Solutions Manual

Chapter 13

Section 13.1

Exercise 13.1.1

For this exercise we treat the sizes as if they were powers of 10 (i.e. petabyte is 10^{15}). Results would be slightly different if powers of 2 were used (i.e. petabyte is 2^{50}).

(a) Petabyte disk is 4000 times bigger than 250 gigabyte disk. Since $log_2 4000 \approx 12$, there must be about 12 doublings of the disk size. If each doubling occurs every 18 months, it would take $\frac{18 \cdot 12}{12} = 18$ years.

(b) Terabyte memory is 1000 times bigger than a gigabyte memory. Since $log_2 1000 \approx 10$, there must be about 10 doublings of memory. Thus, it would take about $\frac{18 \cdot 10}{12} = 15$ years.

(c) Terahertz processor is approximately 167 times faster than a 6 gigahertz (2 cores, 3 gigahertz each) processor. Thus, there must be approximately $log_2 167 \approx 7.4$ or about 8 doublings. So it would take about 12 years.
Note that recently the microprocessor design trend has shifted from increasing the operational frequency to increasing the number of cores. This is mainly due to the memory-wall and power-wall. However, if we assume that the operating frequency of the processor stays the same or grows at a much slower rate, and instead the number of cores doubles every 18 months, then the result would be equivalent (about 12 years).

(d) Year 2015 is 84 months from 2008, so there would be about $\frac{84}{18} \approx 5$ doublings. Therefore, a typical configuration would be:

- $6 \cdot 2^5 = 192$ gigahertz processor (closer to 64 core, 3 gigahertz each than to 2 cores, 96 gigahertz each).
- $250 \cdot 2^5 \approx 8$ terabyte disk.
- $1 \cdot 2^5 = 32$ gigabyte main memory.

Exercise 13.1.2

There would be $\frac{300 \cdot 12}{18} = 200$ doublings in the next 300 years. If a 2008 typical computer runs at 6 gigahertz (2 cores, 3 gigahertz each), then the Data's processor speed would be $6 \cdot 10^9 \cdot 2^{200}$ hertz. If we approximate $2^{10} \approx 10^3$, we get $6 \cdot 10^9 \cdot 10^{60}$ which is 6 duovigintillion hertz.

Section 13.2

Exercise 13.2.1

(a) Capacity of the disk is the product of 10 surfaces, times 100000 tracks, times 1000 sectors, times 1024 bytes, or 1024 gigabytes, which is about 1 terabyte.

(b) The average number of sectors per track is 1000, and each sector is 1024 bytes. So the average number of bytes per track is 1024000 bytes or 8192000 bits. The outermost track has length of $3.5 \cdot \pi \approx 11$ inches. 20% of the track are gaps, so data occupies 8.8 inches. The average density of the outermost track is then 930909 bits per inch or roughly 0.89 megabits per inch.
The innermost track length is $1.5 \cdot \pi \approx 4.7$ inches. 20% of the track are gaps, so the data occupies about 3.76 inches, and the average density is about 2178723 bits per inch or, roughly, 2 megabits per inch. The average density of bits in the sectors of a track is then about $\frac{0.89 + 2}{2} = 1.445$ megabits per inch.

(c) The maximum seek time occurs when the head needs to move across all tracks (e.g. from the outermost track to the innermost). Thus, the maximum seek time is $1 + 0.0002 \cdot 99999 \approx 21$ ms.

(d) The maximum rotational latency would be a full rotation of the disk (e.g. when the head arrived just as the start of the needed sector passed the head). Thus, the maximum rotational latency is $\frac{60 \cdot 1000}{10000} = 6$ ms.

(e) We can use the track as an approximation of the circle that the head needs to travel. Thus, there are a total of 1000 sectors and 1000 gaps per circle. Since gaps occupy 20% of the circle, they cover $360° \cdot 0.2 = 72°$ of the circle, and so each gap covers $\frac{72°}{1000} = 0.072°$ of the circle arc. Similarly, each sector covers 0.288° of the arc. If the block occupies 64 sectors, the head must pass over 64 sectors and 63 gaps between them. Thus, the total degrees that the head needs to cover is is $64 \cdot 0.288 + 63 \cdot 0.072 = 22.968$. A 10000 rpm disk will make one rotation in $\frac{60 \cdot 1000}{10000} = 6$ ms. Thus, the disk will cover one degree in $\frac{6}{360} = \frac{1}{60}$ ms. Therefore, the transfer time for one block is $\frac{22.968}{60} = 0.3828$ ms.

