unsigned int>>
- o Cell3DsFaces : vector<vector<unsigned int>>
- o ShortestPathNodes : std::map<unsigned int, list<unsigned int>>
- o ShortestPathEdges : std::map<unsigned int, list<unsigned int>>
- PolygonalMesh()
- PolygonalMesh(unsigned int p, unsigned int q)
- PolygonalMesh CreateDual(){
- void ExportTXT()

Utilities

- bool check_arguments(unsigned int p, unsigned int q, unsigned int b, unsigned int c)
- bool ImportMesh(PolygonalMesh& mesh, string filename)
- bool ImportCell0Ds(PolygonalMesh& mesh, string filename)
- bool ImportCell1Ds(PolygonalMesh& mesh, string filename)
- bool ImportCell2Ds(PolygonalMesh& mesh, string filename)
- bool ImportCell3Ds(PolygonalMesh& mesh, string filename)
- PolygonalMesh Triangulation_1(PolygonalMesh mesh, unsigned int b, unsigned int T)
- PolygonalMesh Triangulation_2(PolygonalMesh mesh, unsigned int b, unsigned int T)
- int get_id_2(Point p, PolygonalMesh geodetic)
- int get_edge(unsigned int id1, unsigned int id2, PolygonalMesh geodetic)
- void side(PolygonalMesh& geodetic, double bb, Point p0, Point u, Point r1, Point r2, int b, unsigned int& vertex_cnt, unsigned int& edge_cnt)
- int get_id_1(Point p, PolygonalMesh geodetic)

COME

TestLibrary

C PolygonalMeshTest

- imp1 : bool
- imp2 : bool
- imp3 : bool
- polygons : PolygonalMesh[3];
- duals : PolygonalMesh[3];
- PolygonalMeshTest()
- TEST_F(PolygonalMeshTest, Imports)
- TEST_F(PolygonalMeshTest, NonNullEdges)
- TEST_F(PolygonalMeshTest, NonNullEdges_Dual)
- TEST_F(PolygonalMeshTest, NonNullEdges_Geodetic)
- TEST_F(PolygonalMeshTest, NonNullEdges_Goldberg)
- TEST_F(PolygonalMeshTest, SameLengthEdges)
- TEST_F(PolygonalMeshTest, SameLengthEdges_Dual)
- TEST_F(PolygonalMeshTest, NonNullFaces)
- TEST_F(PolygonalMeshTest, NonNullFaces_Dual)
- TEST_F(PolygonalMeshTest, NonNullFaces_Geodetic)
- TEST_F(PolygonalMeshTest, NonNullFaces_Goldberg)
- TEST_F(PolygonalMeshTest, InCircumference)
- TEST_F(PolygonalMeshTest, InCircumference_Dual)
- TEST_F(PolygonalMeshTest, InCircumference_Geodetic)
- TEST_F(PolygonalMeshTest, InCircumference_Goldberg)

ShortestPathLib

Dijkstra

- vector<vector<unsigned int>> AdjList(const Eigen::MatrixXi& edges, unsigned int V)
- PolygonalMesh Dijkstra(PolygonalMesh& mesh, const unsigned int& src, const unsigned int& dest)

DUALE

INPUT

- Funzione membro — agisce sull'istanza della chiamata

OUTPUT

- PolygonalMesh // La nostra mesh duale

DUALE

Faccia \rightarrow *vertice*

(*Vertice* \rightarrow *faccia*)

DUALE

- Mappiamo vecchi lati (come estremi) alle facce adiacenti
- Creiamo nuovi lati tra centri di vecchie facce adiacenti
- Per ogni vecchio vertice
 - Prendiamo i lati provenienti da esso
 - Prendiamo i nuovi lati “duali” costruiti sui lati
 - Uniamo come faccia
- Salviamo tutto e proiettiamo sulla sfera unitaria

SHORTEST PATH

INPUT

- PolygonalMesh& mesh // La nostra mesh
- const unsigned int& src // Nodo di partenza
- const unsigned int& dest // Nodo di arrivo

OUTPUT

- PolygonalMesh // La nostra mesh con marker per il cammino minimo

DIJKSTRA

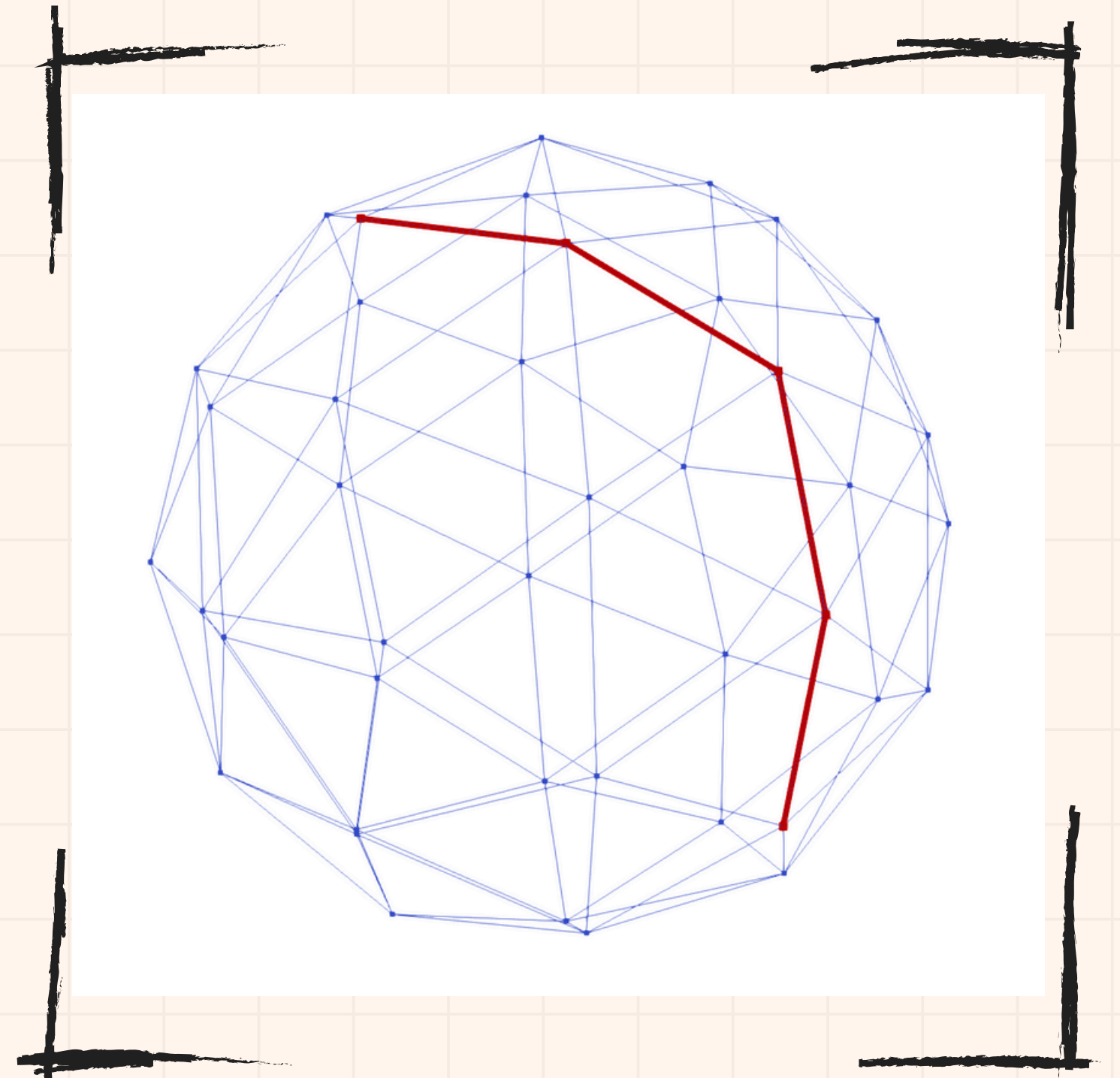
- Semplice ed efficiente
- Nodi di partenza e arrivo dati
- Scelta di grafo pesato o non (sempre positivo)

DIJKSTRA

- Priority queue (min. heap)
 - Elementi sono i vertici
 - Priorità è la distanza
- AdjList() // Lista di adiacenza, ideale per grafi sparsi
- 2 vettori
 - predecessori — inizialmente -1
 - distanze — inizialmente inf.
- Logica Dijkstra
 - Esplora nodi raggiungibili con distanza minima nota
 - Aggiorna distanze di altri nodi se trova percorso più breve

DIJKSTRA

- Ricostruisce cammino minimo da nodo di arrivo a nodo di partenza
 - Camminando "all'indietro" e invertendo
- Salva in PolygonalMesh con marker
 - 0 non presente
 - 1 presente

