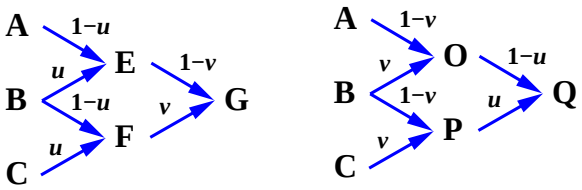
CS3621 Introduction to Computing with Geometry
Drill Exercises – Blossoming of Bézier Curves (I)

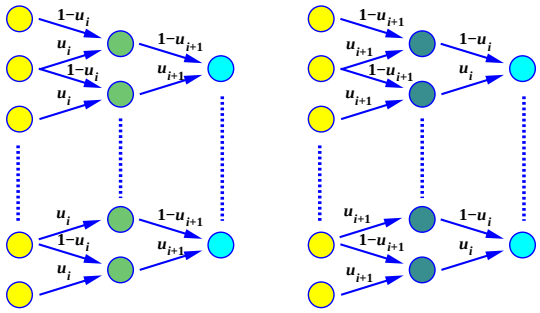
This set of drill exercises extends de Casteljau's algorithm and provides you with some initial information of the powerful *blossoming principle*. You **should** do all unmarked problems because they are very easy. Problems marked with [1] require simple algebraic manipulations. They are still easy. Problems marked with [2] require some mathematical background in discrete math and linear algebra. You are encouraged to work with others. **Similar problems will appear in Quiz 2 and the Final.**

1. De Casteljau's algorithm uses the same value u to compute every column. What if a different value is used for each column? More precisely, given three points $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and two values u and v, the left diagram below shows the way of computing $\mathbf{E}$ and $\mathbf{F}$ using u and $\mathbf{G}$ (from $\mathbf{E}$ and $\mathbf{F}$) using v. The right diagram shows the computation of using v to compute $\mathbf{O}$ and $\mathbf{P}$ and u to compute $\mathbf{Q}$ (from $\mathbf{O}$ and $\mathbf{P}$).



   Show that $\mathbf{G} = \mathbf{Q}$. This means the order of using u and v in the computation *does not* affect the final result. For convenience, we shall use $\mathcal{B}(u, v|\mathbf{A}, \mathbf{B}, \mathbf{C})$ to indicate the computation of using u first followed by v on the input $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. This is what the left diagram says. For the right diagram, we have $\mathcal{B}(v, u|\mathbf{A}, \mathbf{B}, \mathbf{C})$. Thus, what you need to prove is $\mathcal{B}(u, v|\mathbf{A}, \mathbf{B}, \mathbf{C}) = \mathcal{B}(v, u|\mathbf{A}, \mathbf{B}, \mathbf{C})$.

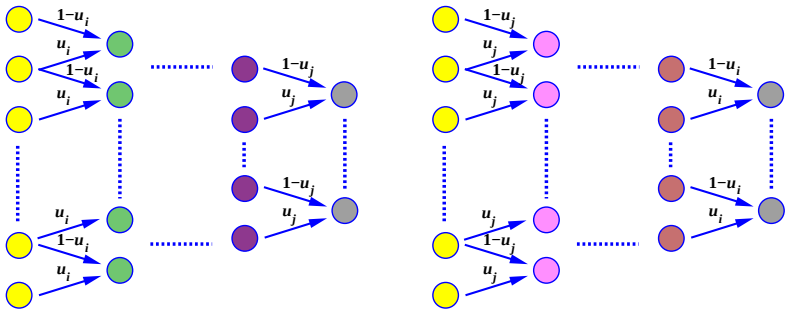
2. Suppose we are given $n + 1$ points $\mathbf{p}_0$, $\mathbf{p}_1$, ..., $\mathbf{p}_n$. De Casteljau's algorithm creates n columns to compute a point on a Bézier curve. We now use n values u_1, ..., u_i, u_{i+1}, ..., u_n, and write the result of this computation as $\mathcal{B}(u_1, u_2, \ldots, u_i, u_{i+1}, \ldots, u_n|\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$. Now interchange u_i and u_{i+1}. More precisely, we use u_{i+1} for computing the i-th column and u_i for the $(i + 1)$-th column. Denote the result as $\mathcal{B}(u_1, u_2, \ldots, u_{i+1}, u_i, \ldots, u_n|\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$. The following diagrams show the two involved columns and the values u_i and u_{i+1} used in the computation.



