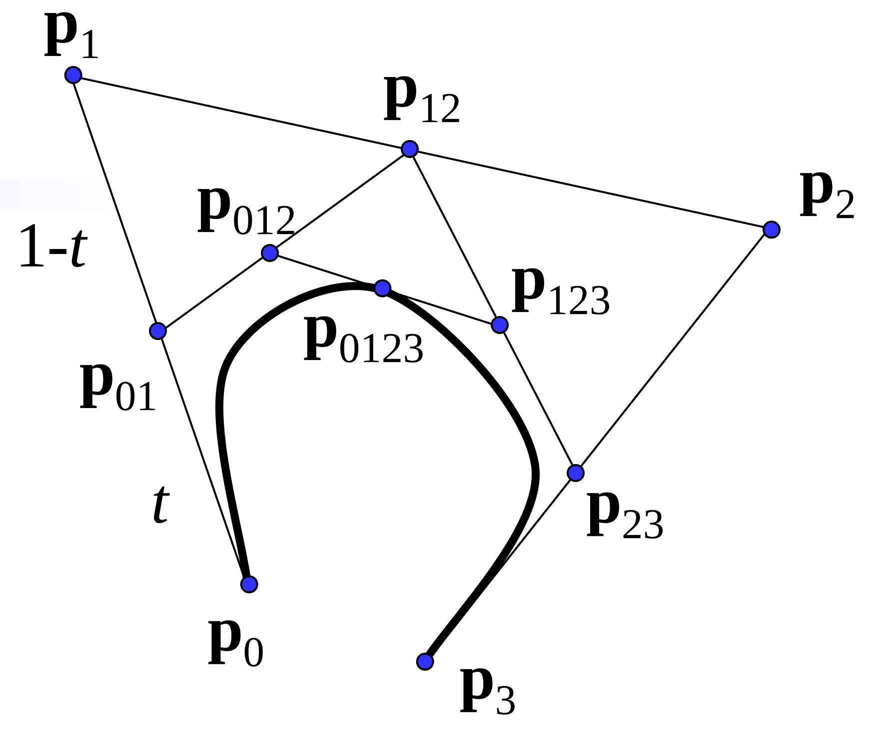
Blossoms

CS 419
Advanced Topics in Computer Graphics
John C. Hart

Borrowed somewhat from Tom Sederberg's notes

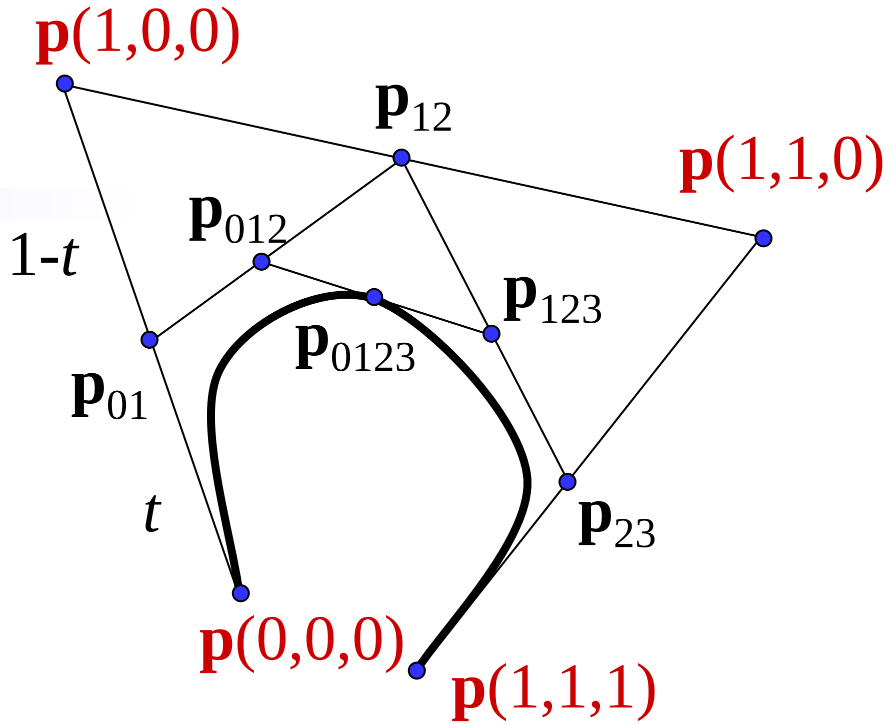
de Casteljau

- de Casteljau algorithm evaluates a point on a Bezier curve by scaffolding lerps
- Blossoming renames the control and intermediate points, like $\mathbf{p}_{12}$, using a polar form, like $\mathbf{p}(0,t,1)$