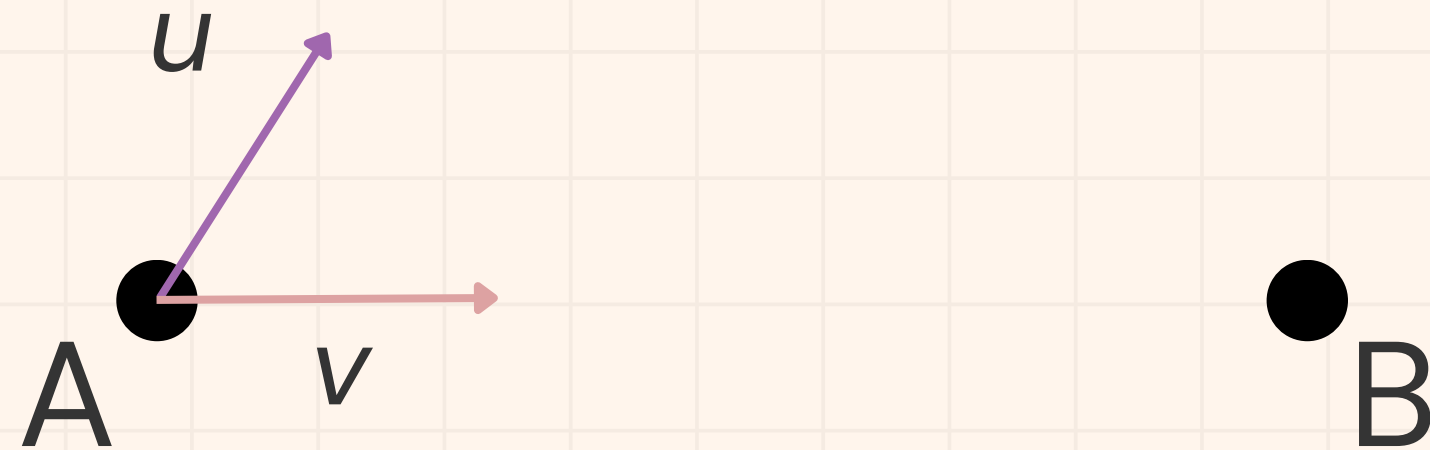


TRIANGOLAZIONE 1

$$u = (C-A)/b$$

$$v = (B-A)/b$$

$$b = 3$$

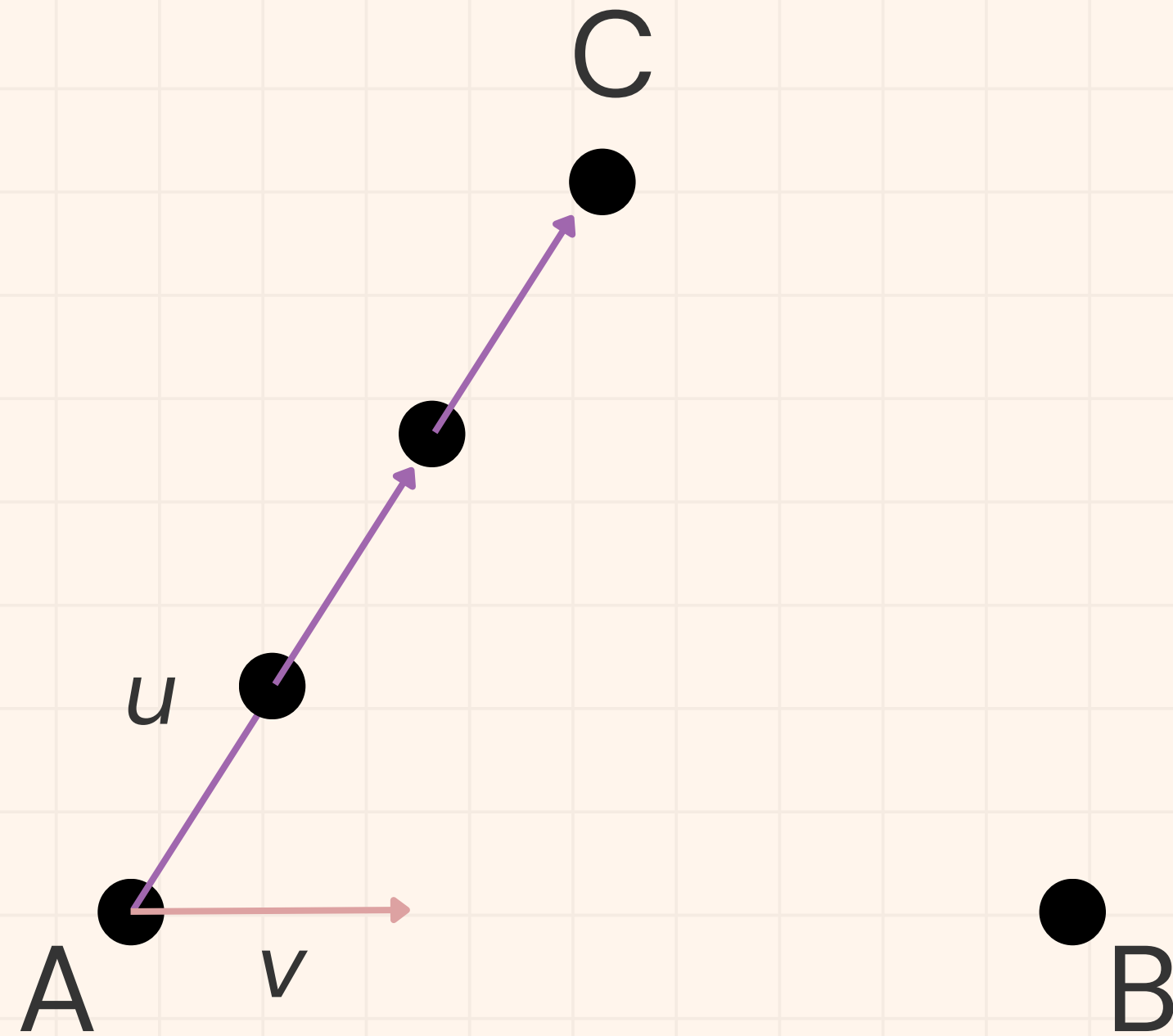


TRIANGOLAZIONE 1

$$u = (C-A)/b$$

$$v = (B-A)/b$$

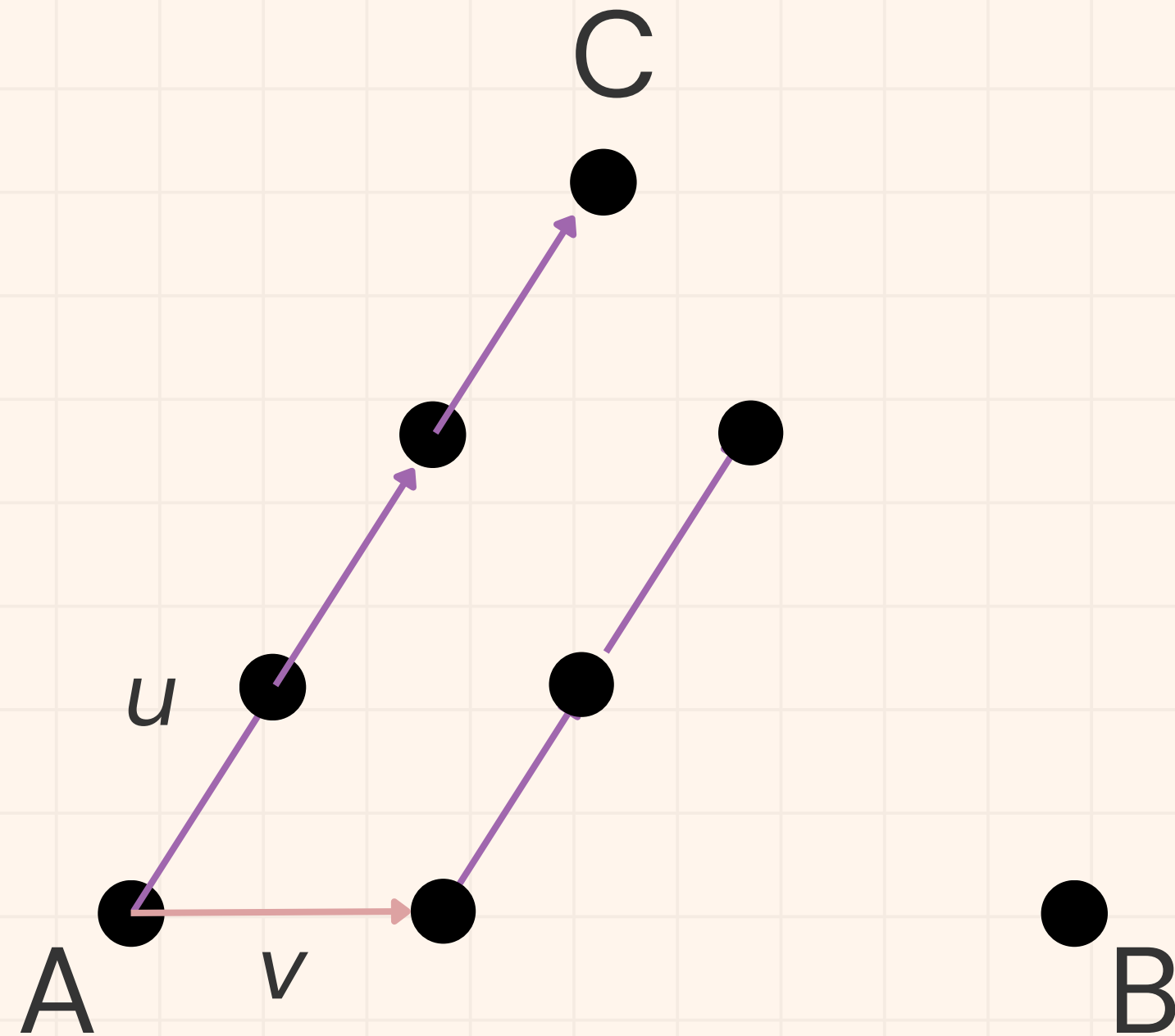
$$b = 3$$



TRIANGOLAZIONE 1

$$u = (C-A)/b$$
$$v = (B-A)/b$$

$$b = 3$$

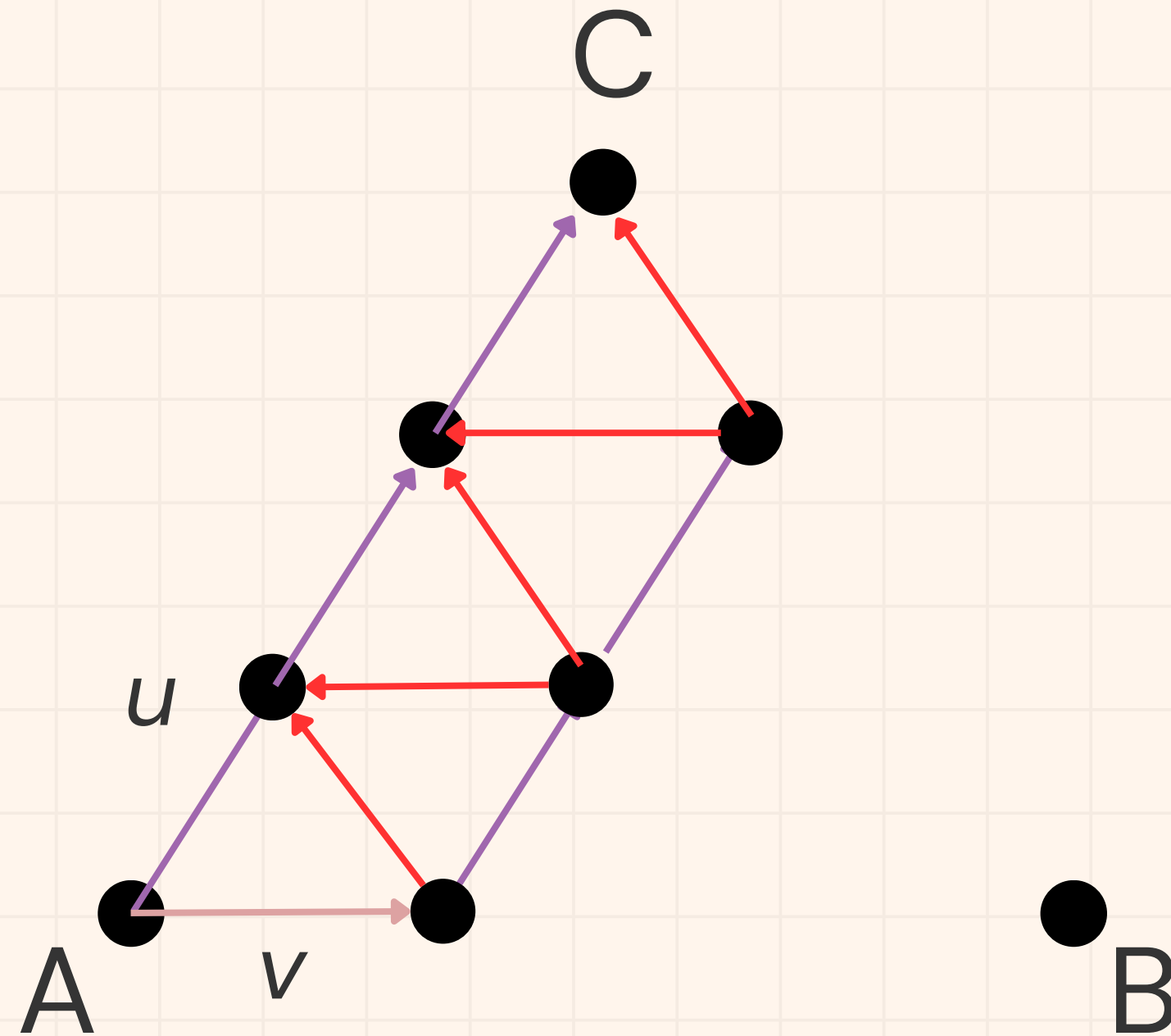


TRIANGOLAZIONE 1

$$u = (C-A)/b$$

