An Example of BPP Speedup: The Collision Problem

The keystone of any decent physics engine is collision detection. Suppose we have two rigid bodies A and B which are moving in space as time passes. My speculation is that these objects are typically represented as a collection of control points, which the graphics card uses as reference to render the actual object. From the standpoint of the physics engine then, the control points *are* the object. So we can regard the objects A and B abstractly as sets of control points.

There is an abstract mathematical problem called the collision problem which, with a little imagination, can be easily seen to aid the possible speedup of collision detection processes. In the interest of full disclosure, I should mention that this is not the typical motivation for thinking about this problem. The main use I've seen mentioned around the internet is in determining whether or not two graphs are isomorphic. But for the purposes of intuitive motivation we'll keep thinking about physics engines. Suppose we have a pair of rigid bodies which are represented in the physics engine by a set of N *control points*. Intuitively around half of these points will correspond to one rigid body and the rest will correspond to the other one. Let C_i be the set of positions of these points at time $t = 0$. Let C_f be the set of positions at time $t = 1$. This implicitly defines a function

$$f : C_i \to C_f \tag{1}$$

which maps all of the initial coordinates to their new coordinates after the game engine updates. For the sake of simplicity we will assume that the rigid bodies are the same 'size' (which in our context means they are represented by the same number of control points). Suppose two objects have collided. What does this mean for the control points? In a simplification of the problem, we can assume that what this mean is that *every control point of one object has collided with some other control point of the other object.* What this would mean for the function f is that it maps pairs of control points in C_i to the same point in space. Mathematically, this means that A and B collide if f is a **two to one function**. We say a function f is two to one if

$$\text{For all } a \in C_i, \text{ there exists a } b \in C_i \text{ such that } f(a) = f(b) \tag{2}$$

Of course, if the two objects don't collide, then f will be a one to one function on C. In case you forgot, a function f is **one to one** if

$$\text{For any } a, b \in C, a \neq b \implies f(a) \neq f(b) \tag{3}$$

So now we can express the collision problem with the proper motivation:

The Collision Problem: Suppose we have a function $f : C_i \to C_f$ and we are told in advance that f is either one to one or two to one. Which is it?

The Solution: Channeling the Birthday Problem

Recall from the setup that $|C_i| = N$. From context we know that $|C_f|$ is either N or $\frac{N}{2}$, but for the sake of simplicity we'll can pretend that it's size N in either case. (There just might be some points that aren't mapped to at all.) The best non-probabilistic algorithm would be to simply take a sample of $\frac{N}{2} + 1$ points in C_i, and check whether or not the function sends any of them to the same place. In order to be 100 percent sure that f is not two to one, we *need* to check this many points. (Why?) Thus, the complexity of the best deterministic algorithm for solving this problem is $O(\frac{N}{2}) = O(N)$.

However, suppose that we only have time to sample less than half of the elements in C_i. I can't be 100 percent sure from this sample of whether f is one to one or two to one, but as long as I'm more than 50 percent sure, I have a valid probabilistic algorithm! If the number of control points that I have to sample to be 50 percent sure is significantly smaller than half of N, then this will be a more powerful algorithm than the deterministic one.

Suppose I sample n control points uniformly at random from C_i. Label them $\{c_1, c_2, ..., c_n\}$. Let's define the events

$$B_n = \text{"At least two of the sampled points 'collide'."}$$
$$T = \text{"f is two to one"}$$

Let $t = \text{P(T)}$. Presumably t is some quantity which is fixed in advance. We don't know what it is, and we hopefully won't need to care what it is. But for the sake of being careful, we'll carry it around with us. Now we define the output of our program -

$$S = \begin{cases} 1 & \text{if } B_n \text{ occurs} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

S represents the output to our program. We know in advance that f is either one to one or two to one. We need to calculate the required sample size such that the probability that S is correct is greater than or equal to one half. For each $i = 1, 2, ..., n$, let

$$D_i = \text{"First i points in my sample do \textit{not} collide."}$$

Then

$$P(B_n|T) = 1 - P(\bigcap_{i=1}^{n} D_i|T) \tag{5}$$

Assuming that we chose our sample uniformly at random, all output points are equally likely, and given that f is two to one, there are $\frac{N}{2}$ possible (unique) outputs to choose from. I'm going to drop the $|T$ part to make this easier to look at, and I'm also going to define $M = \frac{N}{2}$. Then we can say the following:

$$\begin{aligned}
P(\bigcap_{i=1}^{n} D_i) &= P(D_1)P(D_2|D_1)P(D_3|D_1 \cap D_2)...P(D_n|\bigcap_{i=1}^{n-1} D_i) && (6)\\
&= (\frac{M}{M})(\frac{M-1}{M})(\frac{M-2}{M})...(\frac{M-(n-1)}{M}) && (7)\\
&= (1 - \frac{1}{M})(1 - \frac{2}{M})...(1 - \frac{n-1}{M}) && (8)\\
&\leq (e^{-\frac{1}{M}})(e^{-\frac{2}{M}})...(e^{-\frac{n-1}{M}}) && (9)\\
&= e^{-\frac{1}{M}\sum_{i=1}^{n-1} i} && (10)\\
&= e^{-\frac{(n-1)(n-2)}{2M}} = e^{-\frac{(n-1)(n-2)}{N}} && (11)
\end{aligned}$$

Thus

$$P(B_n|T) > 1 - e^{-\frac{(n-1)(n-2)}{N}} \tag{12}$$

Now let us consider the probability that S will return the correct answer. First consider the case when f is one to one. Then the probability that B_n occurs *must* be 0, so our program will return a 0, the correct answer, with certainty. Now suppose that f is two to one. Then, we can set the conditional probability that we just found greater than $\frac{1}{2}$...

$$(1 - e^{-\frac{(n-1)(n-2)}{N}})t > \frac{1}{2} \tag{13}$$
$$\implies (1 - e^{-\frac{(n-1)(n-2)}{N}}) > \frac{1}{2} \tag{14}$$
$$\implies e^{-\frac{(n-1)(n-2)}{N}} < \frac{1}{2} \tag{15}$$
$$\implies -\frac{(n-1)(n-2)}{N} < ln(\frac{1}{2}) \tag{16}$$
$$\implies (n-1)(n-2) > Nln(2) \tag{17}$$
$$\implies n^2 - 3n + 2 > Nln(2) \tag{18}$$
$$\implies n = O(\sqrt{Nln(2)}) = O(\sqrt{N}) \tag{19}$$

So as long as we choose a sample of size on the order of the square root of the number of control points, we have a probability greater than one half of outputting the correct answer to this decision problem!