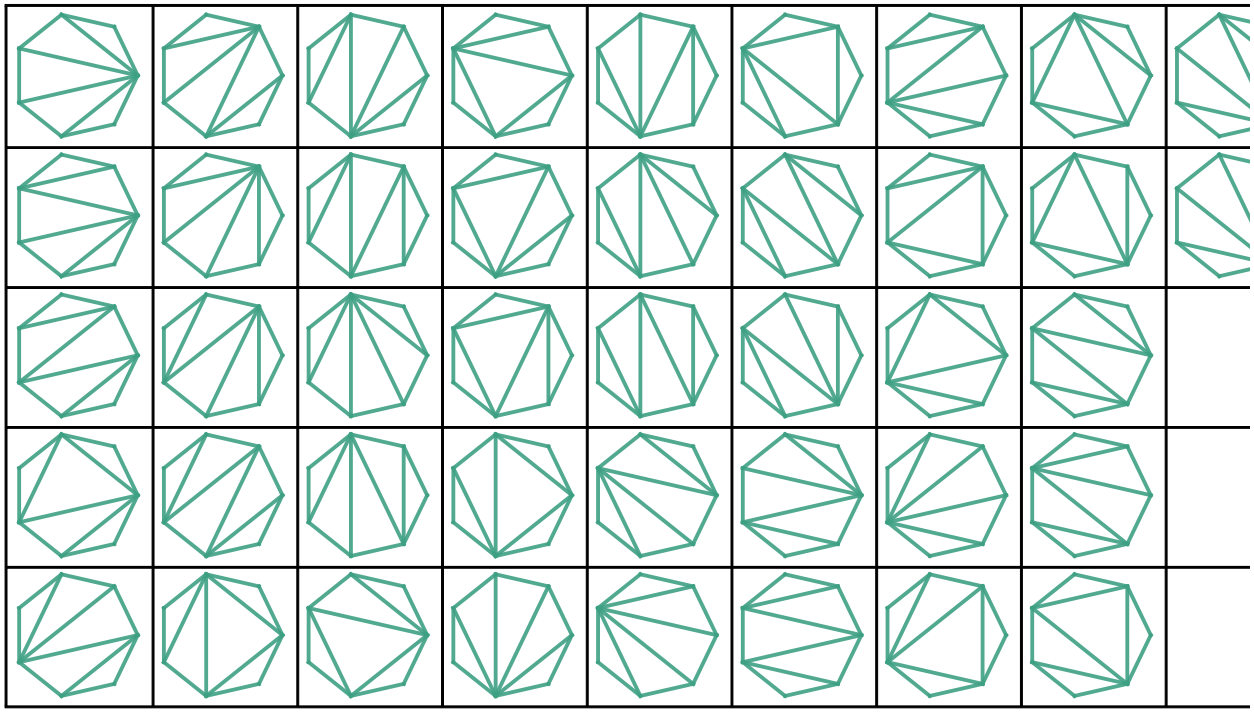
Code from Class

```
ts[n_Integer] := ts[Range[0, n + 1]];
ts[{_, _}] = {ds[]};
ts[vs_List] := Module[{l, r, k, t1, t2, tds},
  Union @@ Table[
    l = ts[Prepend[vs[[k ;;]], vs[[1]]]];
    r = ts[vs[[2 ;; k]]];
    Flatten[Table[
      tds = Join[t1, t2];
      If[k > 3, AppendTo[tds, d[vs[[2]], vs[[k]]]]];
      If[k < Length[vs], AppendTo[tds, d[vs[[1]], vs[[k]]]]];
      tds,
      {t1, l}, {t2, r}]],
    {k, 3, Length[vs]}
   ]
 ]
```