$$v = (B-A)/b$$

$$b = 3$$

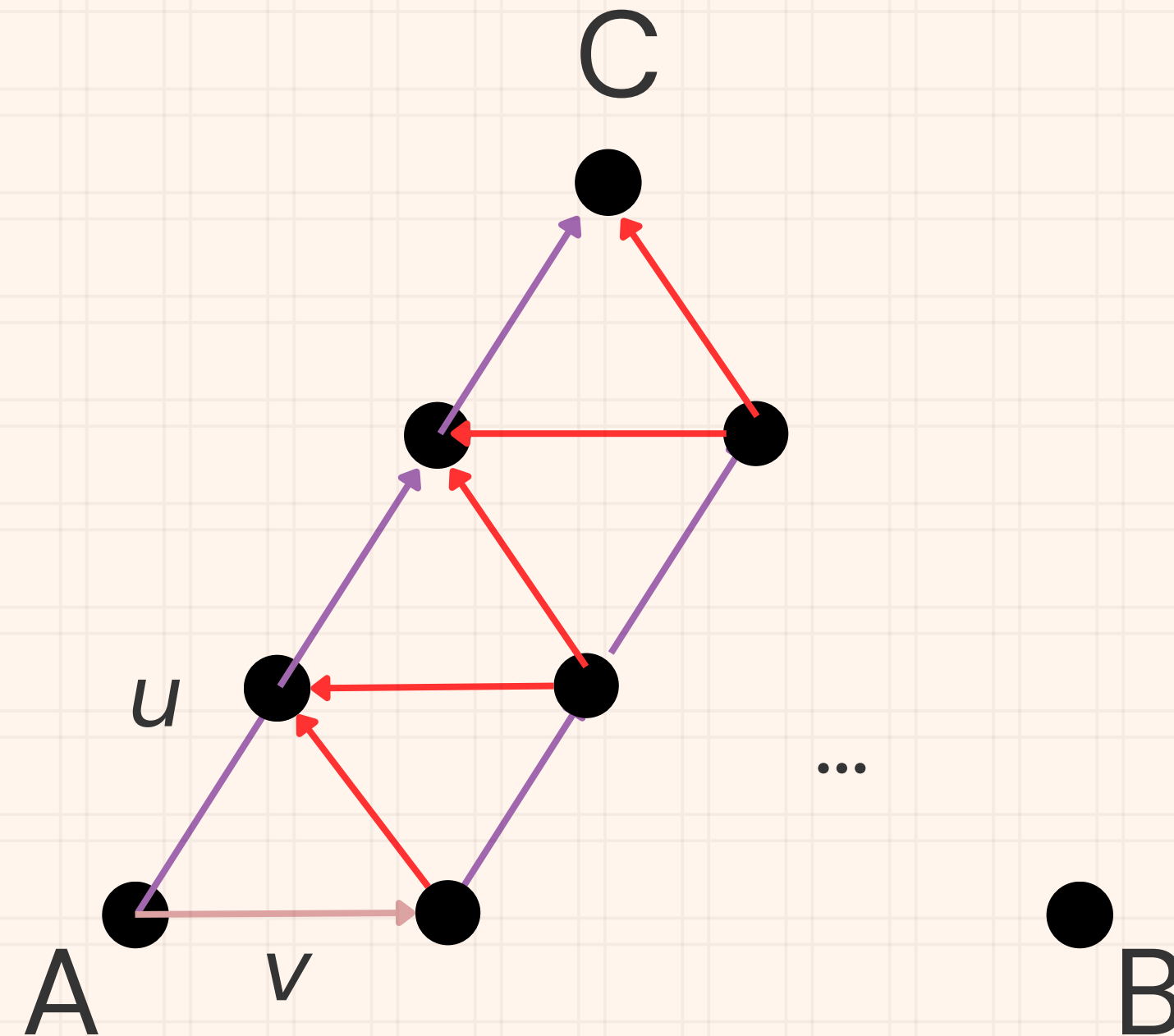


TRIANGOLAZIONE 1

$$u = (C-A)/b$$

$$v = (B-A)/b$$

$$b = 3$$

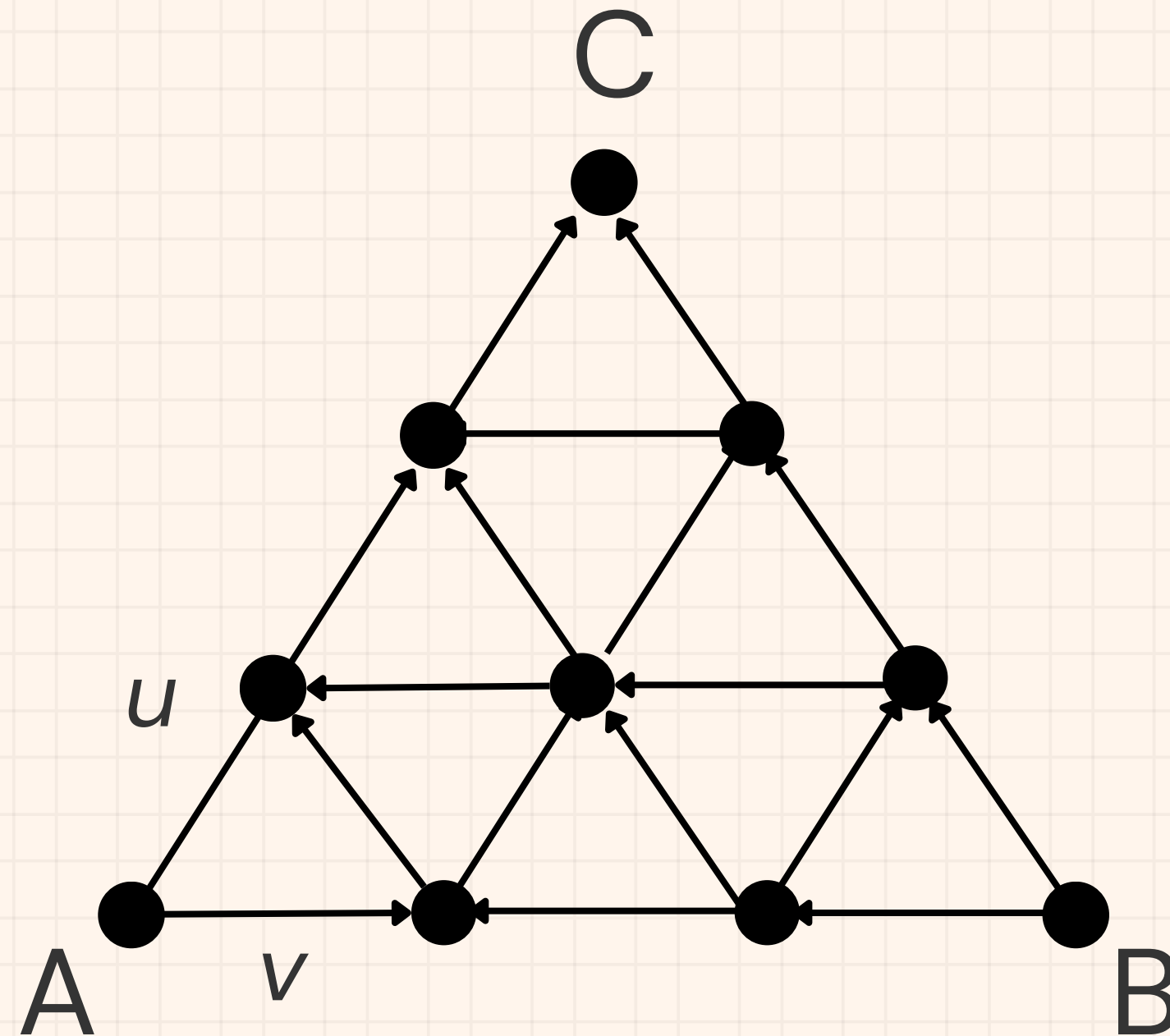


TRIANGOLAZIONE 1

$$u = (C-A)/b$$

$$v = (B-A)/b$$

$$b = 3$$



TRIANGOLAZIONE 2

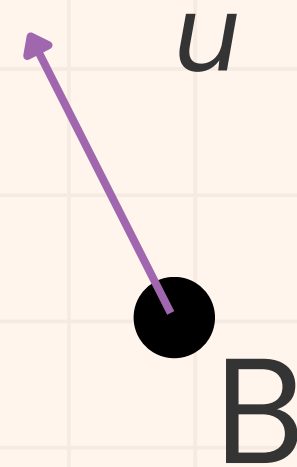
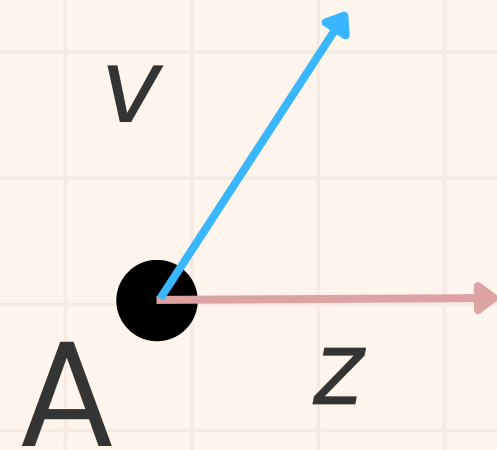
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

