There are the following JAVA classes implementing the multi-frontal direct solver algorithm for finite difference problem:

- *A.java*
- *A1.java*
- *A2.java*
- *AN.java*
- *Aroot.java*
- *BS.java*
- *E2.java*
- *Eroot.java*
- *Executor.java*
- *Main.java*
- *P1.java*
- *P2.java*
- *P3.java*
- *Production.java*
- *Vertex.java*

We have already implemented all the classes for the exemplary solver execution for finite difference method with workflow corresponding to the mesh with 6 elements, as it is presented in Figures 1 and 2.

Now, we are going to replace the code to work with time-dependent problem, using Euler scheme with respect to time mixed with finite element method with linear basis functions for space.

Please replace system in *A1* by (0.1)

$$\begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} a_0^t \frac{h}{3} + a_1^t \frac{h}{6} - dt \left(-a_0^t \frac{1}{h} + a_1^t \frac{1}{h} - 1 + a_0^t \right) \\ a_0^t \frac{h}{6} + a_1^t \frac{h}{3} - dt \left(a_0^t \frac{1}{h} - a_1^t \frac{1}{h} \right) \end{bmatrix} \tag{0.1}$$

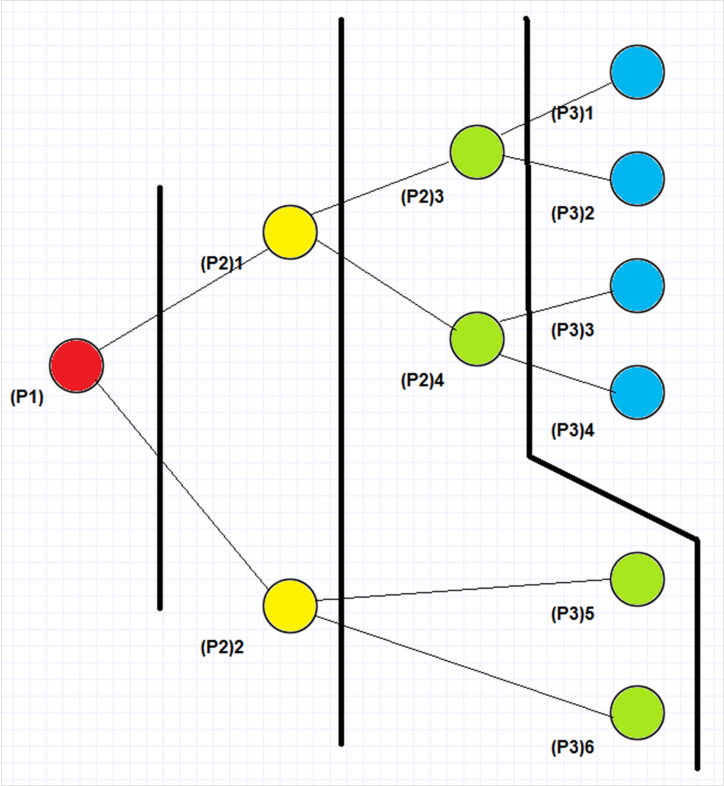

Figure 1: Graph of tasks responsible for construction of the elimination tree over the mesh with 6 elements.

replace system in *A* by (0.2)

$$\begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} \begin{bmatrix} u_{i-1} \\ u_i \end{bmatrix} = \begin{bmatrix} a^t_{i-1}\frac{h}{3} + a^t_i\frac{h}{6} - dt\left(-a^t_{i-1}\frac{1}{h} + a^t_i\frac{1}{h}\right) \\ a^t_{i-1}\frac{h}{6} + a^t_i\frac{h}{3} - dt\left(a^t_{i-1}\frac{1}{h} - a^t_i\frac{1}{h}\right) \end{bmatrix} \tag{0.2}$$

replace system in *AN* by (0.3)

$$\begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} \begin{bmatrix} u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} a^t_{N-1}\frac{h}{3} + a^t_N\frac{h}{6} - dt\left(-a^t_{N-1}\frac{1}{h} + a^t_N\frac{1}{h}\right) \\ a^t_{N-1}\frac{h}{6} + a^t_N\frac{h}{3} - dt\left(a^t_{N-1}\frac{1}{h} - a^t_N\frac{1}{h} + 1 + +a^t_N\right) \end{bmatrix} \tag{0.3}$$

To do that, it is necessary to add the element diameter h and the time step dt into the Java code. Additionally, the previous time step solution `x_old` must be stored. This should be done in the *Vertex* class:

```
1   class Vertex {
```