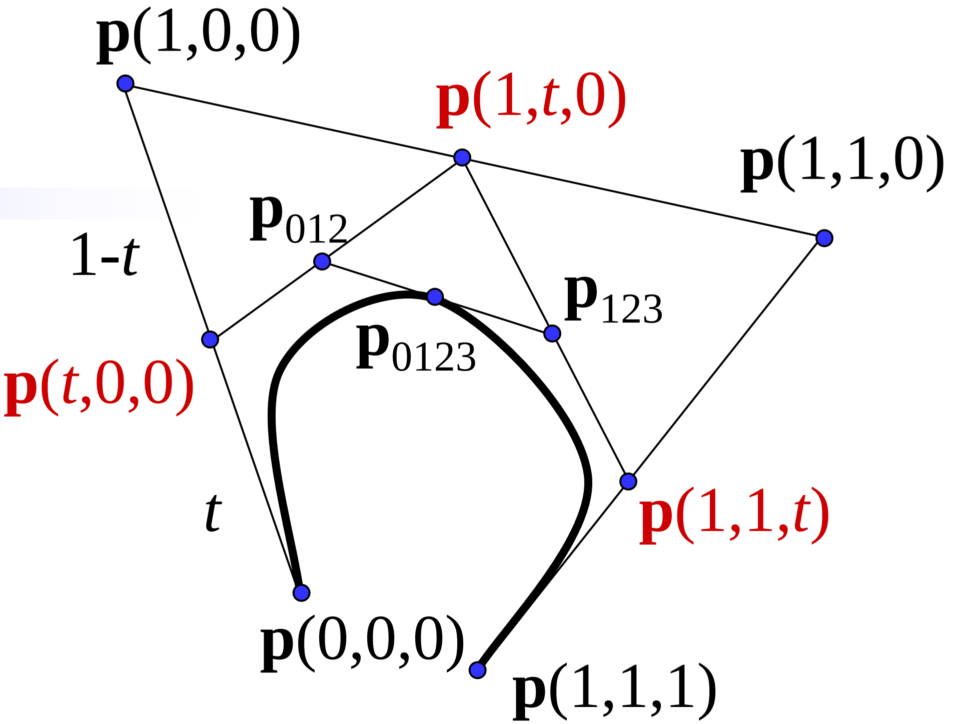
de Casteljau

- de Casteljau algorithm evaluates a point on a Bezier curve by scaffolding lerps
- Blossoming renames the control and intermediate points, like $\mathbf{p}_{12}$, using a polar form, like $\mathbf{p}(0,t,1)$



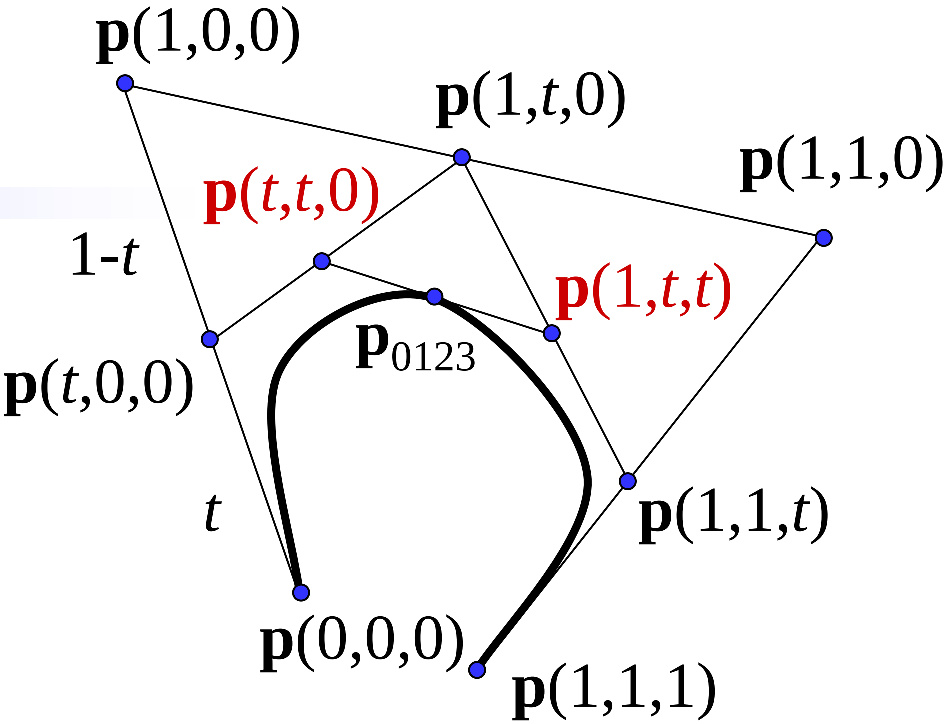
de Casteljau

- de Casteljau algorithm evaluates a point on a Bezier curve by scaffolding lerps
- Blossoming renames the control and intermediate points, like $\mathbf{p}_{12}$, using a polar form, like $\mathbf{p}(0,t,1)$



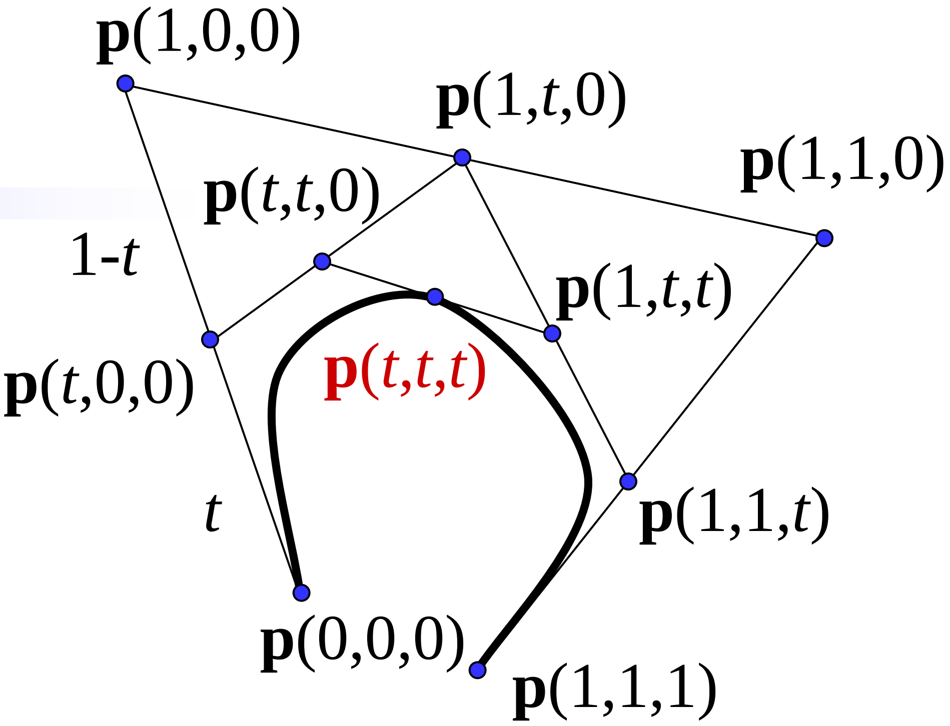
de Casteljau

- de Casteljau algorithm evaluates a point on a Bezier curve by scaffolding lerps
- Blossoming renames the control and intermediate points, like $\mathbf{p}_{12}$, using a polar form, like $\mathbf{p}(0,t,1)$