(f) Example 13.2 mentions that the average distance traveled by the head is $\frac{1}{3}$ of the way across the disk (Exercise 13.2.3 proves this). Therefore, the average seek time is $1 + 0.0002\left(\frac{100000}{3}\right) \approx 7.67$ ms.

(g) Average rotational latency is the time to rotate the disk half way around, which is 3 ms.

Exercise 13.2.2

The next request can be for a block on any cylinder with equal probability. Therefore, the average number of cylinders the head will need to travel is the product of the sum: 0 (if the request is for the block on cylinder 8192) + (1+2+...+8191) (if the request is for the block to the left of cylinder 8192) + (1+2+...+(65536-8192)) (if the request is for the block to the right of the cylinder 8192), over the total number of cylinders 65536. That is, the average number of cylinders the disk head will need to move is $\frac{\frac{8191 \cdot 8192}{2} + \frac{57344 \cdot 57345}{2}}{65536} = 25600$.
The time it takes Megatron 747 to move 25600 cylinders is $1 + 0.00025 \cdot 25600 =$

7.4 ms. Therefore the average time to read a block would be 7.4 + 4.17 + 0.13 = 11.7 ms.

Exercise 13.2.3

If the disk head is at cylinder 1, the average distance the head will move is $\frac{(1 + 2 + 3 + ... + 65535)}{65536}$. If the disk head is at cylinder 2, the average distance the head will move is $\frac{(1 + 1 + 2 + 3 + ... + 65534)}{65536}$ and so on. We notice that the expressions a above consist of the two separate sums, that change as our starting point for the head changes. in other words, we have the sum of all track to the left of the head and the sum of the tracks to the right of the head. For instance, when the head is at cylinder 1, the sums are 0 and the sum of all the distances to all cylinders $(1 + 2 + 3 + ... + N - 1)$. And when the head is at cylinder 2, the sums are 1 and $(1 + 2 + 3 + ... + N - 2)$. In general, we can see that this can be expressed as $\frac{(x-1)(x-1+1)}{2N} + \frac{(N-x)(N-x+1)}{2N}$. Furthermore, we have N such sums, one for each possible head position, and therefore, the average distance is

$$\begin{aligned}
&\frac{1}{N}\sum_{x=1}^{N}\left(\frac{(x-1)(x)}{2N}+\frac{(N-x)(N-x+1)}{2N}\right)= \\
&\quad=\frac{1}{2N^2}\sum_{x=1}^{N}((x-1)(x)+(N-x)(N-x+1)) \\
&\quad=\frac{1}{2N^2}\sum_{x=1}^{N}\left(2x^2-(2N+2)x+N+N^2\right) \\
&\quad=\frac{1}{2N^2}\left(2\sum_{x=1}^{N}x^2-(2N+2)\sum_{x=1}^{N}x+N^2+N^3\right) \\
&\quad=\frac{1}{2N^2}\left(2\frac{N(N+1)(2N+1)}{6}-(2N+2)\frac{N(N+1)}{2}+N^2+N^3\right) \\
&\quad=\frac{(N+1)(2N+1)}{6N}-\frac{(N+1)^2}{2N}+\frac{N}{2N}+\frac{N^2}{2N} \\
&\quad=\frac{(N+1)(2N+1)-3(N+1)^2+3N+3N^2}{6N} \\
&\quad=\frac{2N^2-2}{6N}=\frac{N^2-1}{3N}
\end{aligned}$$

For large N, this expression approaches $\frac{N}{3}$ which is $\frac{1}{3}$ of the way across the disk.