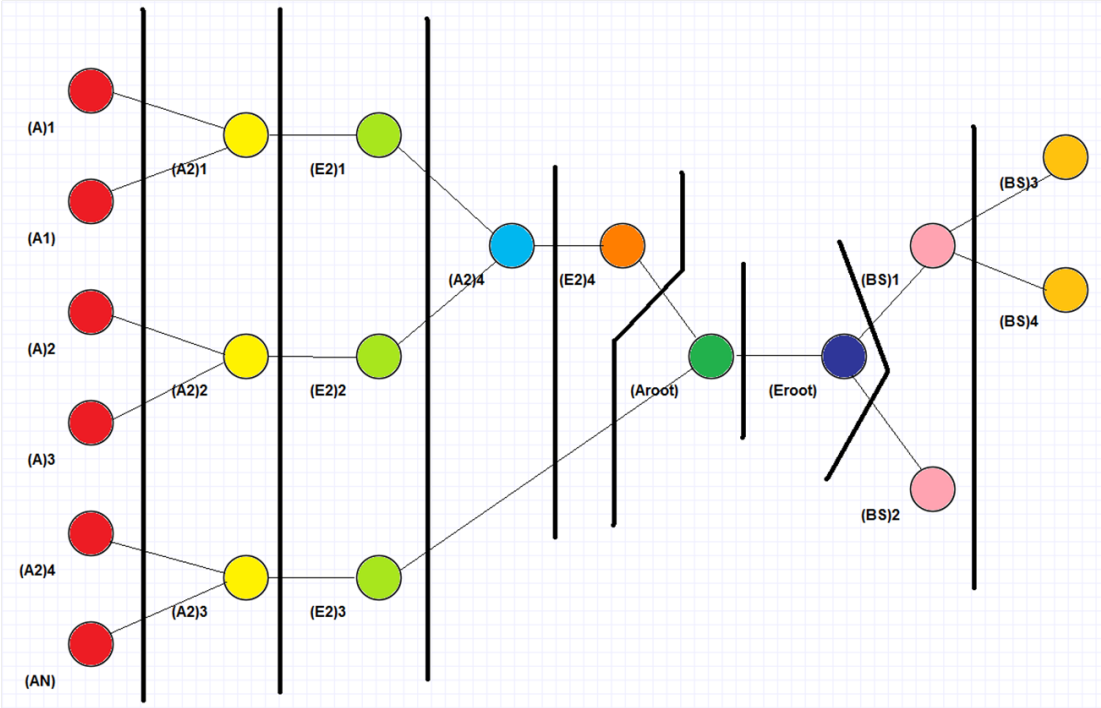

Figure 2: Graph of tasks responsible for execution of the solver algorithm over the constructed elimination tree.

```
2     Vertex(Vertex Left, Vertex Right, Vertex Parent, String Lab){
3       this.left=Left;
4       this.right=Right;
5       this.parent=Parent;
6       this.label=Lab;
7       a = new double[3][3];
8       b = new double[3];
9     }
10  String label;
11  Vertex left;
12  Vertex right;
13  Vertex parent;
14  double[][] a;
15  double[] b;
15  double[] x;
16  double[] x_old;
```

```
17  static double h;
18  static double dt;
19  void set_left(Vertex Left){left=Left;}
20  void set_right(Vertex Right){right=Right;}
21  void set_parent(Vertex Parent){parent=Parent;}
22  void set_label(String Lab){label=Lab;}
23  }
```

Hint: The *A1* updated production looks like this:

```
1   class A1 extends Production {
2     A1(Vertex Vert,CyclicBarrier Barrier){
3       super(Vert,Barrier);
4     }
5   Vertex apply(Vertex T) {
6       System.out.println("A1");
7       vert.a[1][1] = h/3.0;
8       vert.a[2][1] = h/6.0;
9       vert.a[1][2] = h/6.0;
10      vert.a[2][2] = h/3.0;
11      //copying values to x_old for next iteration
12      vert.x_old[0] = vert.x[0];
13      vert.x_old[1] = vert.x[1];
14      vert.x_old[2] = vert.x[2];
15      vert.b[1] = vert.x_old[1] * h / 3.0 + vert.x_old[2] * h / 6.0
16        - dt * (vert.x_old[1] / h - vert.x_old[2] / h - 1 + vert.x_old[1])
17      vert.b[2] = vert.x_old[1] * h / 6.0 + vert.x_old[2] * h / 3.0
18        - dt * (-vert.x_old[1]/h + vert.x_old[2]/h);
19      return vert;
20    }
21  }
```