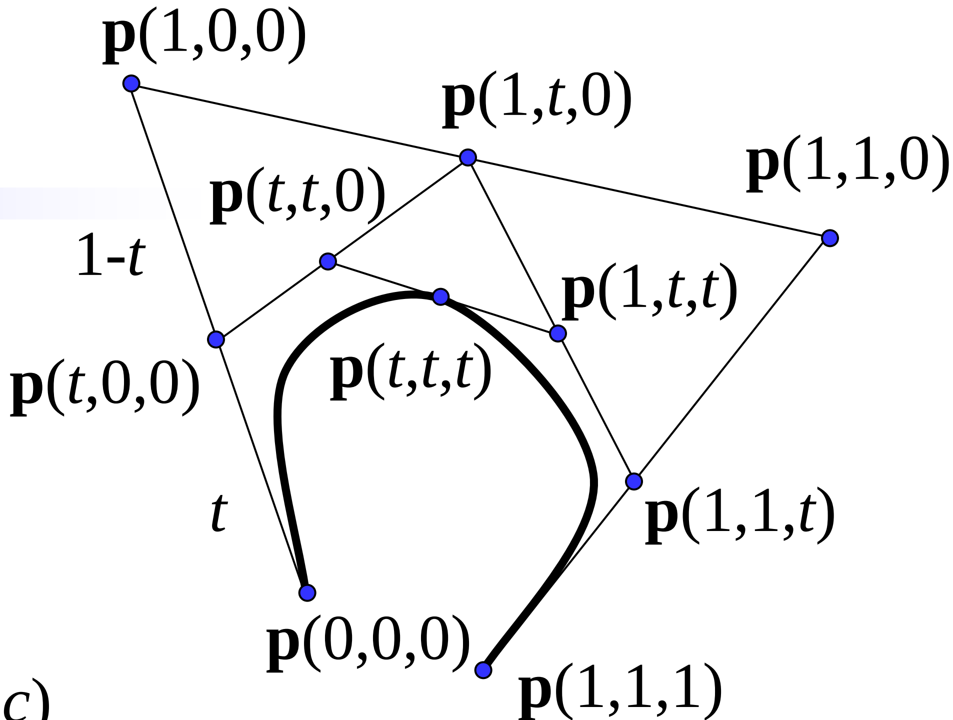


de Casteljau

- de Casteljau algorithm evaluates a point on a Bezier curve by scaffolding lerps
- Blossoming renames the control and intermediate points, like $\mathbf{p}_{12}$, using a polar form, like $\mathbf{p}(0,t,1)$



Blossoming Rules



- # of parameters = degree
- Order doesn't matter

  $\mathbf{p}(a,b,c) = \mathbf{p}(b,a,c)$

- Linear in any parameter

  $\alpha\mathbf{p}(a,b,c) = \mathbf{p}(\alpha a,b,c) = \mathbf{p}(a,\alpha b,c)$

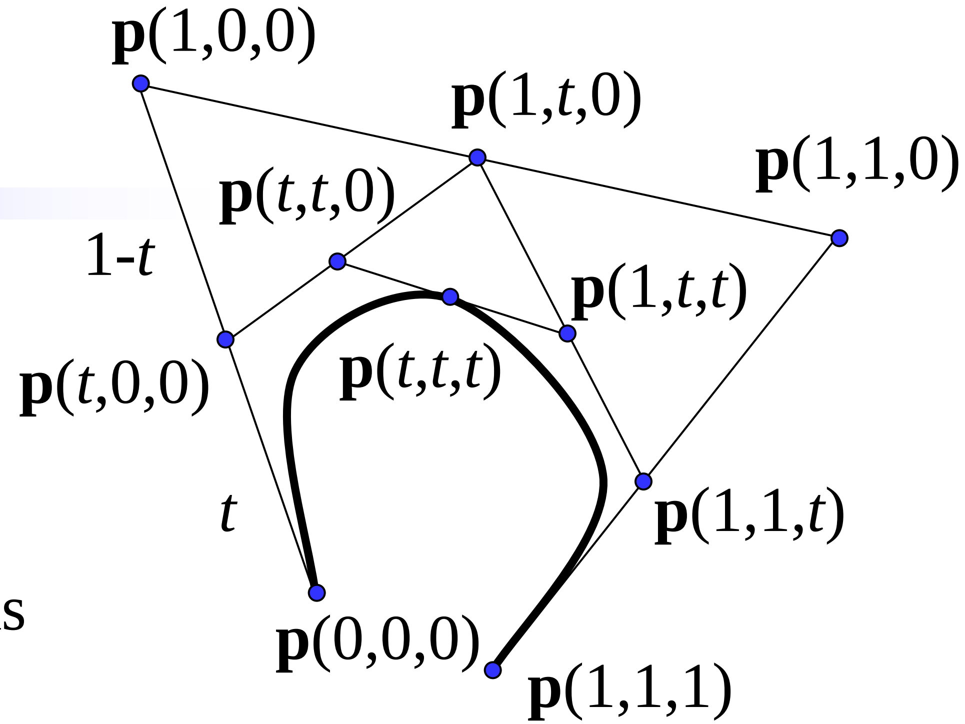
  $\mathbf{p}(a,b+c,d) = \mathbf{p}(a,b,d) + \mathbf{p}(a,c,d)$