   Show that $\mathcal{B}(u_1, u_2, \ldots, u_i, u_{i+1}, \ldots, u_n|\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n) = \mathcal{B}(u_1, u_2, \ldots, u_{i+1}, u_i, \ldots, u_n|\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$. More precisely, if each column is computed with a different value, then interchanging two adjacent ones *does not* affect the final result. (**Hint:** Use (1))

3. [1]In fact, the results are still the same even though any two values are interchanged. More precisely, if the given computation is specified as $\mathcal{B}(u_1, \ldots, u_i, \ldots, u_j, \ldots, u_n|\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$. After interchanging u_i and u_j, the result is $\mathcal{B}(u_1, \ldots, u_j, \ldots, u_i, \ldots, u_n|\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$. The following diagrams show the involved columns and the positions of u_i and u_j. Show that $\mathcal{B}(u_1, \ldots, u_i, \ldots, u_j, \ldots, u_n|\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n) = \mathcal{B}(u_1, \ldots, u_j, \ldots, u_i, \ldots, u_n|\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$.

   Hint: Is this problem difficult? No, it is quite easy. Just use (2).

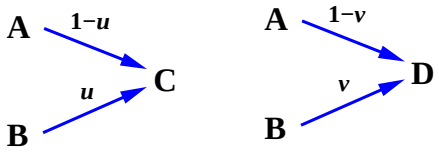


4. **[Symmetry]**[1] Show that any permutation of the parameters u's yields the same result. A permutation of n items is simply a reordering of these items. For example, $[a, b, d, c]$, $[d, a, b, c]$, $[b, d, c, a]$, $[d, c, b, a]$ are permutations of $[a, b, c, d]$.

   Hint: Consider "sorting" u_1, u_2, ..., u_n to the given permutation. How do you do sorting? Use (2) and/or (3).

5. **[Diagonal]** Given $\mathcal{B}(u_1, \ldots, u_i, \ldots, u_n | \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$, show that $\mathcal{B}(\underbrace{u, u, \ldots, u}_{n u\text{'s}} | \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$ is the point corresponding to u on the Bézier curve defined by control points $\mathbf{p}_0$, $\mathbf{p}_1$, ..., $\mathbf{p}_n$.

6. Suppose the following is used to compute $\mathbf{C}$ using u and $\mathbf{D}$ using v (from $\mathbf{A}$ and $\mathbf{B}$). That is, $\mathbf{C} = \mathcal{B}(u|\mathbf{A}, \mathbf{B})$ and $\mathbf{D} = \mathcal{B}(v|\mathbf{A}, \mathbf{B})$. Show that $\mathcal{B}((1-\alpha)u+\alpha v|\mathbf{A}, \mathbf{B}) = (1-\alpha)\mathcal{B}(u|\mathbf{A}, \mathbf{B})+\alpha\mathcal{B}(v|\mathbf{A}, \mathbf{B})$.



7. **[Multi-Affine]**[1] Suppose we have $\mathcal{B}(u_1, \ldots, u_i, \ldots, u_n | \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$ and two values α and β such that $\alpha + \beta = 1$ (*i.e.*, $\beta = 1 - \alpha$ or $\alpha = 1 - \beta$). Prove the following identify:

$$\mathcal{B}(u_1, \ldots, \alpha u + \beta v, \ldots, u_n | \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n) = \alpha\mathcal{B}(u_1, \ldots, u, \ldots, u_n | \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n) + \beta\mathcal{B}(u_1, \ldots, v, \ldots, u_n | \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$$

   where u, v and $\alpha u + \beta v$ are in the i-position replacing u_i.

   Hint: A simple algebraic manipulation with the help of (6) yields the result.

8. Show that $\mathbf{p}_i = \mathcal{B}(\underbrace{0, \ldots, 0}_{n-i}, \underbrace{1, \ldots, 1}_{i} | \mathbf{p}_0, \ldots, \mathbf{p}_n)$. More precisely, if we plug $n - i$ zeros followed by i ones into the "function" $\mathcal{B}()$, we get the i-th control point $\mathbf{p}_i$.

9. Recall that points on the first column computed by de Casteljau's algorithm are $\mathbf{p}_0^1$, $\mathbf{p}_1^1$, ..., $\mathbf{p}_{n-1}^1$, each of which is computed as

$$\mathbf{p}_i^1 = (1 - u)\mathbf{p}_i^0 + u\mathbf{p}_{i+1}^0$$

   where $\mathbf{p}_i^0$ and $\mathbf{p}_{i+1}^0$ are the northwest and southwest points on column zero (*i.e.*, the given control points). Show that

$$\mathbf{p}_i^1 = \mathcal{B}(\underbrace{0, \ldots, 0}_{n-i-1}, u, \underbrace{1, \ldots, 1}_{i} | \mathbf{p}_0, \ldots, \mathbf{p}_n)$$

   where $0 \le i \le n - 1$. Thus, replacing the last 0 of $\mathbf{p}_i$ obtained in the previous problem with u yields $\mathbf{p}_i^1$.

   Hint: Use (6) and/or (7).