C
•

$$b = 2$$



TRIANGOLAZIONE 2

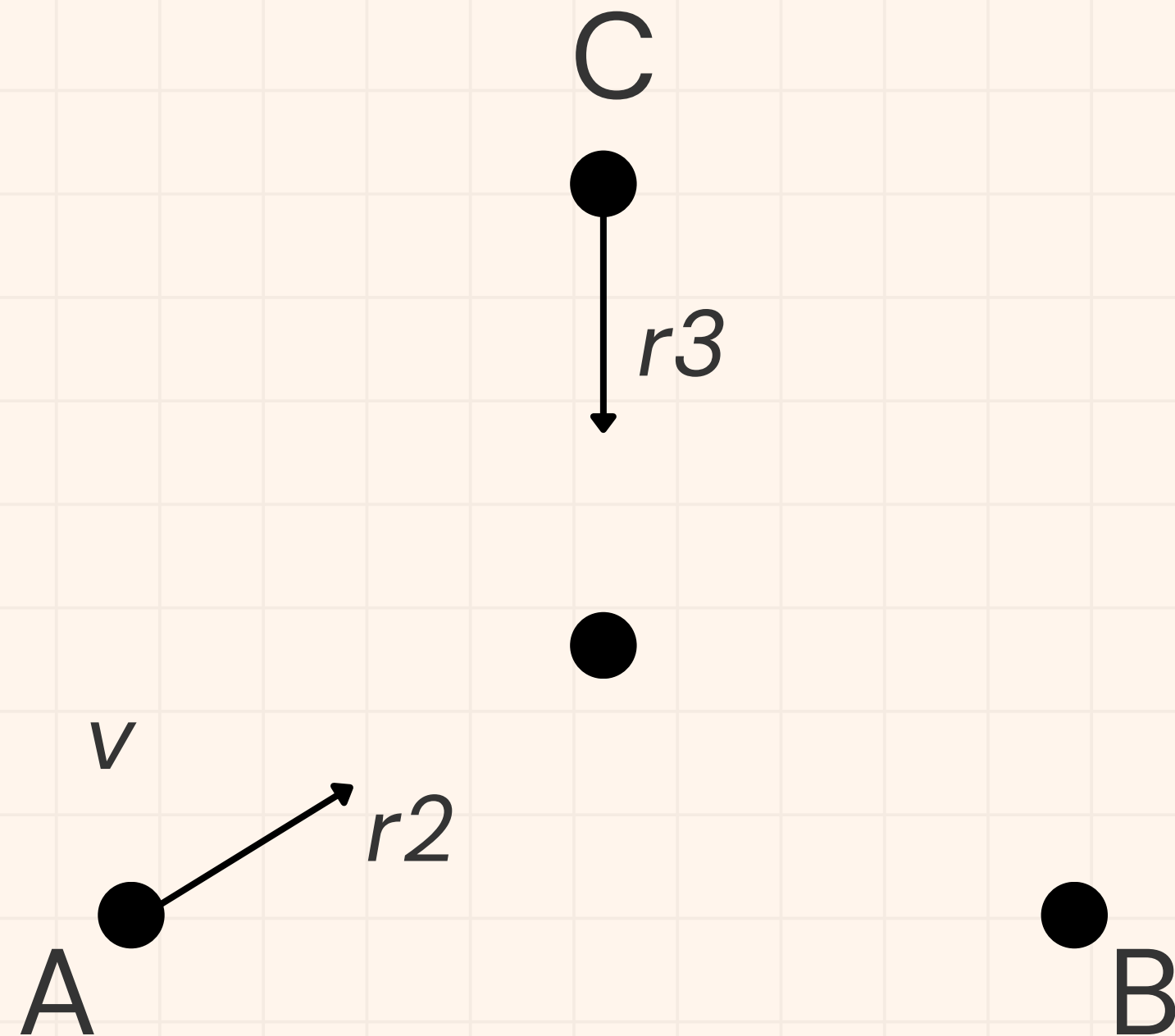
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

$$d = (A+B+C)/3$$

$$b = 2$$



$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$

TRIANGOLAZIONE 2

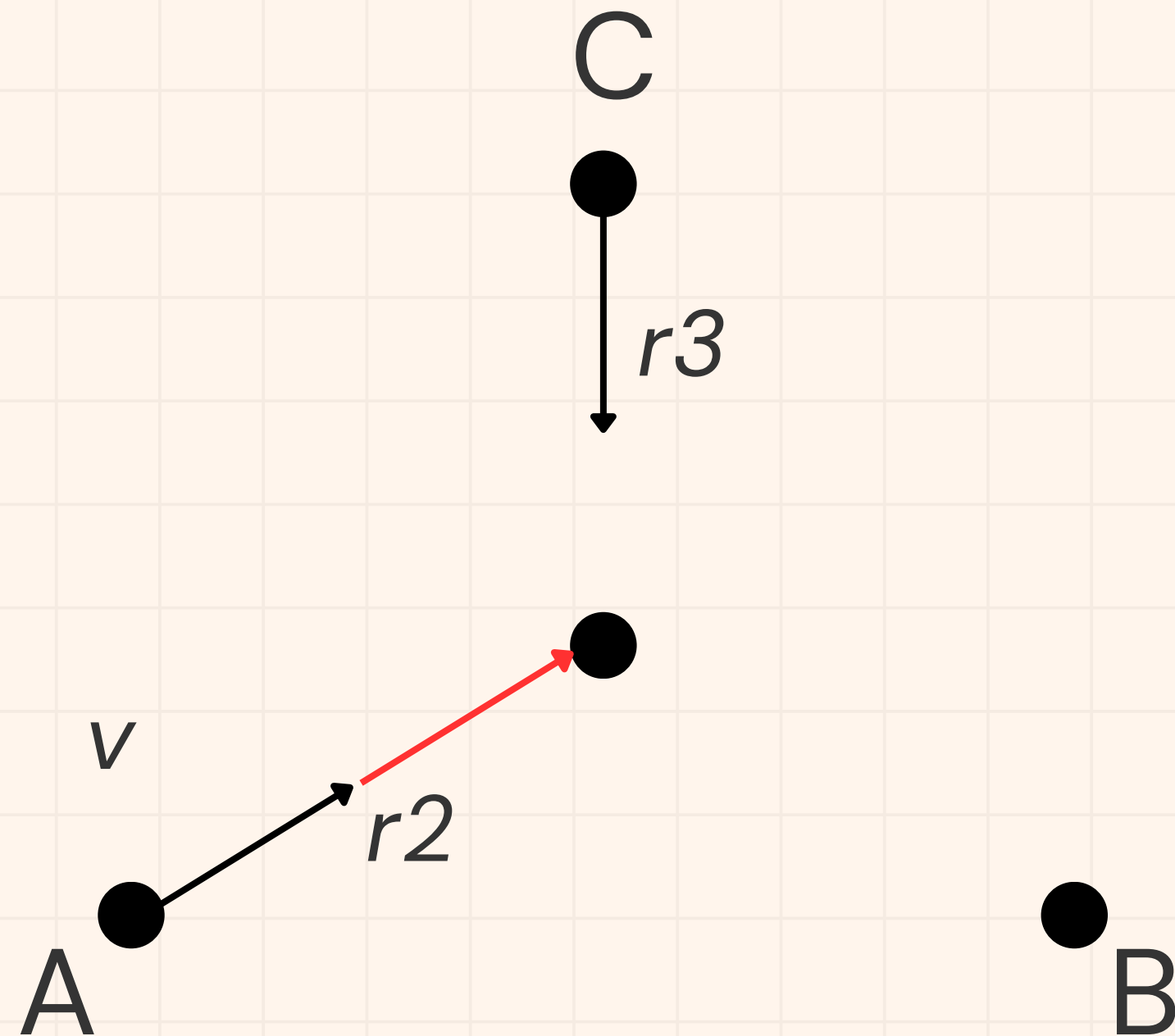
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

$$d = (A+B+C)/3$$

$$b = 2$$



$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$

TRIANGOLAZIONE 2

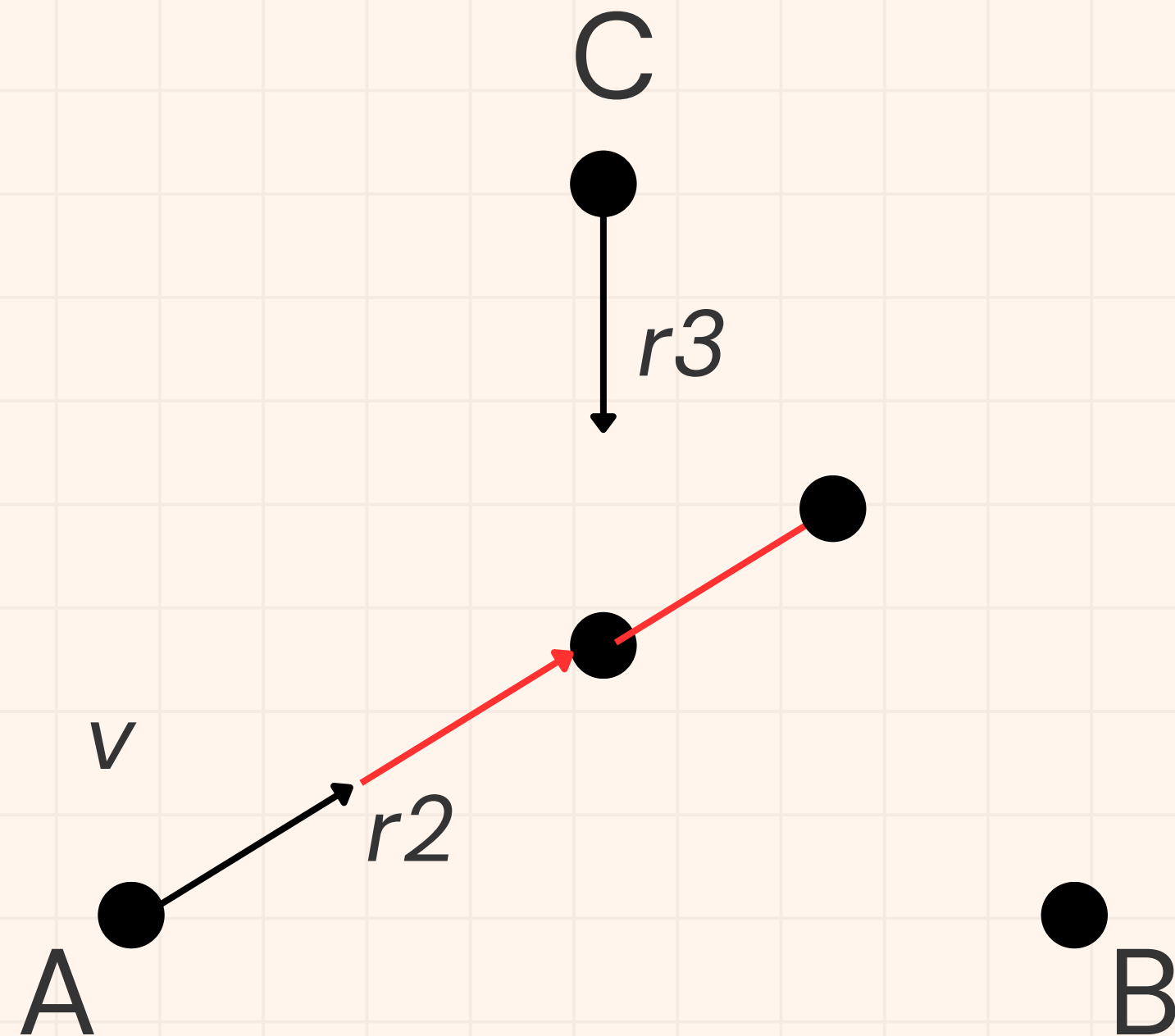
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