- Lerping:

$$
\begin{aligned}
& (1-t)\,\mathbf{p}(0,0,0) + t\,\mathbf{p}(1,0,0) \\
= \quad & \mathbf{p}(0\,(1-t),0,0) + \mathbf{p}(1\,t,0,0) \\
= \quad & \mathbf{p}(0\,(1-t) + 1\,t,0,0) \\
= \quad & \mathbf{p}(t,0,0)
\end{aligned}
$$

Evaluation

- Goal is to find $\mathbf{p}(t)$ by diagonalization, by manipulating blossoms into $\mathbf{p}(t,t,t)$
- de Casteljau algorithm blossoms into Bernstein polynomials



$$\mathbf{p}(t) = \mathbf{p}(t,t,t) = (1\text{-}t)\,\mathbf{p}(t,t,0) + t\,\mathbf{p}(t,t,1)$$

$$= (1\text{-}t)[(1\text{-t})\,\mathbf{p}(t,0,0) + t\,\mathbf{p}(t,0,1)] + t\,[(1\text{-}t)\,\mathbf{p}(t,0,1) + t\,\mathbf{p}(t,1,1)]$$

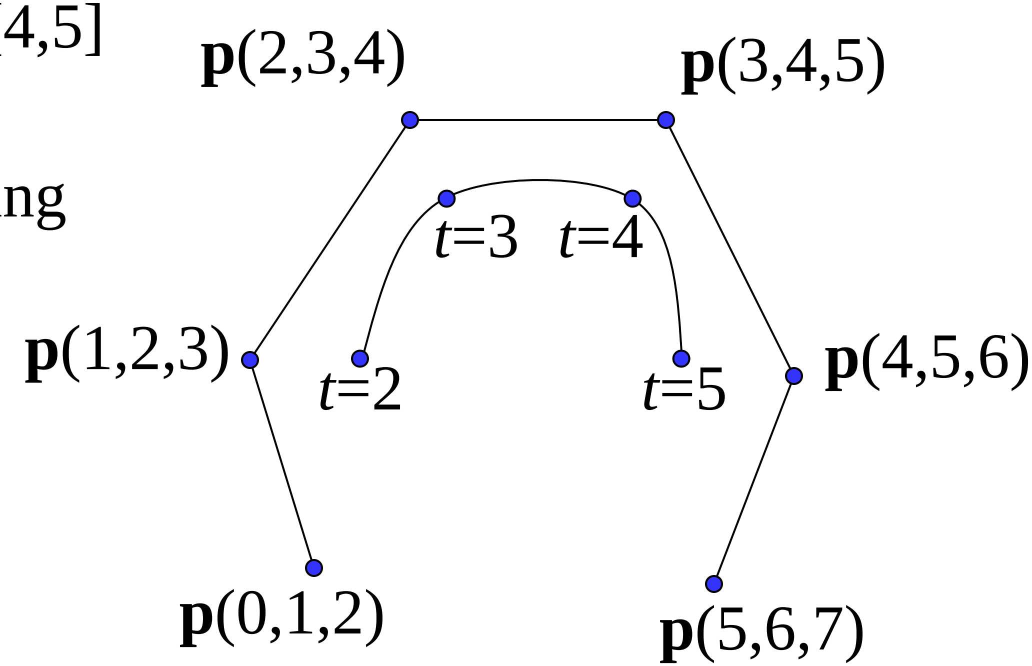
$$= (1\text{-}t)^2\,\mathbf{p}(t,0,0) + 2\,(1\text{-}t)\,t\,\mathbf{p}(t,0,1) + t^2\,\mathbf{p}(t,1,1)$$

$$= (1\text{-}t)^2[(1\text{-}t)\mathbf{p}(0,0,0)+t\mathbf{p}(1,0,0)]+2\,(1\text{-}t)t\,[(1\text{-}t)\mathbf{p}(0,0,1)+t\mathbf{p}(1,0,1)]+t^2[(1\text{-}t)\mathbf{p}(0,1,1)+t\mathbf{p}(1,1,1)]$$

$$= (1\text{-}t)^3\,\mathbf{p}(0,0,0) + 3\,(1\text{-}t)^2\,t\,\mathbf{p}(0,0,1) + 3\,(1\text{-}t)\,t^2\,\mathbf{p}(0,1,1) + t^3\,\mathbf{p}(1,1,1)$$

B-Spline Blossoms

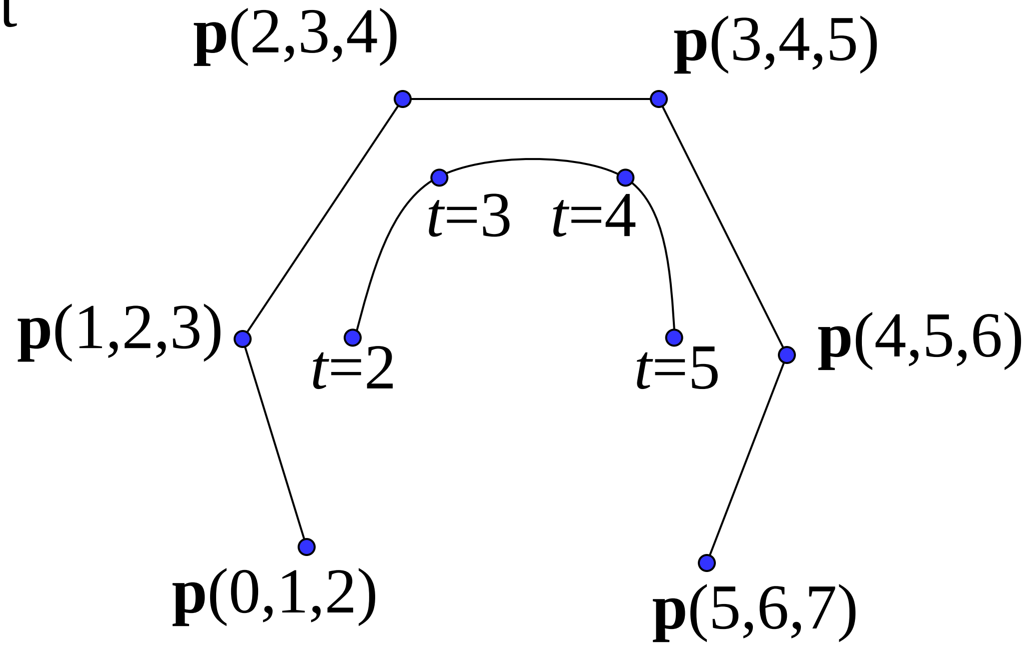
- Three segments: [2,3],[3,4],[4,5]
- Points within each segment influenced by four surrounding control points
- Knots influenced by three surrounding control points
- Need two extra knots at each end of knot vector (in general, *d*-1 extra knots at each end)



knot vector: [0 1 2 3 4 5 6 7]