Exercise 13.2.4

The inner track is shorter than the outer track by $\frac{1.75}{0.75}=\frac{7}{3}$. Thus, the probability density function for the random sector position could be thought of as a trapezoid with altitude 1 and bases b and $\frac{3}{7}b$. Since the area of this trapezoid must be 1, b is $\frac{7}{5}$, and the probability density function is $p(x)=\frac{3}{5}+\frac{4}{5}x$, with x ranging from 0 to 1 (0 if the sector is on the innermost cylinder and 1 if sector is on the outermost cylinder). We need to find the average distance between two randomly chosen sectors, which is: $\int_0^1\int_0^1 p(x)p(y)\,|x-y|\;dx\;dy$. Since the direction of the moves is irrelevant, we can omit the absolute value by doubling and restricting to the case where $x > y$. Thus, we get: $2\int_0^1 \int_0^x \left(\frac{3}{5} + \frac{4x}{5}\right)\left(\frac{3}{5} + \frac{4y}{5}\right)(x - y)\ dy\ dx$. Integrating, we get: $\frac{121}{375}$, which is slightly less than one third of the tracks.

Exercise 13.2.5

Since the disk controller postpones all other requests until the block is ready to be written back to the disk, we can safely assume that the head will not move between the time we read the block and the time we are ready to write the block back. Because it takes less time to modify the block in memory than it does for the disk to rotate, we can have two cases:

Case1 is when the block is small enough so that the modification takes less time than it does for the disk to complete the current rotation (i.e. the same rotation that was reading the block). In this case the only delay between start of the read and start of the write is 1 disk rotation.

Case2 is when block is large (e.g. occupies most of the track). In this case even though the modification takes less time than the disk rotation, it takes more time than it does for the disk to complete the current rotation. In this case, the only delay between the start of read and start of write is 2 rotations.

Taking the worst case (case 2), the time to modify a block is then equals to: seek time + rotational latency + 2 rotations + transfer time (note that we do not explicitly add the transfer time for the read since the cost of 2 rotations includes the time it will take to read the block). Thus, on average, using the numbers from Example 13.2, it would take $6.46 + 4.17 + 2 \cdot 8.33 + 0.13$ or 27.42 ms.

Section 13.3

Exercise 13.3.1

(a)

Cylinder of Request	First time available
8000	$0 + 7 + 4.3 = 11.3$
4000	$11.3 + 2 + 4.3 = 17.6$
48000	$17.6 + 12 + 4.3 = 33.9$
40000	$33.9 + 3 + 4.3 = 41.2$

(b)

Cylinder of Request	First time available
8000	0 + 7 + 4.3 = 11.3
4000	11.3 + 11 + 4.3 = 26.6
48000	26.6 + 12 + 4.3 = 42.9
40000	42.9 + 10 + 4.3 = 57.2

Exercise 13.3.2

(a) Limiting the movement of the head to only the half of the cylinders improves the average seek time. Which would be $1 + \frac{\frac{65536}{6}}{4000} = 3.73$ ms. Thus, the average rate would be one block per $3.73 + 4.17 + 0.13 = 8.03$ ms. Or about 125 blocks per second. System then could read about 250 blocks per second.

(b) Mirrored Megatron 747 with no restriction has the average rate of one block per 10.76 ms, on each disk, or about 93 blocks per second. System then can read about 186 blocks per second.

(c) Majority of requests may tend to come for the blocks located on the same half of the cylinders, creating a situation where one of the disks will be idle while some requests are waiting for the other disk. The mirrored disk with no restrictions would service such request quicker.

Exercise 13.3.3

(a) Assume there are n requests for the pass. To perform one pass, the head must travel across all tracks (65536), which would take about $n + 16.38$ ms (n starts plus time to travel across all tracks). This is the seek time.
The rotational latency is 4.17 ms, thus the total rotational latency would be $4.17n$ ms.
The transfer time is 0.13 ms, and thus the total transfer time would be $0.13n$ ms.
The total time for the pass would be: $n+16.38+4.17n+0.13n = 5.3n+16.38$ ms.
Since the requests arrive at constant rate A, there would be another n requests ready by the time it takes to complete a pass. Therefore, $5.3n$ +

$16.38 = An$ or $n = \dfrac{16.38}{(A - 5.3)}$. We need to find out the time for the pass, which is An and is $\dfrac{16.38A}{(A - 5.3)}$.