$$d = (A+B+C)/3$$

$$b = 2$$



$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$

TRIANGOLAZIONE 2

$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

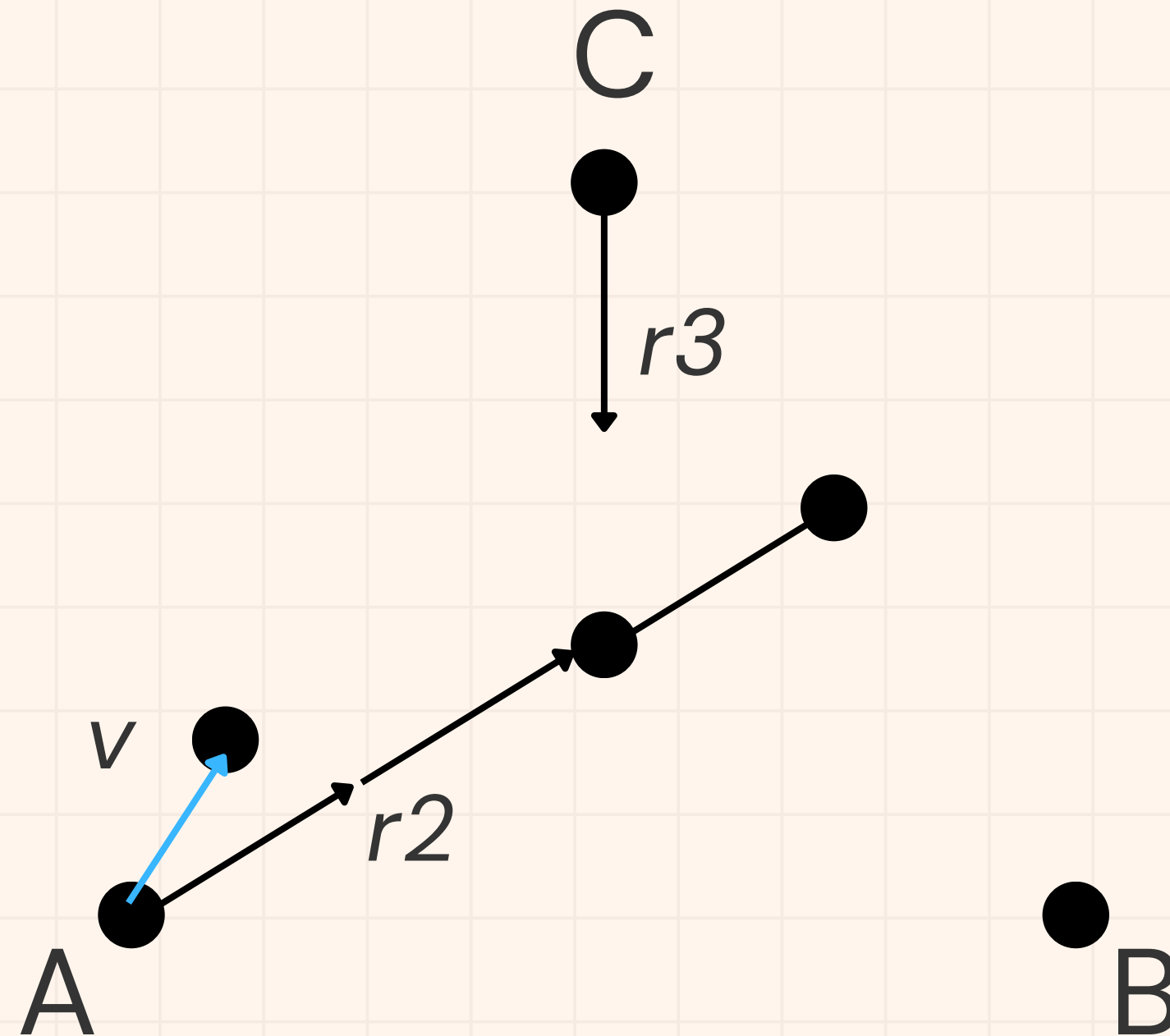
$$d = (A+B+C)/3$$

$$b = 2$$

$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$



TRIANGOLAZIONE 2

$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

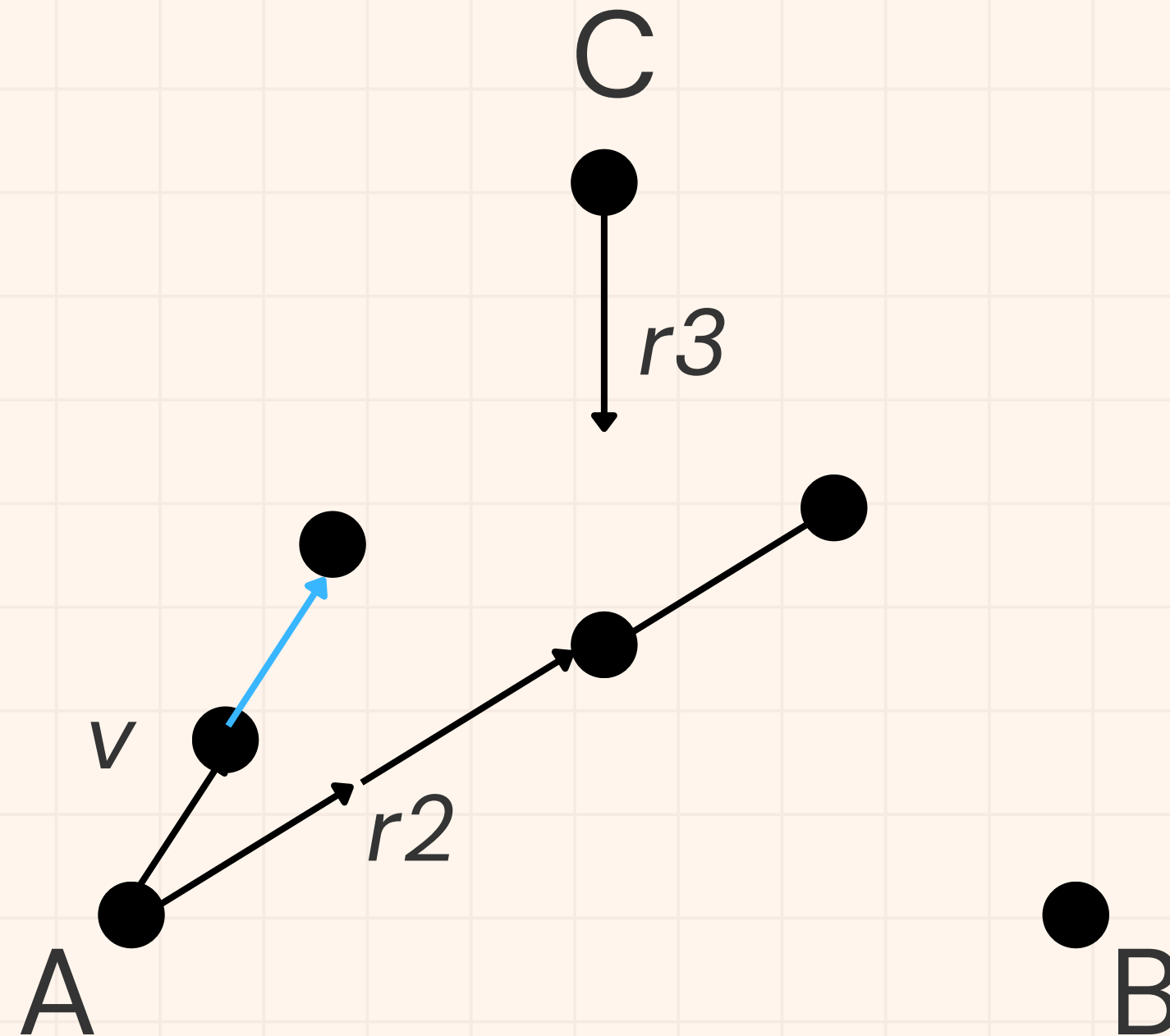
$$d = (A+B+C)/3$$

$$b = 2$$

$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$



TRIANGOLAZIONE 2

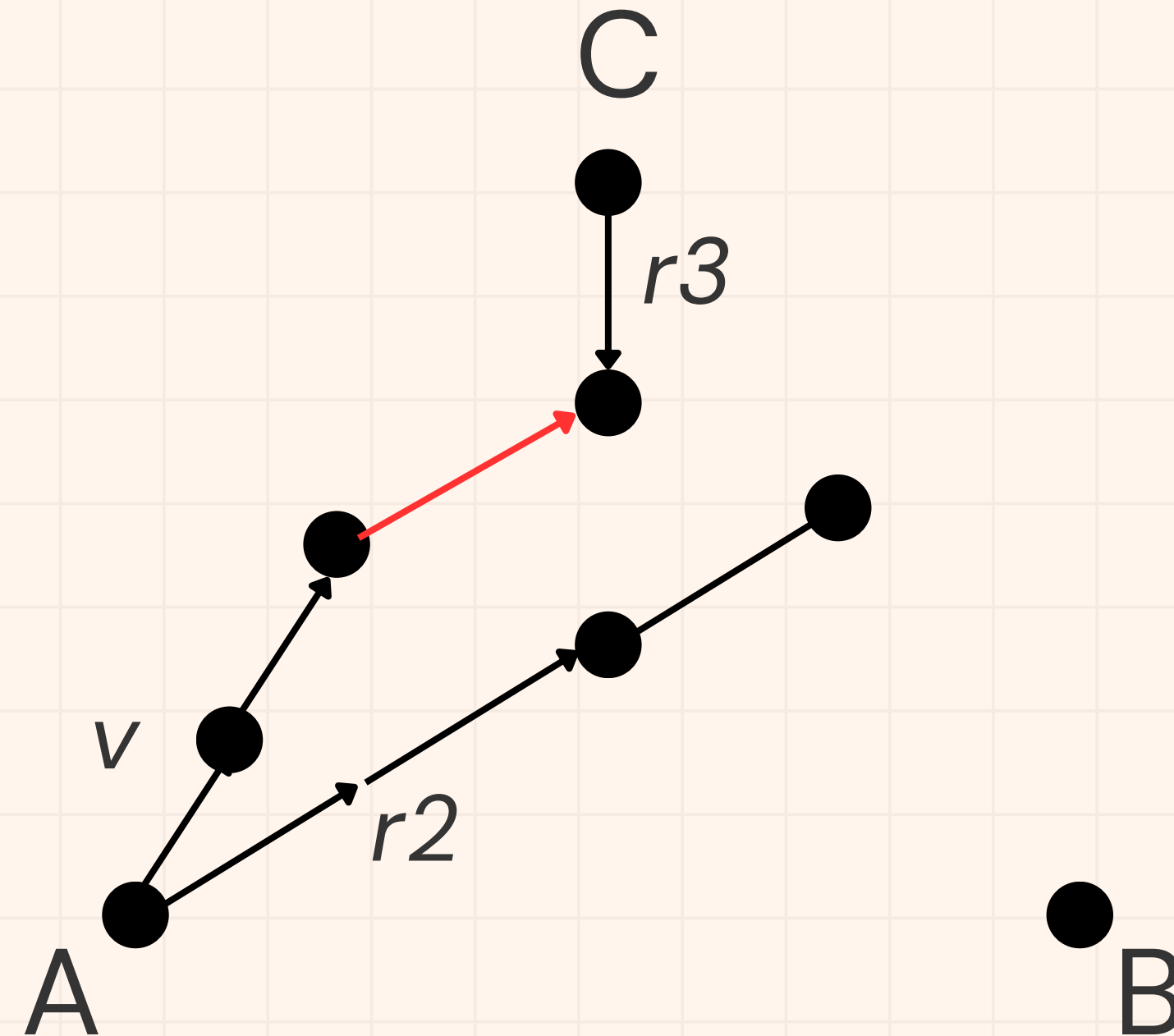
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