Bohm Blossoms

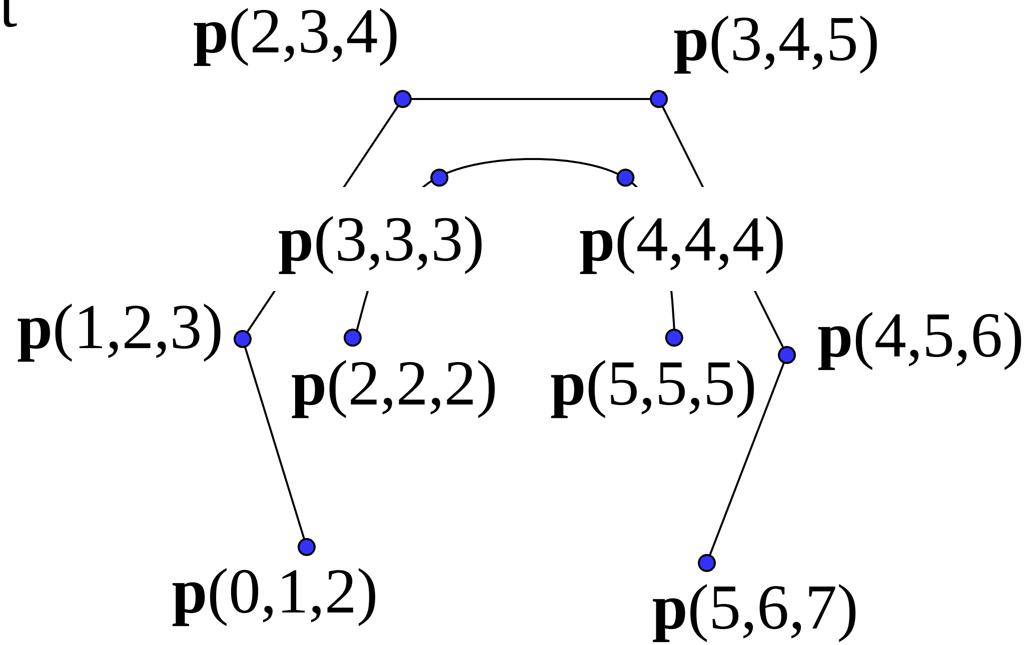
- Trick: Think of each segment as a Bezier curve



knot vector: [0 1 2 3 4 5 6 7]

Bohm Blossoms

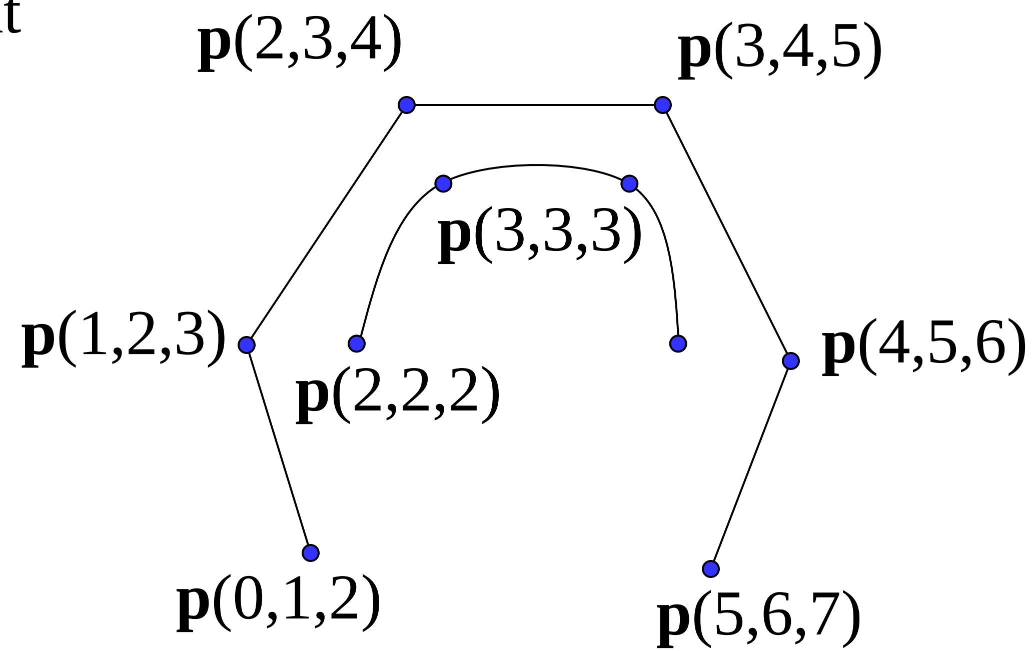
- Trick: Think of each segment as a Bezier curve



knot vector: [0 1 2 3 4 5 6 7]

Bohm Blossoms

- Trick: Think of each segment as a Bezier curve
- Where should the other two control points go for the [2,3] segment?