Drawing all triangulations of an n-gon

```
triangulations[n_Integer] := Module[{r, b, g},
  r = RandomReal[{0.0, 1.0}];
  b = RandomReal[{0.0, 1.0}];
  g = RandomReal[{0.0, 1.0}];
  lll = Table[
    Union[ds @@ Range[0, n + 1] /. j_Integer :> d[j, j + 1] /. n + 2 :> 0, s], {s, ts[n]}];
  Multicolumn[lll /. ds[ls___] :> Graphics[{Thick, RGBColor[r, b, g, 0.9], ls}] /.
     d[i_, j_] :> Line[{i, j}] /. j_Integer :> {Cos[2 π j / (n + 2)], Sin[2 π j / (n + 2)]},
   {Automatic, 10}, Frame → All, ItemSize → 5]
 ]
```

```
triangulations[5]
```



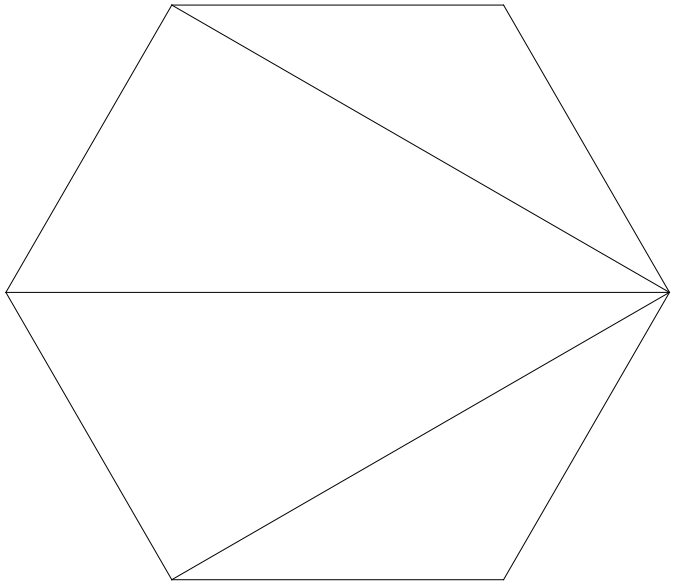
Code to draw one triangulations

```
poly[tr_ds, r_:0.0, b_:0.0, g_:0.0] := Module[{bb},
  bb = Union[
    ds @@ Range[0, Length[tr] + 2] /. j_Integer :> d[j, j + 1] /. Length[tr] + 3 :> 0, tr];
  Graphics[{Thick, RGBColor[r, b, g, 0.9], List @@ bb}] /. d[i_, j_] :> Line[{i, j}] /.
   j_Integer :> {Cos[2 π j / (Length[tr] + 3)], Sin[2 π j / (Length[tr] + 3)]}
 ]
```

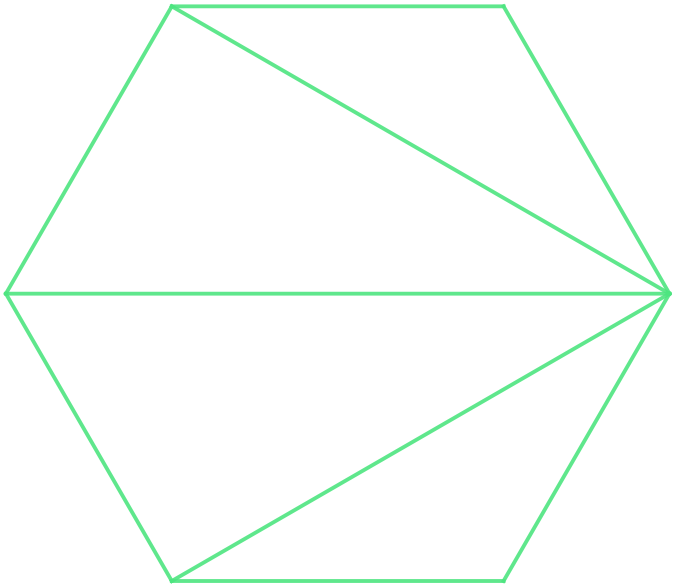
```
ts[4]
```

```
{ds[d[0, 4], d[0, 3], d[0, 2]], ds[d[0, 4], d[1, 3], d[0, 3]],
 ds[d[1, 3], d[1, 4], d[0, 4]], ds[d[1, 4], d[1, 3], d[1, 5]], ds[d[2, 4], d[0, 4], d[0, 2]],
 ds[d[2, 4], d[1, 4], d[0, 4]], ds[d[2, 4], d[1, 4], d[1, 5]], ds[d[2, 4], d[2, 5], d[0, 2]],
 ds[d[2, 4], d[2, 5], d[1, 5]], ds[d[3, 5], d[0, 3], d[0, 2]], ds[d[3, 5], d[1, 3], d[0, 3]],
 ds[d[3, 5], d[1, 3], d[1, 5]], ds[d[3, 5], d[2, 5], d[0, 2]], ds[d[3, 5], d[2, 5], d[1, 5]]}
```

```
poly[ds[d[0, 4], d[0, 3], d[0, 2]]]
```



```
poly[ds[d[0, 4], d[0, 3], d[0, 2]], 0.3, 0.9, 0.5]
```