10. **[De Casteljau Net]**[2] The points on the k-th column are computed from those on the $(k-1)$-th column as follows:

$$\mathbf{p}_i^k = (1-u)\mathbf{p}_i^{k-1} + u\mathbf{p}_{i+1}^{k-1}$$

Show that

$$\mathbf{p}_i^k = \mathcal{B}(\underbrace{0,\ldots,0}_{n-(k+i)},\underbrace{u,\ldots,u}_{k},\underbrace{1,\ldots,1}_{i}|\mathbf{p}_0,\ldots,\mathbf{p}_n)$$

Therefore, we can recover *every* intermediate point in the computation of de Casteljau's algorithm. To recover the i-th point on column k, add i ones to the end, preceded by k u's, and fill the remaining $n-(k+i)$ positions with zeros! Therefore, all given points on column 0 has no u's, all points on column one have one u, all points on the second column have two u's and so on. Moreover, the i-th point on any column has i 1's.

Hint: Use mathematical induction. Check your discrete math book if you have forgotten mathematical induction. You will need (3) or (4) and the multi-affine property (7).

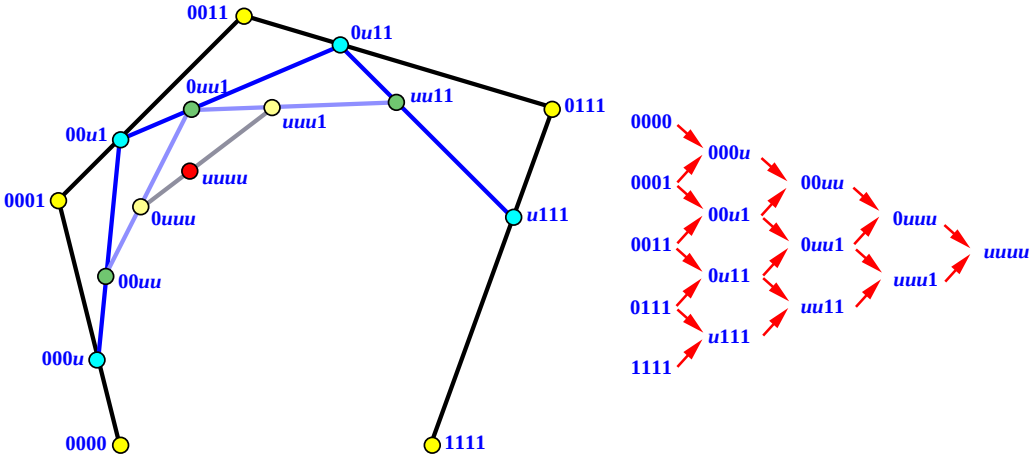
Let us study the impact of item 10 above. Let us use the arguments of function $\mathcal{B}()$ as labels. More precisely,

$$\mathbf{p}_i^k = \mathcal{B}(\underbrace{0,\ldots,0}_{n-(k+i)},\underbrace{u,\ldots,u}_{k},\underbrace{1,\ldots,1}_{i}|\mathbf{p}_0,\ldots,\mathbf{p}_n)$$

is replaced by a label

$$\underbrace{0\ldots0}_{n-(k+i)}\ \underbrace{u\ldots u}_{k}\underbrace{1\ldots1}_{i}$$

In the case of five points (*i.e.*, $n=4$) we have four columns in the computation of de Casteljau's algorithm and hence four arguments for "function" $\mathcal{B}()$. Thus, the labels of control points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_4$ are 0000, 0001, 0011, 0111 and 1111. The labels of the points on the first column are $000u$, $00u1$, $0u11$, $u111$; the labels of the points on the second column are $00uu$, $0uu1$, $uu11$; the labels of the points on the third column are $0uuu$ and $uuu1$; and, finally, the only points on the fourth column has label $uuuu$. The left diagram below shows the de Casteljau's net and the right one shows the computation triangle. Instead of using points, we use labels!



With this labeling scheme, we can find *any* point that is involved in the computation of de Casteljau's algorithm. For example, if we have $n+1$ control points and want to find the i-th point on column k (*i.e.*, $\mathbf{p}_i^k$), since this point has a label of

$$\underbrace{0\ldots0}_{n-(k+i)}\ \underbrace{u\ldots u}_{k}\underbrace{1\ldots1}_{i}$$

its coordinate is simply computed as

$$\mathbf{p}_i^k = \mathcal{B}(\underbrace{0,\ldots,0}_{n-(k+i)},\underbrace{u,\ldots,u}_{k},\underbrace{1,\ldots,1}_{i}|\mathbf{p}_0,\ldots,\mathbf{p}_n)$$

But, one important question remains: *What is the "function" $\mathcal{B}()$?* If we compute this function using de Casteljau's algorithm, then all intermediate points will be computed and become available. Hence, using this labeling scheme seems an overkill. Yes, you are right. There has to be some way for evaluating "function" $\mathcal{B}()$ efficiently without knowing the intermediate points. We shall cover this topic in the next drill exercise.