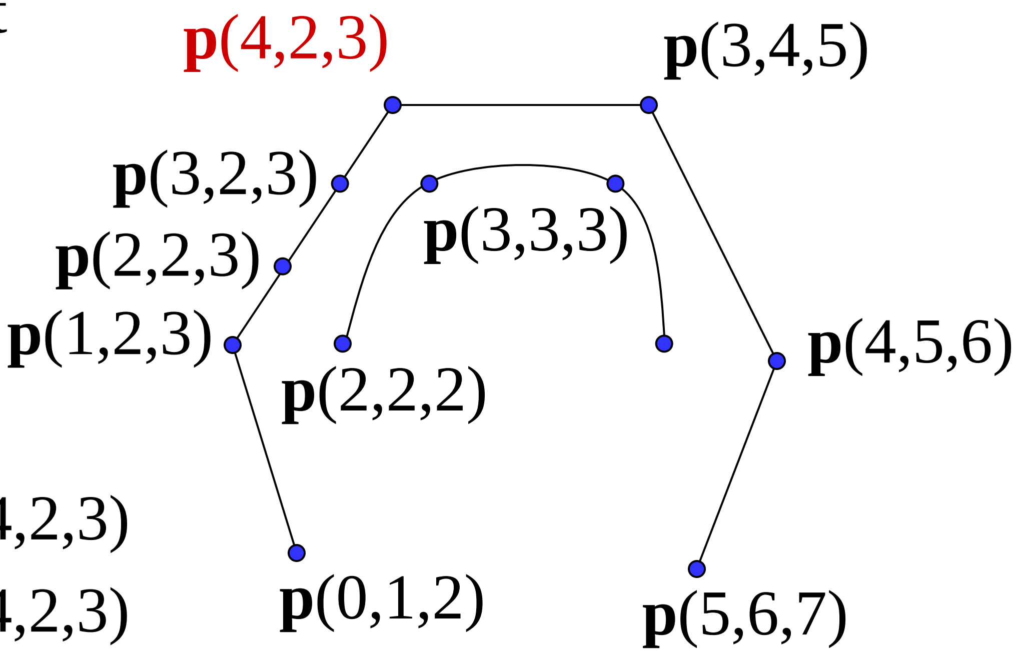
knot vector: [0 1 2 3 4 5 6 7]

Bohm Blossoms

- Trick: Think of each segment as a Bezier curve
- Where should the other two control points go for the [2,3] segment?
- Need to find:

$\mathbf{p}(2,2,3) = 2/3\ \mathbf{p}(1,2,3) + 1/3\ \mathbf{p}(4,2,3)$

$\mathbf{p}(3,2,3) = 1/3\ \mathbf{p}(1,2,3) + 2/3\ \mathbf{p}(4,2,3)$



knot vector: [0 1 2 3 4 5 6 7]

Bohm Blossoms

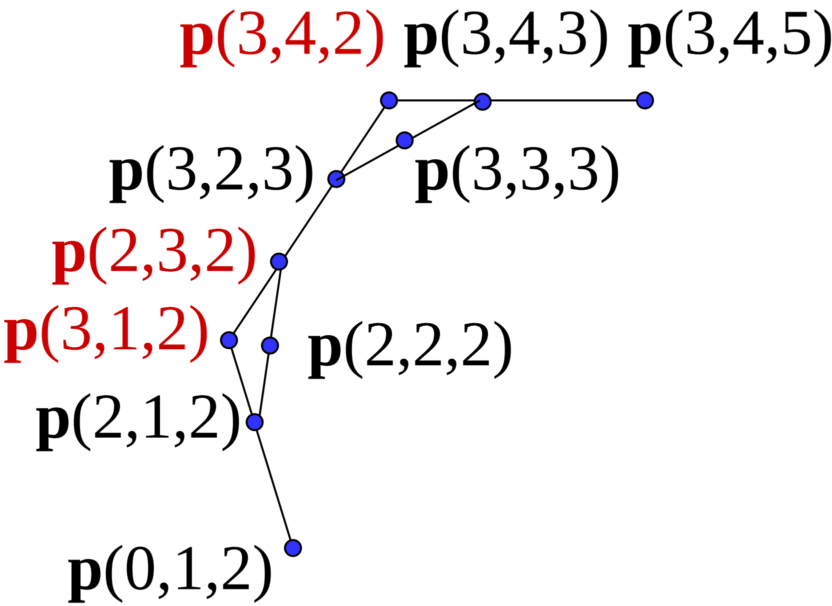
- Where are the endpoints located?
- Need to find:

p(2,1,2) = 1/3 **p**(0,1,2) + 2/3 **p**(3,1,2)

p(3,4,3) = 2/3 **p**(3,4,2) + 1/3 **p**(3,4,5)

p(2,2,2) = 1/2 **p**(2,1,2) + 1/2 **p**(2,3,2)

p(3,3,3) = 1/2 **p**(3,2,3) + 1/2 **p**(3,4,3)