Code to return data on binary tree

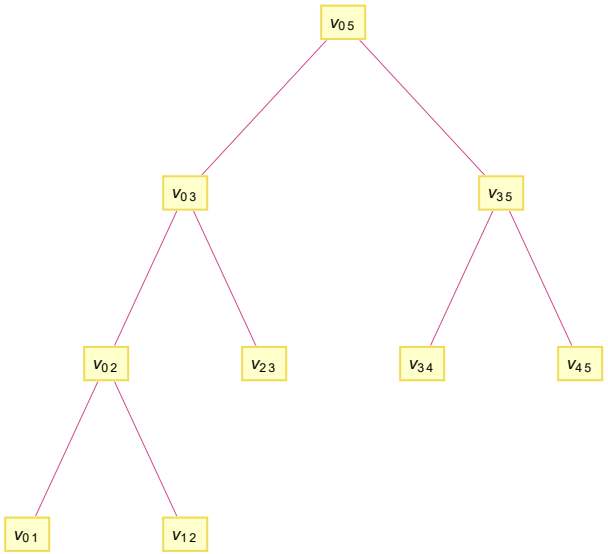
```
Btree6[bbb_ds] :=
 Module[{tree = {}, compt = {}, k = 0, rk = 0, rcompt, p, q, ngon, lk = 0},

  (*This part looks for common a point between two edges starting the edge d[0,5]*)
  ngon = Union[
    ds @@ Range[0, Length[bbb] + 1] /. j_Integer :> d[j, j + 1] /. Length[bbb] + 3 :> 0, bbb];
  min = Min[Range[0, Length[bbb] + 2]];
```

```
max = Max[Range[0, Length[bbb] + 2]];
compt =
 Select[Range[min, max], MemberQ[ngon, d[min, #]] && MemberQ[ngon, d[#, max]] &];
If[compt ≠ {},
 k = compt[[1]];

 (*Constructing a tree starting with the root d[0,5]*)
 AppendTo[tree, {d[min, max], d[min, k]}];
 AppendTo[tree, {d[min, max], d[k, max]}],
 AppendTo[tree, {d[min, max], d[min, max - 1]}];
 AppendTo[tree, {d[min, max], d[max - 1, max]}];
 k = max - 1
 ];

(*This table holds the right tree*)
Table[
  Table[
   If[MemberQ[bbb, d[p, q]],
    If[p ≤ k,
     rcompt = Select[Range[p, q], MemberQ[ngon, d[p, #]] && MemberQ[ngon, d[#, q]] &];
     If[rcompt ≠ {},
      rk = rcompt[[1]];
      AppendTo[tree, {d[p, q], d[p, rk]} ];
      AppendTo[tree, {d[p, q], d[rk, q]} ],
      AppendTo[tree, {d[p, q], d[p, q - 1]}];
      AppendTo[tree, {d[p, q], d[q - 1, q]}];
     ];
    ]
   ],
   {q, 0, k}
  ],
  {p, 0, k}
 ] // Flatten;

(*This table holds the left tree*)
Table[
  Table[
   If[MemberQ[bbb, d[m, n]],
    If[k ≤ m ≤ Length[bbb] + 2,
     rcompt = Select[Range[m, n], MemberQ[ngon, d[m, #]] && MemberQ[ngon, d[#, n]] &];
     If[rcompt ≠ {},
      lk = rcompt[[1]];
      AppendTo[tree, {d[m, n], d[m, lk]} ];
      AppendTo[tree, {d[m, n], d[lk, n]} ],
      AppendTo[tree, {d[m, n], d[m, n - 1]}];
      AppendTo[tree, {d[m, n], d[n - 1, n]}];
     ];
    ]
   ],
   {n, k, Length[bbb] + 2}
  ],
```