$$d = (A+B+C)/3$$

$$b = 2$$



$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$

TRIANGOLAZIONE 2

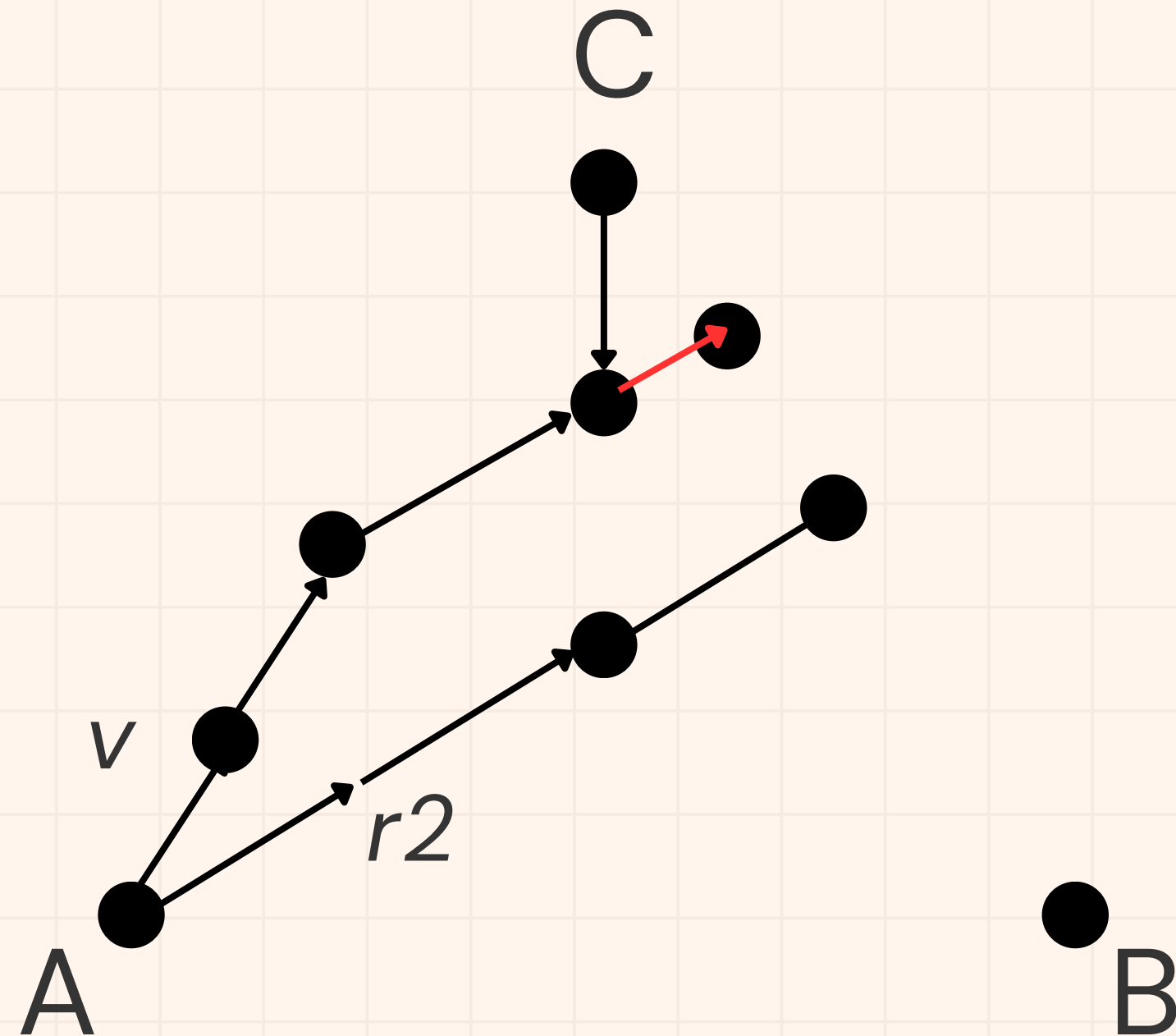
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

$$d = (A+B+C)/3$$

$$b = 2$$



$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$

TRIANGOLAZIONE 2

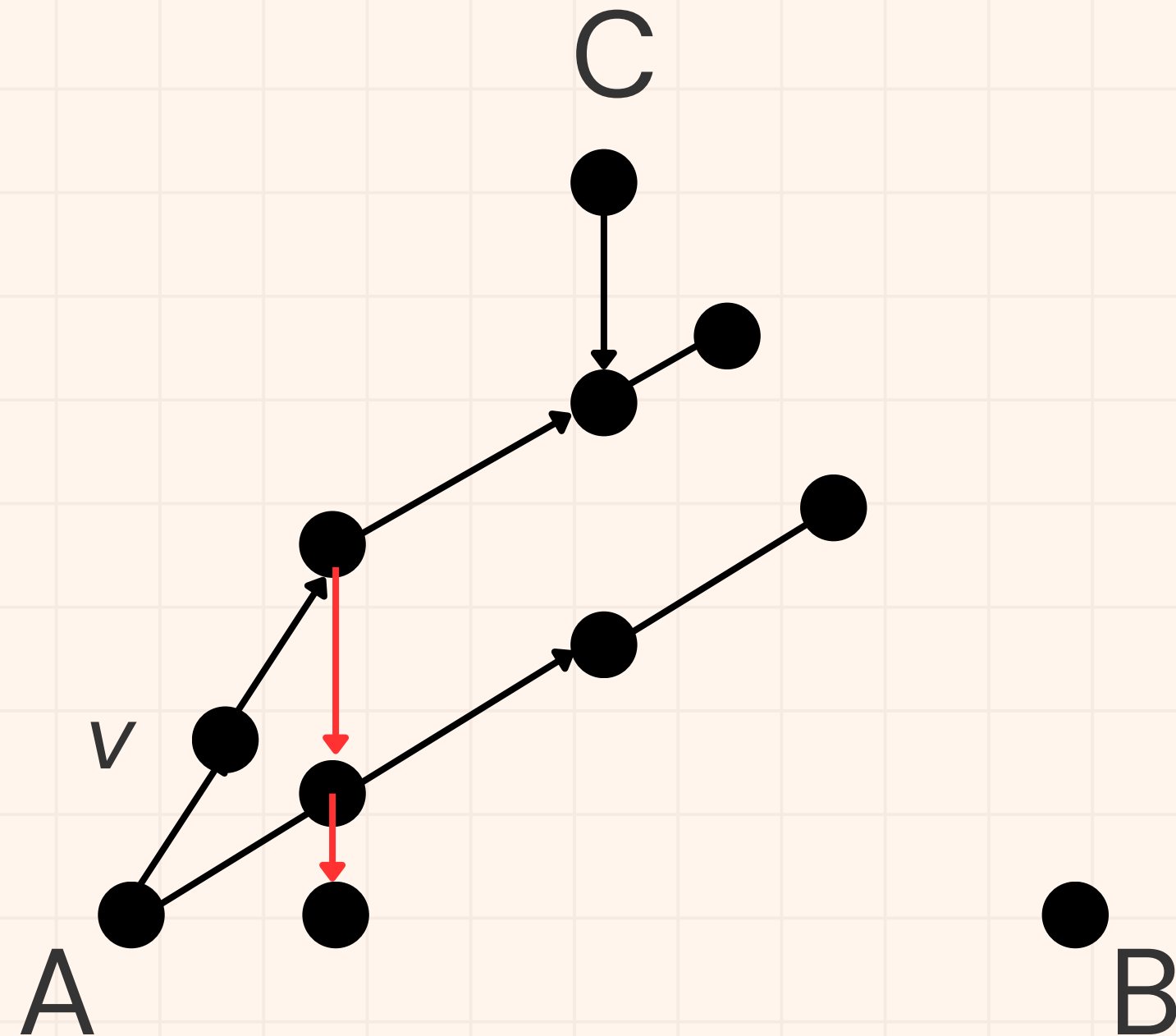
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

$$d = (A+B+C)/3$$

$$b = 2$$



$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$

TRIANGOLAZIONE 2

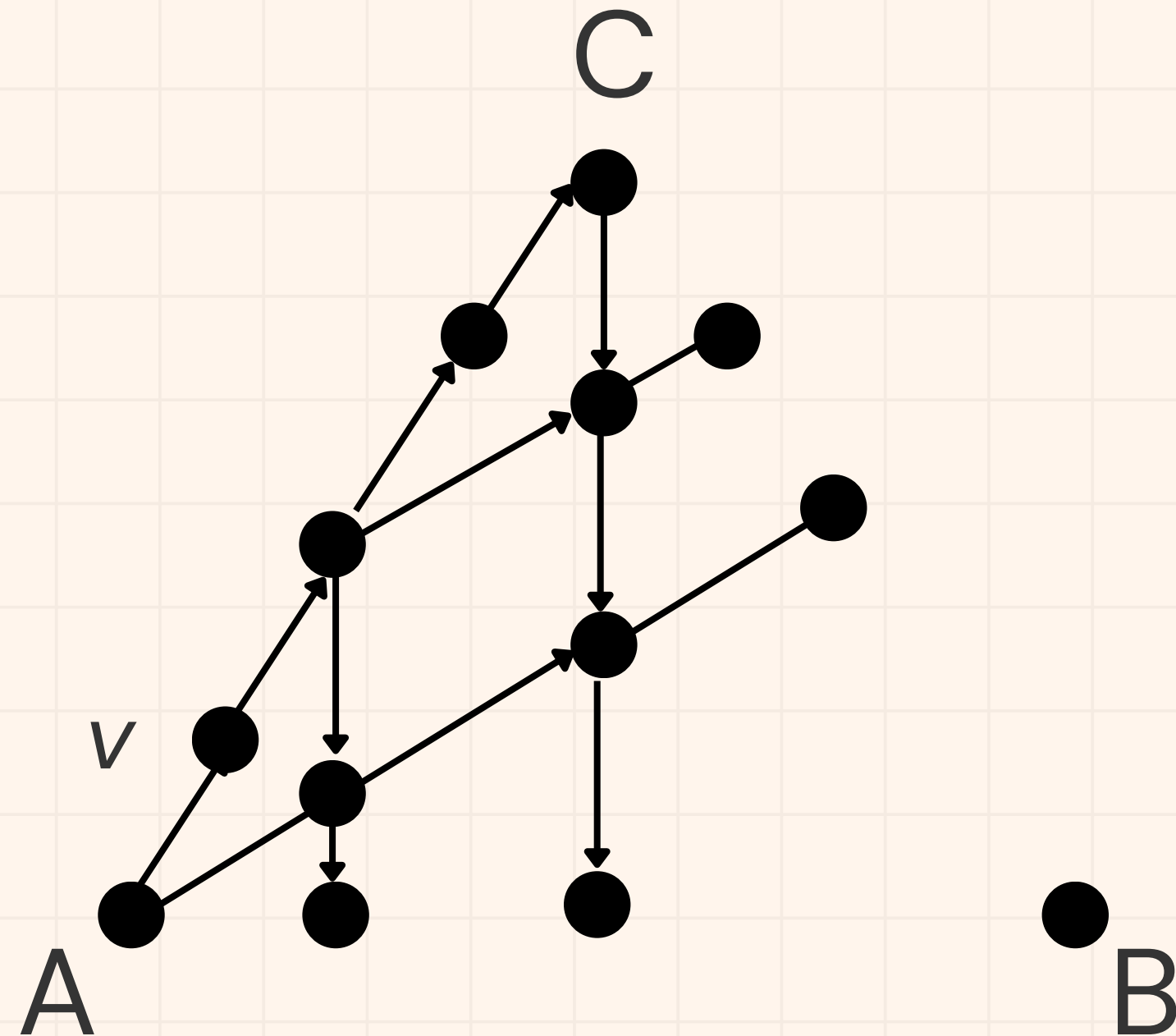
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