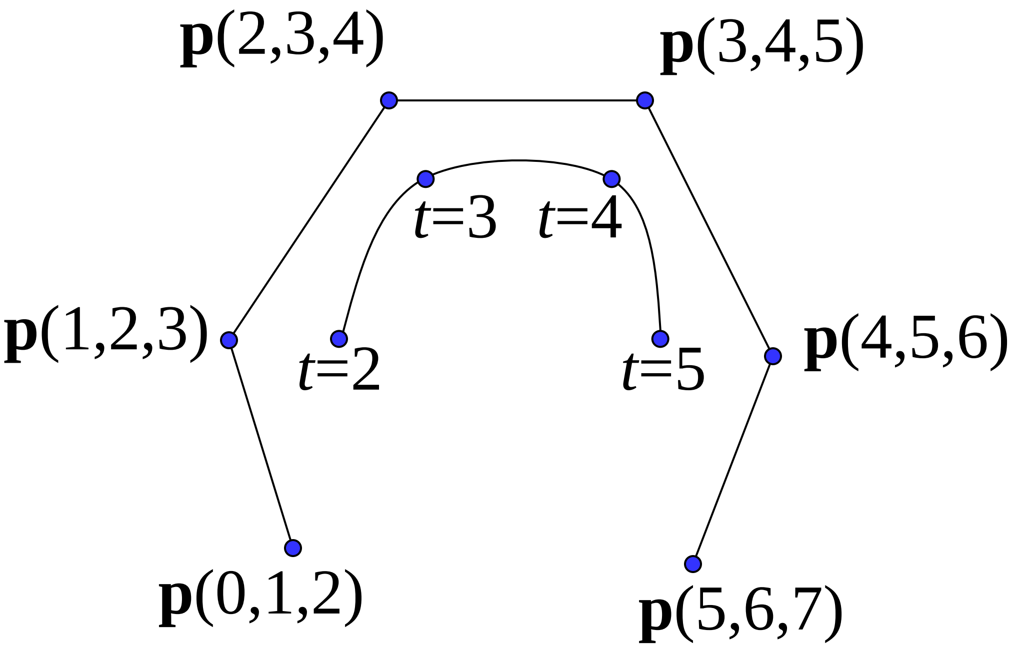


- Reveals how to turn a B-spline into a Bezier curve!

knot vector: [0 1 2 3 4 5 6 7]

Knot Insertion

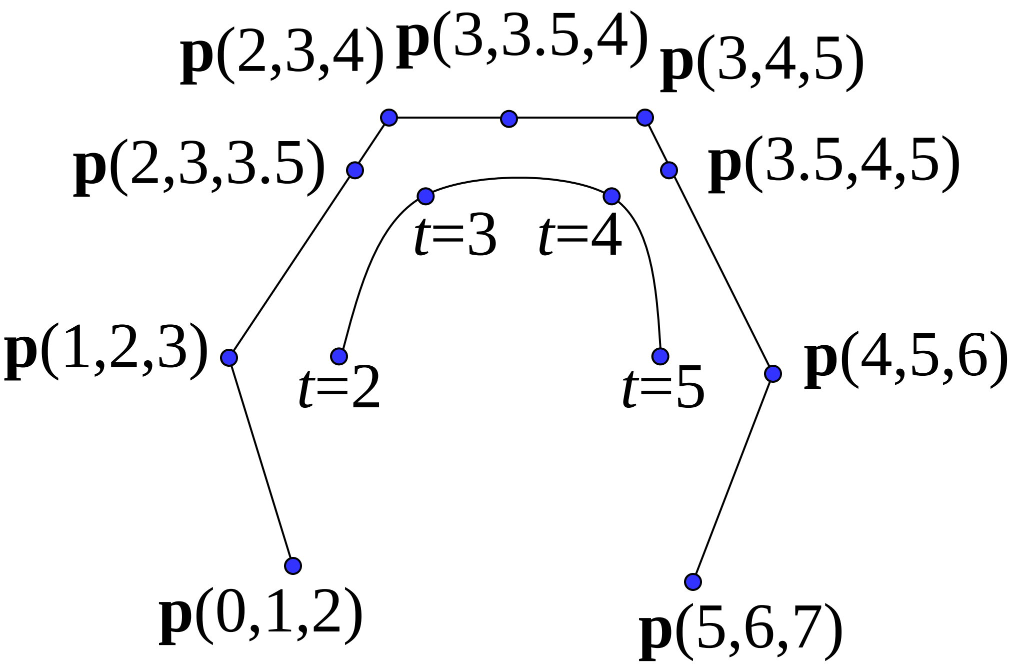
- Suppose we want to add a knot at t = 3.5



knot vector: [0 1 2 3 4 5 6 7]

Knot Insertion

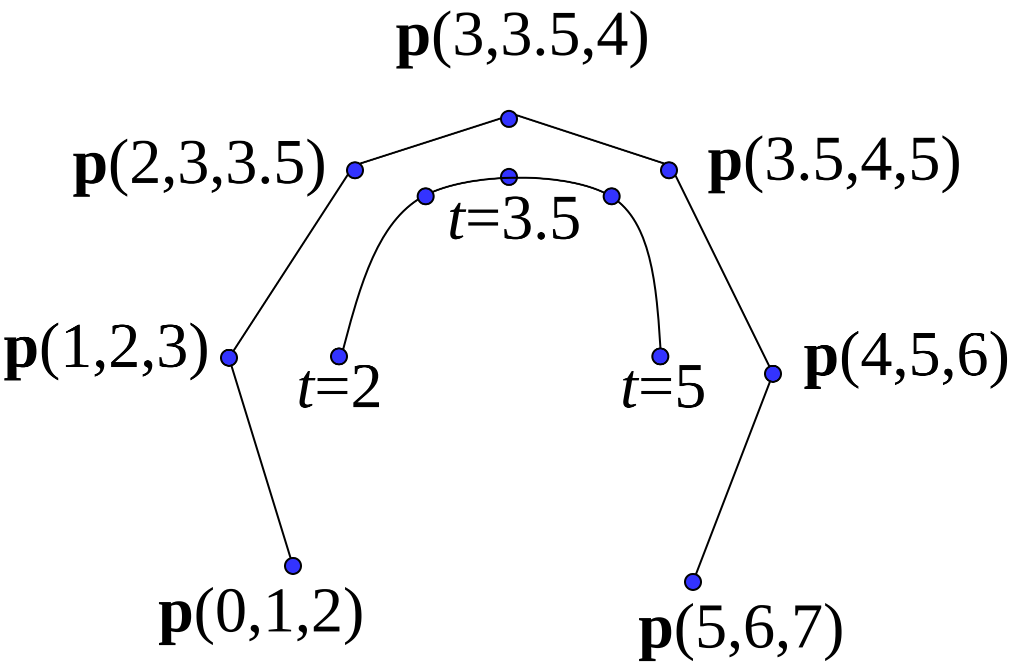
- Suppose we want to add a knot at t = 3.5
- Then we need new cp's **p**(2,3,3.5), **p**(3,3.5,4) and **p**(3.5,4,5) and can get rid of **p**(2,3,4) and **p**(3,4,5)



knot vector: [0 1 2 3 4 5 6 7]