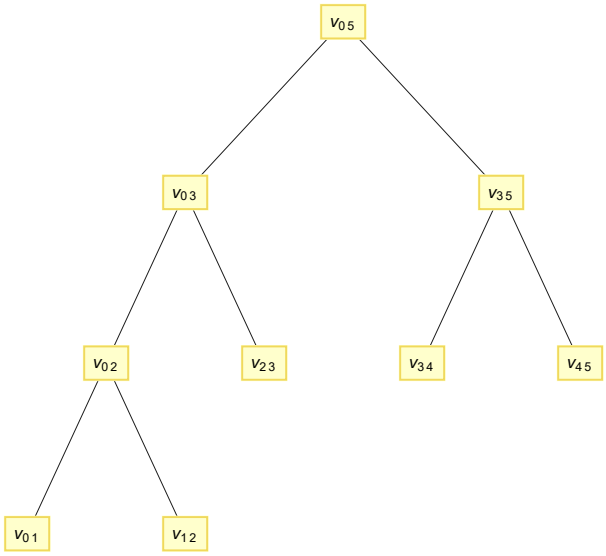
```
    {m, k, Length[bbb] + 2}
   ] // Flatten;
  tree
 ]
```

Drawing a single tree. It call the function Btree6

```
bintree[t_ds, r_: 0.0, b_: 0.0, g_: 0.0] := Module[{bt, bt1, bt2, min, max},
  min = Min[Range[0, Length[t] + 2]];
  max = Max[Range[0, Length[t] + 2]];
  bt = Btree6[t];
  bt1 = bt /. {d[i_, j_], d[k_, l_]} :> {i, j} → {k, l};
  bt2 = bt1 /. {x_, y_} :> v<sub>ToString[x] ToString[y]</sub>;
  TreePlot[bt2, Automatic, v<sub>ToString[min] ToString[max]</sub>,
   VertexLabeling → True, (*VertexRenderingFunction→
    ({White,EdgeForm[Black],Disk[#,.1],Black,Text[#2,#1]}&),*)
   EdgeRenderingFunction → Function[{p, vl, el}, {RGBColor[r, b, g, 0.9], Line[p]}]]
 ]

bintree[ds[d[3, 5], d[0, 3], d[0, 2]], 0.8, 0.2, 0.5]
```



```
bintree[ds[d[3, 5], d[0, 3], d[0, 2]]]
```



A poster of triangulations and trees

```
poster[tr_List] := Module[{t, pix, ngons, btrees, tree, r, b, g},
  pix = {};
  ngons = {};
  btrees = {};
  trian = Table[
    r = RandomReal[{0.0, 1.0}];
    b = RandomReal[{0.0, 1.0}];
    g = RandomReal[{0.0, 1.0}];
    AppendTo[ngons, poly[t, r, b, g]];
    tree = bintree[t, r, b, g];
    AppendTo[btrees, tree],
    {t, tr}
   ];
  {ngons, btrees}
 ]
```

```
GraphicsGrid[poster[ts[3]], Frame → All]
```