$$d = (A+B+C)/3$$

$$b = 2$$



$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$

TRIANGOLAZIONE 2

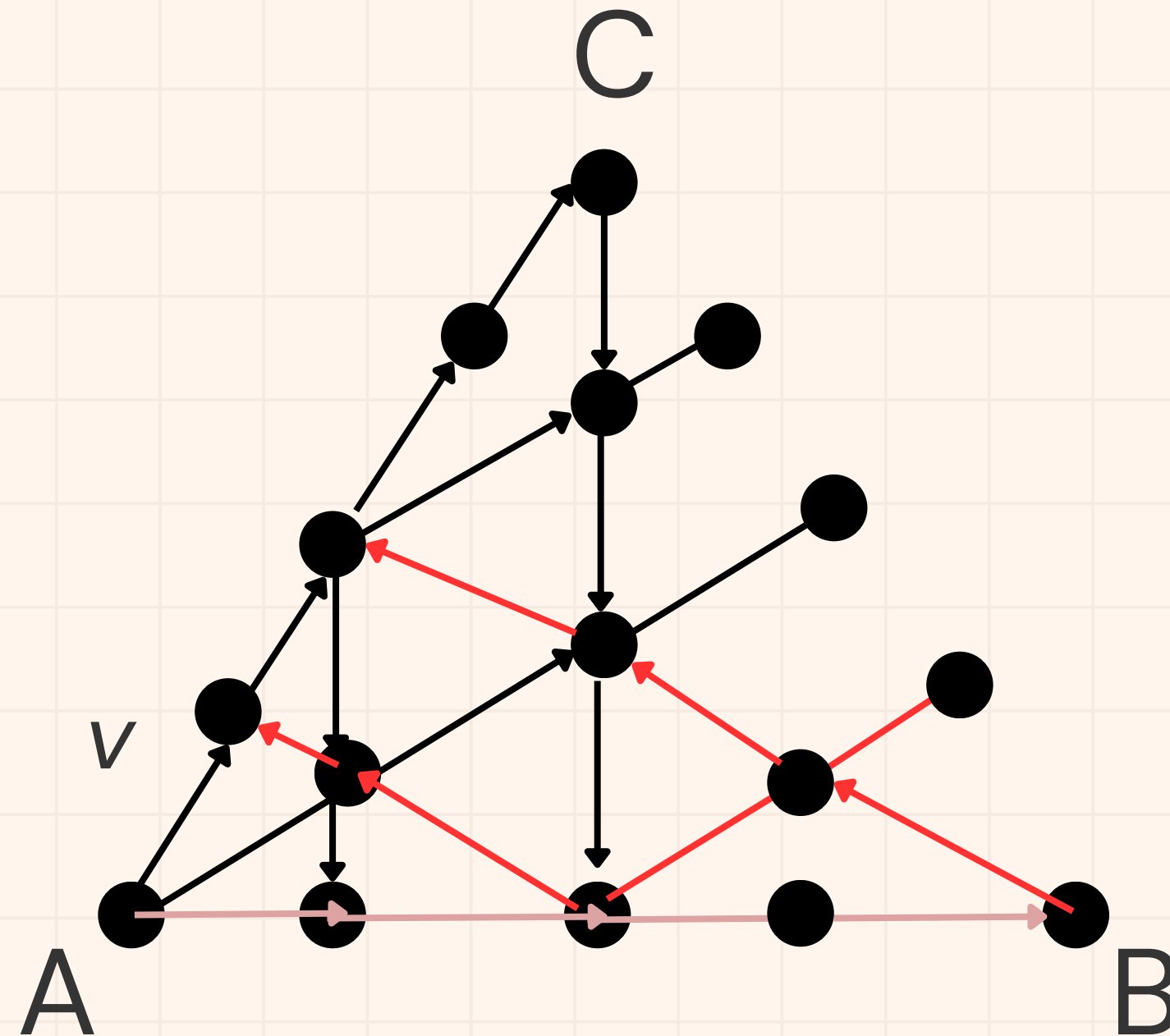
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