Knot Insertion

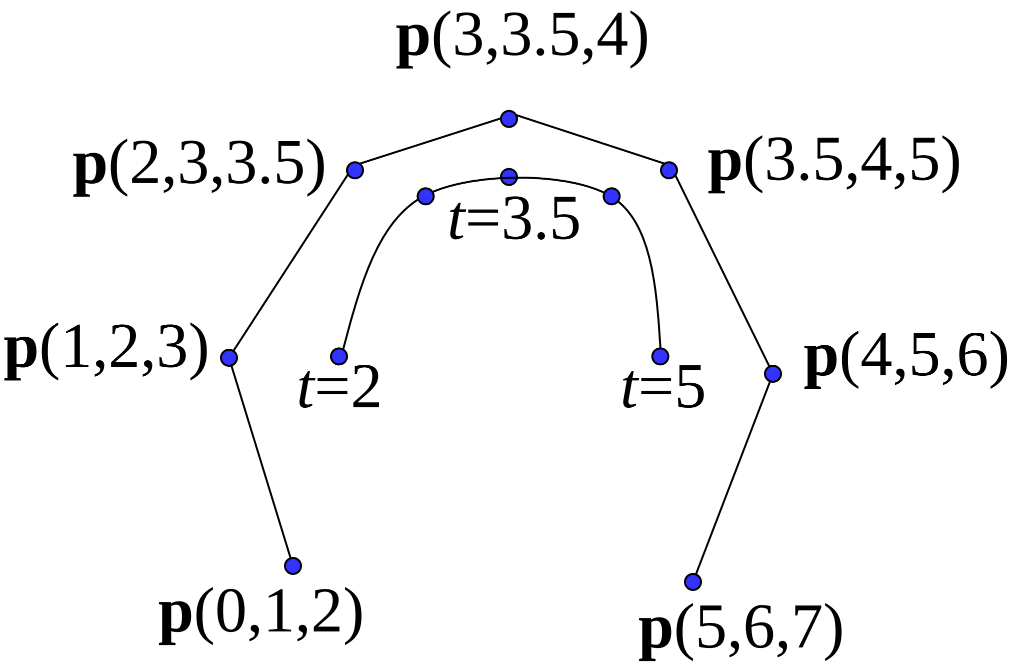
- Suppose we want to add a knot at t = 3.5
- Then we need new cp's **p**(2,3,3.5), **p**(3,3.5,4) and **p**(3.5,4,5) and can get rid of **p**(2,3,4) and **p**(3,4,5)



knot vector: [0 1 2 3 3.5 4 5 6 7]

de Boor Algorithm

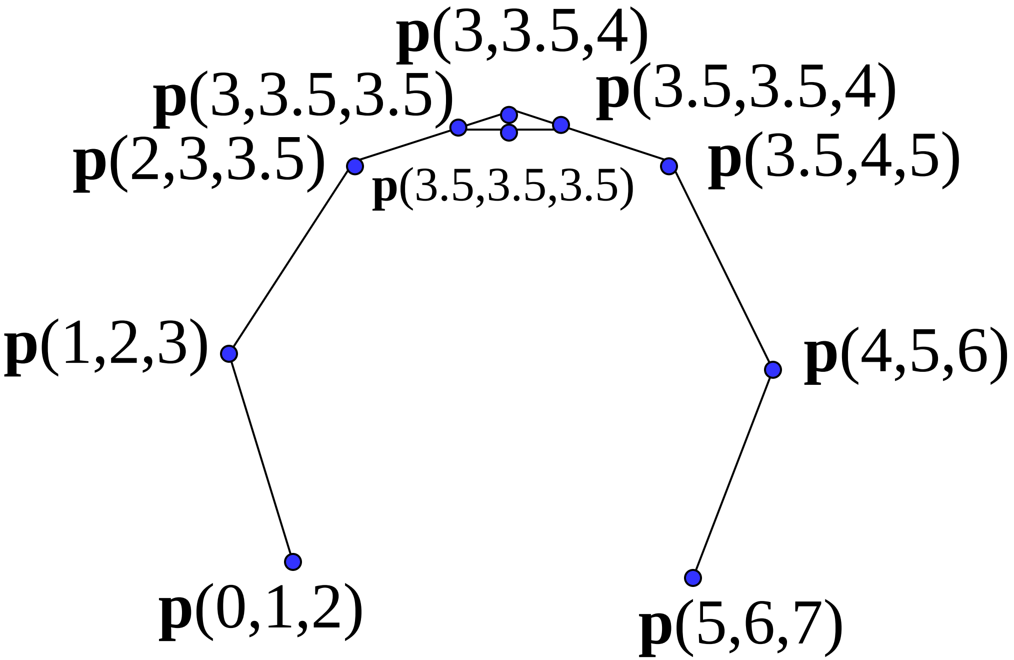
- What if we want to evaluate **p**(3.5)?
- Then create a triple knot at t = 3.5 and figure out where to put the control point **p**(3.5,3.5,3.5)



knot vector: [0 1 2 3 3.5 4 5 6 7]

de Boor Algorithm

- What if we want to evaluate **p**(3.5)?
- Then create a triple knot at t = 3.5 and figure out where to put the control point **p**(3.5,3.5,3.5)
- Need **p**(3,3.5,3.5) and **p**(3.5,3.5,4)
- Also subdivides B-spline into [0,1,2,3,3.5,3.5,3.5] and [3.5,3.5,3.5,4,5,6,7]



knot vector: [0 1 2 3 3.5 3.5 3.5 4 5 6 7]