Suppose we anticipate creating a file of (approximately) n records, where each record is R bytes in length. Suppose, further, that the operations we anticipate performing on the file are

(1) single record fetching (i.e., read/write a single record), and
(2) exhaustive sequential reading

We estimate that the probability of a given operation being an instance of (1) is p. It follows that the probability of a given operation being an instance of (2) is $1 - p$.

Using T_f and T_s to denote the expected running times for single instances of (1) and (2), respectively, we get that the expected running time for an operation is

$$T = p \cdot T_f \; + \; (1 - p) \cdot T_s$$

(Notice that this is simply a "weighted average" of the expected running times for (1) and (2), where the weights correspond to the relative frequencies with which (1) and (2) are performed.)

To fetch a single record, we read in the bucket containing the desired record. We assume that we have a pointer to (i.e., the address of) the first sector of the bucket containing that record, but that we do *not* have any information regarding where (i.e., in which sector(s)) within the bucket the sought-after record appears. (Thus, the entire bucket must be read into RAM to ensure that the sought-after record has been fetched!) Let us also make no assumptions regarding the physical location of a record to be fetched relative to the location of the record previously fetched. (This makes it appropriate to estimate the seek time for each single record fetch to be the "average seek time" for whatever disk drive we are using.) Thus, we get

$$T_f = s + r + btt$$

(See the figure at the end in case you need to be reminded of which quantities are denoted by s, r, etc.)

Regarding operation (2), we assume that the file is organized as a list of *buckets*. We make no assumption about the physical location of one bucket relative to the location of its logical successor. (This makes it appropriate to estimate the seek time for each bucket read to be the "average seek time" for whatever disk drive we are using.) In the context of such a file organization, sequentially reading the file corresponds to reading it "randomly by bucket". Hence, we get

$$T_s = bk \cdot (s + r + btt)$$

Now, we wish to derive a bucket size that minimizes the expected running time of an operation. In order to do this, we express the expected running time T of an operation as a function of bucket size, x, and we find the value of x at which T takes its minimum value.

Among the quantities referred to in the discussion above, only s, r, n, and R do not depend upon bucket size. Letting x denote the size of a bucket (in sectors), we now express the other quantities as functions of x.

The time to transfer a bucket comprising x sectors equals the time to transfer a single sector, stt, multiplied by x. That is,

$$btt(x) = stt \cdot x$$

As for the number of buckets occupied by the file, since a bucket of size x contains $S \cdot x$ bytes and each record is R bytes in length, we get that the number of records that fit into a bucket comprising x sectors is $\lfloor (S \cdot x)/R \rfloor$. Hence, the number of such buckets needed to store the entire file is[1]

$$\left\lceil \frac{n}{\lfloor (S \cdot x)/R \rfloor} \right\rceil$$

For the sake of simplicity (in particular, so that bk is a differentiable function), we ignore the applications of the floor and ceiling functions and write[2]

$$bk(x) = \frac{n \cdot R}{S \cdot x}$$

Going back to the above equations describing T_f, T_s, and T, we see that they can be expressed as functions of bucket size as follows:

$$
\begin{aligned}
T_f(x) &= s + r + btt(x) \\
T_s(x) &= bk(x) \cdot (s + r + btt(x)) \\
T(x) &= p \cdot T_f(x) + (1-p) \cdot T_s(x)
\end{aligned}
$$

By substituting for $btt(x)$, $bk(x)$, etc., according to their defining expressions and then using the distributive property of multiplication over addition (and the fact that $\frac{x}{x} = 1$ for $x \neq 0$), we get, for $x \neq 0$,

$$
\begin{aligned}
T(x) &= p \cdot (s + r + stt \cdot x) + (1-p) \cdot \frac{n \cdot R}{S \cdot x} \cdot (s + r + stt \cdot x) \\
&= p \cdot (s + r + stt \cdot x) + \frac{(1-p) \cdot n \cdot R}{S} \cdot \left(\frac{s+r}{x} + stt \right)
\end{aligned}
$$

Using a well-known method from differential calculus, we find the value of x that minimizes $T(x)$ by finding the roots of the derivative of T. Differentiating T with respect to x, we get

$$T'(x) = p \cdot stt - \frac{(1-p) \cdot n \cdot R \cdot (s+r)}{S \cdot x^2}$$

---

[1]This assumes that the buckets are filled to capacity. In circumstances of high *volatility* (i.e., in which many insertions and deletions of records occur), it would be more accurate to assume that *packing density* was only about 80%, in which case we should divide by 80% in order to arrive at a better estimate of how many buckets would be needed for storing the file.

[2]This tends to underestimate the number of buckets required to store the file. Indeed, what we have described is a lower bound on the number of buckets needed. It might be somewhat more accurate to use $\frac{n}{((S \cdot x)/R) - 1/2} + 1/2$ as the defining expression for $bk(x)$.

Setting $T'(x)$ to zero and applying standard algebraic manipulations yields

$$x^2 = \frac{(1 - p) \cdot n \cdot R \cdot (s + r)}{S \cdot p \cdot stt}$$

Thus, the roots of T are at $x = +c$ and $x = -c$, where

$$c = \sqrt{\frac{(1 - p) \cdot n \cdot R \cdot (s + r)}{S \cdot p \cdot stt}}$$

Because c is a positive number, we ignore the root $-c$. (We are interested in the value $T(x)$ only where $x > 0$, because it is not possible to choose the bucket size to be zero or negative!) To ensure that T's value at c is a minimum, rather than a maximum, we find $T''(x)$ (the second derivative of T) and verify that $T''(c)$ is positive (rather than negative). Letting d denote the value $((1 - p) \cdot n \cdot R \cdot (s + r))/S$, we find that $T''(x) = 2d/x^3$. Since both d and c are positive, so is $T''(c)$.

We conclude that the minimum value of $T(x)$, for x satisfying $x > 0$, occurs at $x = c$. Thus, we choose either $\lfloor c \rfloor$ or $\lceil c \rceil$ as our bucket size according to whether or not, respectively, $T(\lfloor c \rfloor) \leq T(\lceil c \rceil)$. (A bucket is an integral number of sectors, after all.)

R : size of each record (in bytes)
n : number of records in file
bk : number of buckets occupied by the file
S : number of bytes in a sector
stt: sector transfer time
btt: bucket transfer time
r : (average) rotational delay
s : (average) seek time
T_f: time to fetch a single record
T_s: time to read file sequentially (i.e., randomly by bucket)
T : expected time of an operation (weighted average of T_f and T_s)

Figure 1: Notation