In questa sezione descriviamo i test unitari che abbiamo implementato.

Abbiamo deciso di eseguire in tutto 9 test suddivisi per “categoria”: tre test sui costruttori, due test sugli algoritmi di sorting, due test sull’import e un test sull’algoritmo di raffinamento.

Test sui costruttori (TestConstructors):

1. TestVertex: verifica il corretto funzionamento del costruttore dei vertici,
2. TestEdge: verifica il corretto funzionamento del costruttore dei lati;
3. TestTriangle: verifica il corretto funzionamento del costruttore dei triangoli.

Test sugli algoritmi di sorting (TestSorting):

Sono stati eseguiti su vettori di int (vector<int>):

1. TestHeapSort: verifica la corretta implementazione dell’algoritmo dell’Heapsort;
2. TestInsertion: verifica la corretta implementazione dell’algoritmo dell’Insertion Sort.

Test sull’import (TestImport):

Abbiamo creato tre file fittizi contenenti vertici, lati e triangoli, inventati per i test:

* 0dtry.csv, contenente i vertici e avente la stessa struttura del file Cell0Ds.csv;
* 1dtry.csv, contenente i lati e avente la stessa struttura del file Cell1Ds.csv;
* 2dtry.csv, contenente i triangoli e avente la stessa struttura del file Cell2Ds.csv;

I test eseguiti sono i seguenti:

1. TestVertices: verifica la corretta importazione dei vertici;
2. TestEdges: verifica la corretta importazione dei lati;
3. TestTriangles: verifica la corretta importazione dei triangoli;

Test sull’algortimo di raffinamento (TestRefine):

1. TestBisect: verifica il corretto funzionamento della funzione Bisect.

Nonostante siano test unitari, abbiamo dovuto utilizzare i costruttori anche nei test sull’import e nel test sull’algoritmo di raffinamento perché le funzioni testate sono state definite per lavorare con oggetti da noi creati e quindi non per oggetti generici.

La funzione Bisect prende in input: il vettore ordinato di triangoli “sortedTriangles”, il vettore “trianglesList”, il triangolo da raffinare “triangle”, il vettore di vertici “verticesList”, il vettore di lati “edgesList”, il contatore “counter” (se != 0 il triangolo da triangle viene raffinato per mantenere la mesh ammissibile perché è adiacente a un triangolo precedentemente raffinato), il vettore di nuovi vertici “newVertices” e il vettore di lati nuovi “newEdges” (utili nelle successive chiamate per tenere traccia dei vertici e lati creati).

L’outiline della funzione è il seguente:

Nello step 1.a, i nuovi vertici, i nuovi lati frutto della bisezione e i nuovi triangoli vengono inseriti nelle rispettivamente nelle verticesList e newVertices, nelle edgesList e newEdges, nelle trianglesList e sortedTriangles. Inoltre, il lato bisezionato e il triangolo raffinato vengono spenti.

Lo step 1.b viene eseguito quando il triangolo che si sta raffinando è adiacente ad un triangolo precedentemente raffinato ed ha come lato maggiore un lato diverso rispetto a quello del triangolo adiacente. Bisogna quindi mantenere la mesh ammissibile: è necessario connettere il punto medio del lato bisezionato del triangolo adiacente (prec) al vertice opposto appartenente ad uno dei due triangoli creati raffinando il triangolo corrente. Per capire qual è il vertice giusto, abbiamo pensato a due metodi: usiamo il prodotto vettoriale tra il lato(opposite,[edgesList[toBisect].\_vertices[0]]) e il lato(prec,[edgesList[toBisect].\_vertices[0]]) per capire su quale lato giace “prec” , oppure calcoliamo le aree con segno di newT1 e newT3 (vedi figura REF): se il prodotto vettoriale è nullo oppure il prodotto delle aree di newT1 e newT3 > 0 , allora newT1 viene diviso in due nuovi triangoli, altrimenti newT2 viene diviso in due nuovi triangoli.

Nello step 2, i primi due casi indicano le condizioni di terminazione: il lato bisezionato si trova al bordo della mesh oppure il triangolo adiacente ha come lato più lungo il lato precedentemente bisezionato (in questo caso viene creato un nuovo lato per unire il punto medio del lato bisezionato a quello opposto); in entrambi i casi la mesh risulterà ammissibile.

Altrimenti, viene richiamata la funzione Bisect sul triangolo adiacente.