(b) The number of requests serviced in one pass is n from (a). Which is $\dfrac{16.38}{(A - 5.3)}$.

(c) In the best case the request comes in right when the new pass is about to start, so the waiting time would be 0. In the worst case, the request comes in right after the pass has started, so the waiting time would be the time of a full pass. Hence, the average waiting time for a request to get into the pass is half the time it takes for a pass to complete, which is $\dfrac{16.38A}{2(A - 5.3)}$.
Once the request gets into the pass, it must wait to get serviced. In the best case the request is the first one to be processed and the head happens to be positioned at the right cylinder (inner or outer) and the right sector. The wait time to service the request would be 0.
In the worst case, the request is the last one to be processed and is for the last cylinder. In this case the request must wait the time of the full pass minus one transfer time (since the time for a full pass includes the time for this one last request, we only count the seek time and the rotational latency). Therefore, in the best case the request will wait for the service 0 ms, and in the worst case, $2\dfrac{16.38A}{(A - 5.3)} - 0.13$. The average is, $\dfrac{16.38A}{(A - 5.3)} - 0.065$ ms.

Exercise 13.3.4

Since the throughput depends on the number of concurrent requests that the n disks would be able to service, the problem could be reduced to finding the average number of disks that would be servicing requests at one given time. We have n disks, and each requests arrives for any of those disks at random. If the disk is already working on some request, we must stop and wait for it to complete. For instance, there will be only one disk working when the second request came in for the same disk ($p = \frac{1}{n}$). Further, we will have only two disks working if the second request came for any of the $n - 1$ disks and the third request came for any of the 2 working disks ($p = \frac{n-1}{n} \cdot \frac{2}{n}$). Extending this analogy, we get that the average number of working disks is:

$$1 \cdot \frac{1}{n} + 2 \cdot \frac{n-1}{n} \cdot \frac{2}{n} + 3 \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} + ...$$

$$+(n-1)\cdot\frac{n-1}{n}\cdot\frac{n-2}{n}\cdot\ldots\cdot\frac{2}{n}\cdot\frac{n-1}{n}+n\cdot\frac{n-1}{n}\cdot\frac{n-2}{n}\cdot\ldots\cdot\frac{1}{n}$$

Which is

$$\sum_{k=1}^{n}\left(k\cdot\frac{(n-1)!}{n^{k-1}(n-k)!}\cdot\frac{k}{n}\right)=\sum_{k=1}^{n}\frac{k^2(n-1)!}{n^k(n-k)!}$$

For instance, for 4 disks, the average is: $\frac{1}{4}+\frac{3}{4}+\frac{27}{32}+\frac{3}{8}=\frac{71}{32}$

Exercise 13.3.5

Let the cylinder be a unit circle. Then the probability that a random block on the cylinder is at some position less than x between 0 and 1 is x (e.g. the probability that the block is somewhere on the cylinder (that is position < 1) is 1, and the probability that the block is somewhere not passed the middle (that is < 0.5) is 0.5). The probability of k blocks all being at some distance less than x is, x^k. The probability density function is then the derivative kx^{k-1}, and the average of x is $\int_0^1 xkx^{k-1}$, or $\frac{k}{k+1}$.

Section 13.4

Exercise 13.4.1

(a) 1

(b) 0

(c) 1

Exercise 13.4.2

(a) 10

(b) 00

(c) 10

Exercise 13.4.3

8 hours is $\frac{1}{1095}$ of the year. The probability that mirror disk will fail during the copying is then $\frac{1}{1095} \cdot 0.04$, or one in 27375.

Exercise 13.4.4

H hours is $\frac{H}{8760}$ of the year.

(a) The probability of the data loss is a probability of one of the two disks failing $(F + F)$, and the other disk failing while the first disk is repaired $F\frac{H}{8760}$. So the probability of a data loss is $\frac{2F^2H}{8760}$, or $\frac{F^2H}{4380}$.

