Department of Electronics Engineering
National Chiao Tung University
Algorithms, Fall 2017

November 9, 2017
Iris Hui-Ru Jiang

Programming Assignment #1: Bezier Curve

Due 11:59pm, December 11, 2017

1. Introduction

A Bézier curve is defined by a set of control points p_0 through p_n, where n is called its degree. The curve is given by

$$B(t) = \sum_{i=0}^{n} p_i B_{i,n}(t), t \in [0,1],$$

where $B_{i,n}(t)$ is a Bernstein polynomial, which is defined as follows.

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}.$$

Note that $t^0 = 1, (1-t)^0 = 1$, and that the binomial coefficient $\binom{n}{i} = \frac{n!}{i!(n-i)!}$. For example,

$B_{0,0}(t) = 1,$
$B_{0,1}(t) = 1 - t,\ B_{1,1}(t) = t.$
$B_{0,2}(t) = (1-t)^2,\ B_{1,2}(t) = 2(1-t)t,\ B_{2,2}(t) = t^2.$
$B_{0,3}(t) = (1-t)^3,\ B_{1,3}(t) = 3(1-t)^2 t,\ B_{2,3}(t) = 3(1-t)t^2,\ B_{3,3}(t) = t^3.$

A recursive definition of the Bézier curve of degree n expresses it as a point-to-point linear combination of a pair of corresponding points in two Bézier curves of degree n-1. Let $B_{p_0 p_1 \ldots p_n}$ denote the curve determined by points p_0, p_1, …, p_n. Then

$B_{p_0}(t) = p_0$ to start, and
$B(t) = B_{p_0 p_1 \ldots p_n}(t) = (1-t)B_{p_0 p_1 \ldots p_{n-1}}(t) + tB_{p_1 p_2 \ldots p_n}(t).$

2. Problem statement

2.1. Brief description

Write a program to generate the Bézier curve for a given set of points based on **dynamic programming**.

2.2. Input/output specification

The program should be invoked like this:

```
./bezier [test*.in] [test*.out]
```

- Input

The input file describes a set of control points (p_0, p_1, …, p_{N-1} **in order**) and a specified number of sampled points. The first line describes the total number N of control points. In the following N lines, each line indicates the x-coordinate and y-coordinate of one point. Finally, the last line describes the number M of sampled points.

Sample input

```
5
0 300
100 300
300 500
400 100
500 0
5
```

- Output

The program prints the M **evenly sampled** points on the Bezier curve **in order** (according to the input order). Each line indicates the *x*-coordinate and *y*-coordinate of a sampled point. The coordinates should have precisely 2 digits after the decimal point. (Use the format "%.2f").

Sample output

The sampled points are generated by sampling the Bezier curve at *t*=0.00, 0.25, 0.50, 0.75, and 1.00.

```
0     300
126.17    331.64
268.75    306.25
394.92    162.89
500  0
```

3. Evaluation

This program should be implemented based on **dynamic programming**; otherwise, you will be graded failed even if the results are correct. Each case is individually evaluated by the correctness. These output points will be compared with the correct answer in order. You may get partial credits. For example, if there are total 5 sampled points, and you report 4 correct sampled points, you will get 80 (=4/5*100) points. If the implementation takes more than **1 minute** to complete, this case will be graded fail even if the results are correct.

4. References

[1] http://mathworld.wolfram.com/BezierCurve.html

5. Submission

- ATTENTION: Plagiarism MUST be avoided. Every submission will be tested by a plagiarism catcher, running on all submissions from this class (and previous classes). If plagiarism is discovered, all students involved will receive only partial credit. Specifically, if the score received is x and there are n students involved, then the score for each student is x/n.
- Compress the source code, binary code, and a readme file to a zip file named as: StudentID_pa1.zip (e.g., 0015001_pa1.zip)
- Submit the zip file to e3