1999

(a) $$P(t) = \sum_{i=0}^{3} P_i \binom{3}{i} (1-t)^{3-i} t^i$$

$$= (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3$$

(i) C0-c'ty between $P(t)$ and $Q(t)$ implies that $P(1) = Q(0)$

$\therefore P_3 = Q_0$

(ii) C1-c'ty between $P(t)$ & $Q(t)$ implies [A] C0-c'ty
[B] $P'(1) = Q'(0)$

$$P'(t) = -3(1-t)P_0 + [3(1-t)^2 - 6(1-t)t]P_1 + [6(1-t)t - 3t^2]P_2 + 3t^2 P_3$$

$$P'(1) = Q'(0)$$
$$\Rightarrow -3P_2 + 3P_3 = -3Q_0 + 3Q_1$$

Geometrically: (i) is the condition that P_3 & Q_0 are co-incident
(ii) " " " " P_3 $(=Q_0)$ is the midpoint of the line segment $\overline{P_2 Q_1}$

A Bezier curve is guaranteed to lie within its convex hull because
(a) each point on the curve is ~~the~~ a weighted sum of the control points
(b) the weights always sum to 1
(c) none of the weights is negative

(b) There are many reasons for using multiple samples per pixel. Any sensible reason will be considered.

A selection are:

(i) for anti-aliasing: this attempts to ameliorate the objectionable aliasing artefacts; multiple rays are shot through the pixel in some sensible regular or pseudo-random pattern; the resulting values are averaged to give an overall value to the pixel.

(ii) for depth of field; the eye point is jittered over some area representing the lens ~~area of the vir~~ aperture of the virtual camera. Objects in the plane of the "screen" appear in focus; those forward or back appear out of focus.

(iii) for motion blur; the objects in the scene are animated; multiple samples are taken at different times in the animation and averaged to give the effect of motion blur. This, and the previous example are trying to mimic ~~(iv)~~ the ability of real world cameras.

(iv) for capturing specular highlights; multiple rays are spawned from a specular surface distributed according to the specular reflection equations; the allows (at great computational expense) for the simulation of accurate specular highlights.

(v) for area light sources; standard ray tracing light sources are ~~either~~ points (either local or at infinity). Area lights are approximated by ~~taking multiple~~ firing multiple rays at the light so as to adequately cover the area of the light.

(c) ~~Depth sort~~ Assuming that both algorithms take the same time to draw the actual polygons, the computational differences come in ~~the~~ determining the order in which to draw them.
Depth sort requires a sorting operation which, at best, is O(N log N) where N is the number of polygons; there are overheads when polygons need to be split.
BSP tree has an ~~[illegible]~~ O(N log N) pre-processing step to build the tree. This is roughly equivalent to the depth sort algorithm above. The drawing step is, however, only O(N).

If one had to choose between the two, one would choose BSP in a situation where the same model must be rendered many times — thus a single O(N log N) pre-processing step & many O(N) drawing steps c.f. many O(N log N) drawing steps for depth sort.
This is the important factor in choosing.