$$d = (A+B+C)/3$$

$$b = 2$$



$$r1 = (d-B)/b$$

$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$

TRIANGOLAZIONE 2

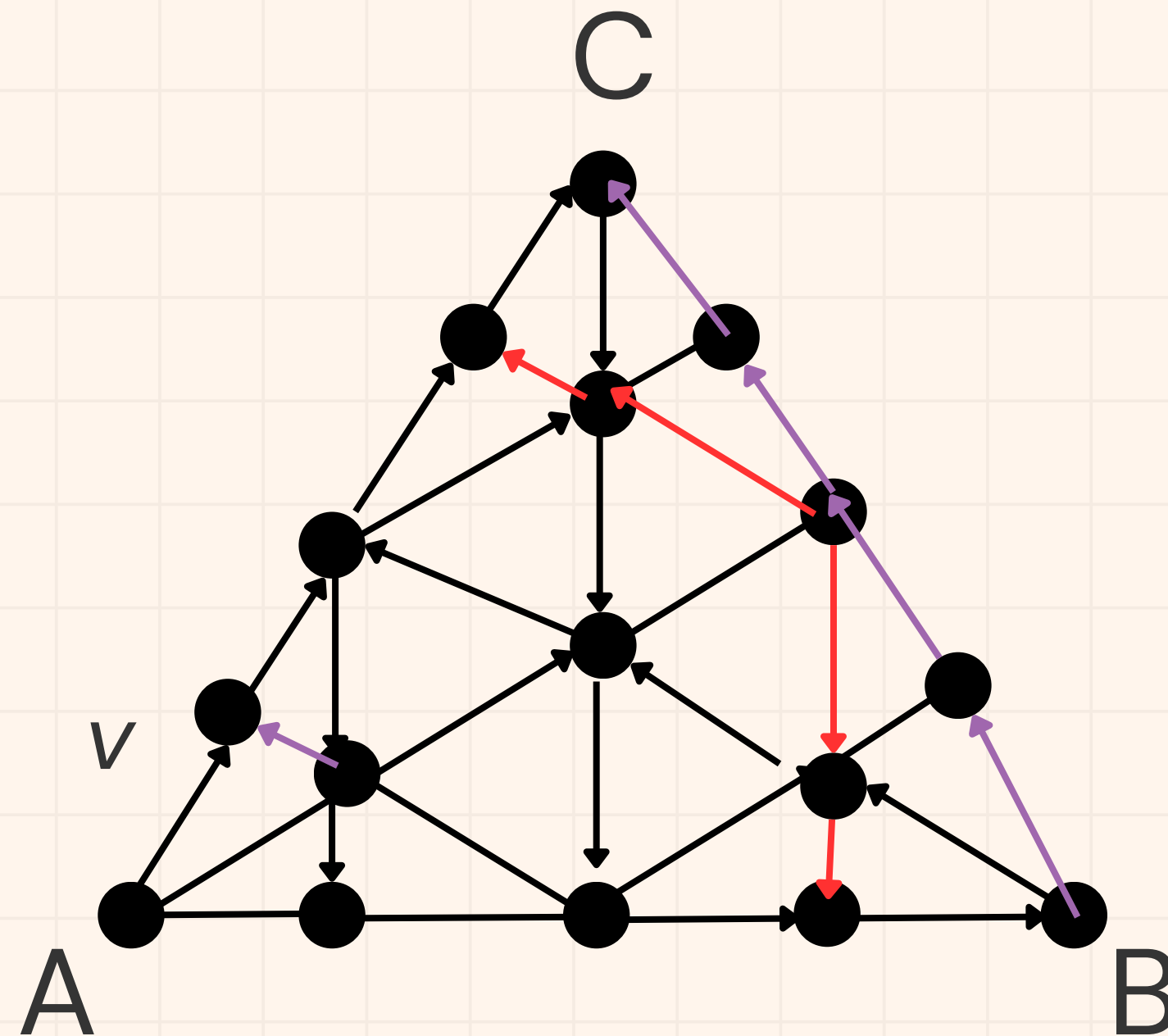
$$u = (C-B)/2b$$

$$v = (C-A)/2b$$

$$z = (B-A)/2b$$

$$d = (A+B+C)/3$$

$$b = 2$$

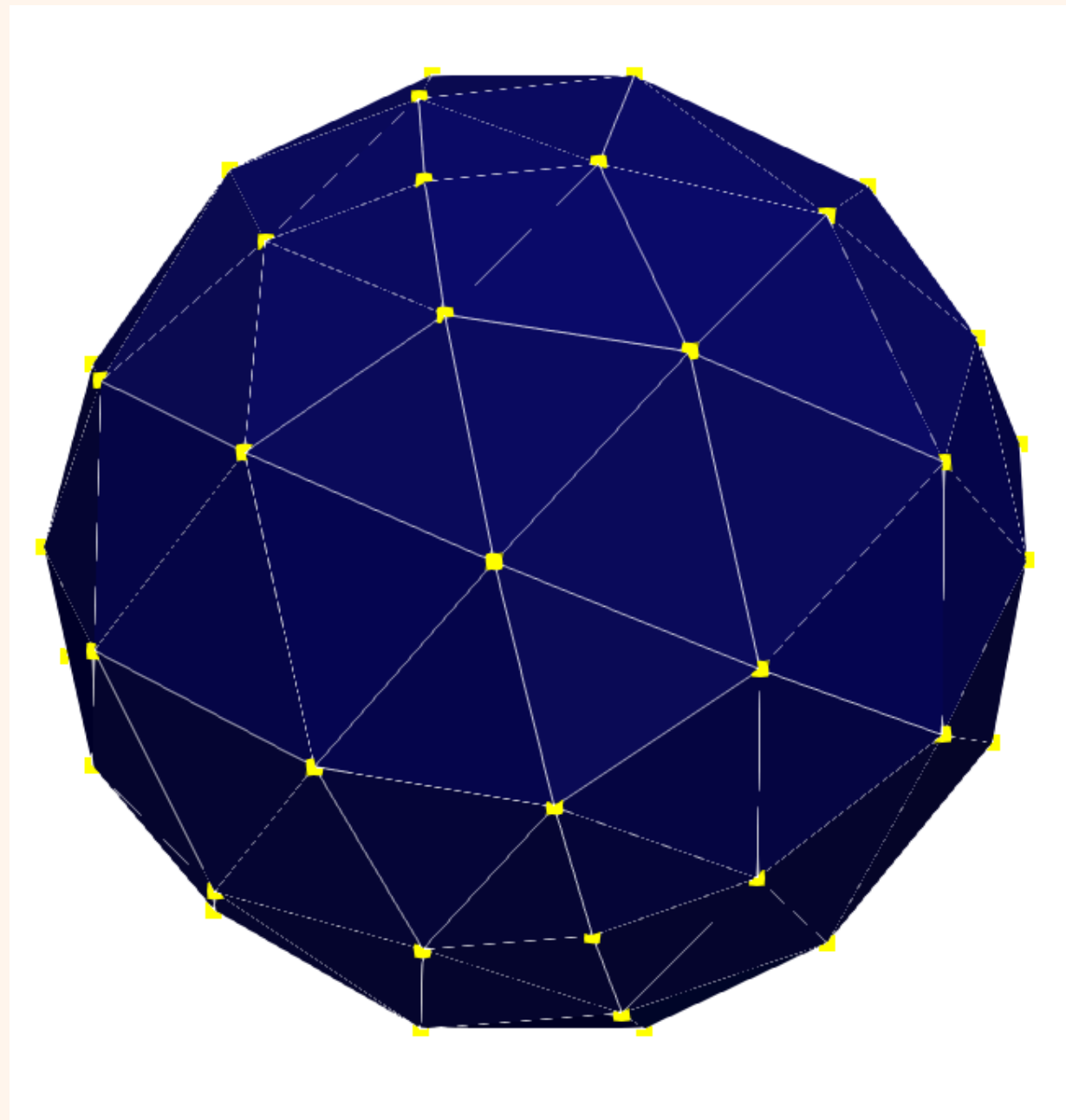


$$r1 = (d-B)/b$$

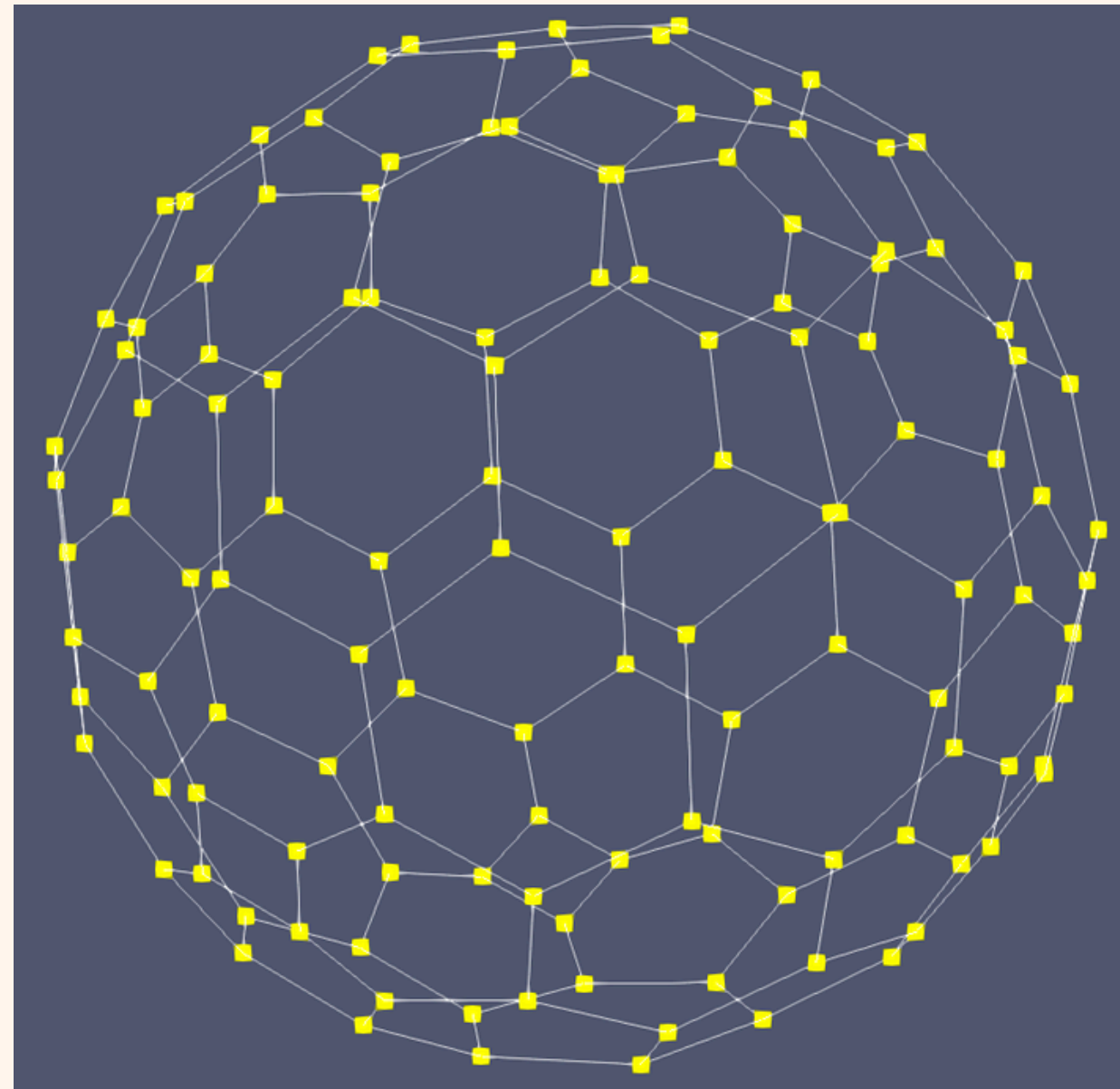
$$r2 = (d-A)/b$$

$$r3 = (d-C)/b$$

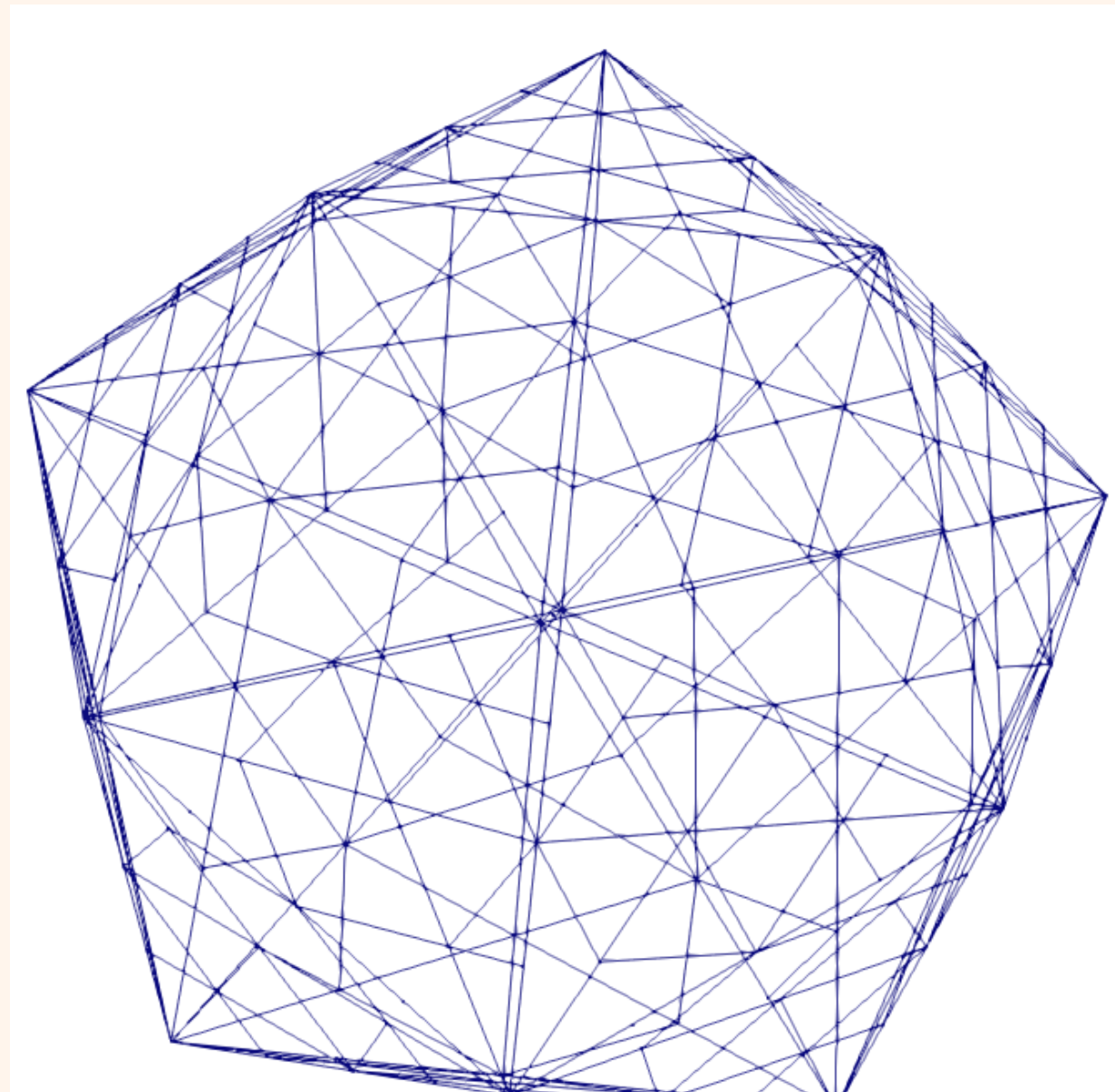
ESEMPIO: $\{3,4\}$ $B=4$, $C=0$



ESEMPIO: $\{4,3\}$ $B=4$, $C=0$



ESEMPIO: $\{5,3\}$ $B=2$, $C=2$



G-TEST

Per concludere presentiamo
l'infrastruttura di testing.

Si tratta di 1 suite di Gtest composta da
15 test in tutto.