(b) In order for the data loss to occur, two additional disks must fail within the H hours of a first disk failure. The probability of first disk failure is NF. The probability of the disk failure within the H hours is $\frac{FH}{8760}$, and hence, the probability of the two disk failures within H hours is $\left(\frac{FH}{8760}\right)^2$. There are $\frac{(N-1)!}{2!(N-1-2)!} = \frac{(N-1)(N-2)}{2}$ possible pairs of disks. Thus, the probability of the data loss is $FN\frac{(N-1)(N-2)}{2}\left(\frac{FH}{8760}\right)^2$.

Exercise 13.4.5

Loss of data will occur when all three disks fail within H hours. Probability of the first disk failure is $3F$. The probability of the rest two disk failing within H hours is $\left(\frac{FH}{8760}\right)^2$. The probability of all three disks failing within H hours is then $\frac{3F^3H^2}{8760^2}$, and the mean time to data failure is $\frac{8760^2}{3F^3H^2}$.

Exercise 13.4.6

(a) 01010110

(b) 00110110

Exercise 13.4.7

(a) 01010110

(b) 00110110

Exercise 13.4.8

Only changes to the redundant disk needed to be made. The corresponding block of the redundant disk should be changed to

(a) 10101010

(b) 11001010

Exercise 13.4.9

(a) The blocks for the disks 5, 6, and 7 are: 10101111, 01111111, and 11101101 respectively.

(b) Disks 5 and 7 need to change to 01111011 and 00111000, respectively.

Exercise 13.4.10

Use the matrix in figure 13.10.

(a) Columns 1 and 7 differ in rows 1 and 2. Either row 1 or 2 can be used to restore disk 1. We pick row 1, and so disk 1 is restored by taking modulo-2 sum of disks 2, 3, and 5. Next, we use row 3 (where disk 7 has a 1) to restore disk 7 by taking a modulo-2 sum of disks 1, 3, and 4.

(b) Use row 1 to recover disk 1 by taking the modulo-2 sum of disks 2, 3, and 5. Then use row 2 to restore disk 4 by taking a modulo-2 sum of disks 1, 2, and 6.

(c) Use row 1 to restore disk 3 by taking a modulo-2 sum of disks 1, 2, and 5. Use row 2 to recover disk 6 by taking a modulo-2 sum of disks 1, 2, and 4.

Section 13.5

Exercise 13.5.1

(a) 15 (for CHAR) + 2 (for INTEGER) + 10 (for DATE) + 8 (for TIME) = 35 bytes.

(b) 16 (for CHAR) + 4 (for INTEGER) + 12 (for DATE) + 8 (for TIME) = 40 bytes.

(c) 16 (for CHAR) + 8 (for INTEGER) + 16 (for DATE) + 8 (for TIME) = 48 bytes.

Exercise 13.5.2

(a) 8 (for REAL) + 17 (for CHAR) + 1 (for BYTE) + 10 (for DATE) = 36 bytes.

(b) 8 (for REAL) + 20 (for CHAR) + 4 (for BYTE) + 12 (for DATE) = 44 bytes.

(c) 8 (for REAL) + 24 (for CHAR) + 8 (for BYTE) + 16 (for DATE) = 56 bytes.

Exercise 13.5.3

(a) 4 (for pointer) + 4 (for pointer) + 1 (for CHAR) + 35 (for the data, from 13.5.1 (a)) = 44 bytes.

(b) 4 (for pointer) + 4 (for pointer) + 4 (for CHAR) + 40 (for the data, from 13.5.1 (b)) = 52 bytes.

(c) 8 (for pointer) + 8 (for pointer) + 8 (for CHAR) + 48 (for the data, from 13.5.1 (c)) = 72 bytes.

Exercise 13.5.4

(a) 8 (for pointer) + 10·2 (for integers) + 36 (for the data, from 13.5.2 (a)) = 64 bytes.

(b) 8 (for pointer) + 10·4 (for integers) + 44 (for the data, from 13.5.2 (b)) = 92 bytes.

(c) 8 (for pointer) + 10·8 (for integers) + 56 (for the data, from 13.5.2 (c)) = 144 bytes.

Section 13.6

Exercise 13.6.1

Megatron 747 disk has 2^{16} tracks per surface and, thus, we need 2 bytes to address each cylinder. There are 16 surfaces, so we need 1 byte to address tracks within a cylinder. Example 13.1 and others use 64 blocks per tracks as an average. If we follow that convention, we will need 1 byte to address blocks in a track, and therefore, will need a total of 4 bytes.

Exercise 13.6.2

We need 3 bytes to address 100000 cylinders, 1 byte to address the 10 tracks per cylinder, and assuming that block size is at least 4K (4 sectors), we would need 1 byte to address blocks in a track. Thus, we would need a total of 5 bytes.

Exercise 13.6.3

(a) We would need the same size of 4 bytes for addressing the block. In addition, we would need to address any byte within the block. If we assume, as in several examples throughout Chapter 13, that the block size is 2^{14} bytes, we would need a total of 6 bytes for the address.

(b) We would need 4 bytes to address the block, plus 4 byte key for the total of 8 bytes of the address.

Exercise 13.6.4

4 + 2 (for device) + 4 (for block address, from exercise 13.6.2 result) = 10 bytes.

Exercise 13.6.5

16 + 4 (for block address) + 2 (for byte address) = 22 bytes.

Exercise 13.6.6

Each entry of the table occupies 4 bytes for physical address (see exercise 13.6.1), plus k bytes for a logical address. Logical address has to be long enough to address all blocks on the disk. In exercise 13.6.1 we assumed that there are about 64 blocks per track. Since there are $2^{16} \cdot 16 = 2^{20}$ tracks on the disk, there are $2^{20} \cdot 64 = 2^{26}$ blocks on the disk, and so we would need 4 bytes for the address. Thus, the entry size is 8 bytes, and there are could be up to 2^{26} entries (one entry for each possible block). Therefore, the maximum size of the table is $8 \cdot 2^{26} = 2^{29}$ bytes, and would occupy $\frac{2^{29}}{4096} = 2^{17}$ 4096-byte blocks (or 512M bytes).

Exercise 13.6.7

On the first day no deletes occur, so $2(100+2) = 204$ bytes are added to the block. The block has now $4096 - 204 = 3892$ bytes. Each subsequent day we delete a record but leave the tombstone in the offset table, so only gain 100 bytes back. Adding tow inserts, we use a total of $2(100 + 2) - 100 = 104$ bytes. Thus, it will take $\frac{3892}{104} = 37$ days to fill up the block. So, after 38 days there will be no room to insert the two records (i.e. we will fail on the 39^{th} day).

Exercise 13.6.8

Suppose it takes time t to swizzle a pointer. Further suppose we have n pointers that need to be swizzled. If we use automatic swizzling, it will take time $\frac{tn}{2}$. And if we swizzle pointers on-demand, it will take tine npt. Thus, it is more efficient to swizzle automatically for $p > 0.5$.

Exercise 13.6.9

If we use no-swizzling, the total performance cost is $np(k \cdot 10) = 10npk$. If we use automatic swizzling, the cost is $20n + np(k \cdot 1) = 20n + npk$. If we use on-demand swizzling, the cost is $np(((k - 1) \cdot 1) + 30) = np(k + 29)$. Elimination the common

term n from all three formulas, we get:
$10pk$ vs. $20 + pk$ vs. $p(k + 29)$.
This shows that no-swizzling is best out of the three approaches only when the number of pointers k is less than 3, the automatic swizzling is better when k is greater than 2 and the probability of the pointer to be followed is greater than $\frac{20}{29}$ (or 68.97%), and the on-demand swizzling is best when k is greater than 3 and the probability of the pointer to to be followed is less than $\frac{20}{29}$ (or 68.97%).

Section 13.7

Exercise 13.7.1

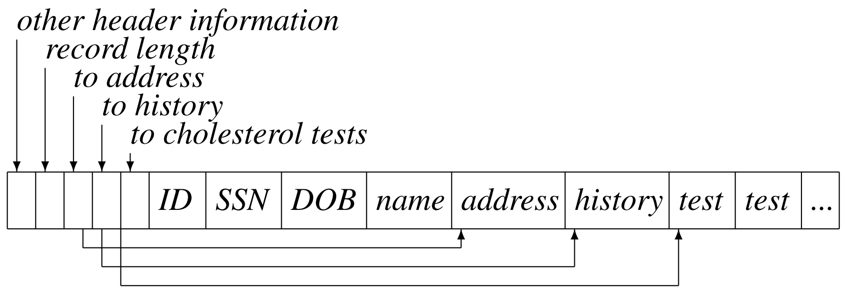
We get 10+10+10 = 30 for the fixed-length fields, 4+4 = 8 for the pointers to varying-length Fields, and 4 for the record length. The total is 42 bytes.

Exercise 13.7.2

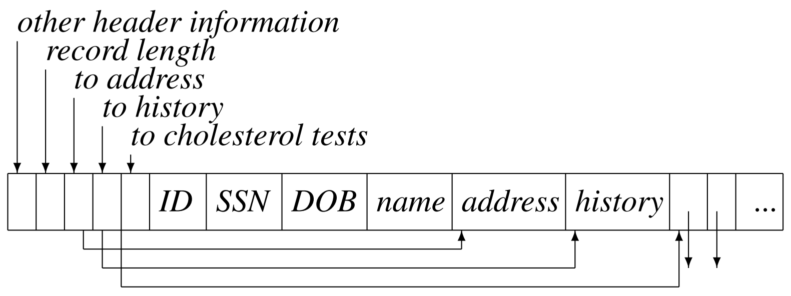
We get 42 (from 13.7.1) + 30 (average for the name) + 50 (average for the address) + 500 (average for the history) = 622 bytes.

Exercise 13.7.3

(a)



(b)



Exercise 13.7.4

This question is open to interpretation. There could be a large number of different tests and each test could be taken by a patient many times. Therefore, the number of test result fields can be unbounded, and it is not clear how to abtain the answers based on the given data.

(a) Assume there are n possible test result values. The expected number of test result fields is then np and so the average number of bytes used in the record to store test results is 4 (for the pointer) + $40np$.

(b) Similar to (a) but we need an additional 4 bytes per test result for the pointer, so the number of bytes for the test results is $4 + 44np$.

(c) If we pre-allocate space in the record for k, we will have to use 4 bytes for a pointer plus $40k$ bytes, plus 4 bytes to a pointer to another block (or chain of blocks). This does not seem to provide us much advantage in terms of saving the storage. If we need less fields than k, we will waste storage. And if we need more fields than k, then it would not matter (as far as storage size) whether we had k fields in the record and x fields outside the record or we had all $k + x$ fields outside the record.

(d) In this case, unlike (c), we do get some benefit from avoiding storing fields outside the record. However, once we have at least one field stored outside the record, there is no furhter penalty for as many other fields to be stored outside the record. The best value for k seems to be around the expected number of fields: $k = np$.

Exercise 13.7.5

If we span records with large size, the best we could do is to have two fragments per block. Thus, the biggest r is $1000 - 2 \cdot 16 - 1 = 967$.

Exercise 13.7.6

Assuming worst case, we need 17.38+8.33 = 25.71 ms to position to the right block to read. With 100 ms delay, this leaves 74.29 ms to read the data (to buffer it in main memory). Thus, we can read a full 8 tracks of the cylinder and still have 7.65 ms left to read about 235 sectors. To simplify, we can assume we will read only 8 full tracks. Therefore we could read about 8388608 bytes. Technically, because there are 16 cylinders, we could read the next 8 tracks in 66.64 ms and so we would alternate between 67.64025 ms and 66.64 ms (assuming that to read next 8 tracks we only need to travel one cylinder). Thus, if we make the playing time for the in-memory buffer about 67.140125 ms, we should be able to stream the data without further delays. Since movie playing rate is 298.262 bytes per ms, we can play about $\frac{8388608}{298.262 \cdot 67.140125} = 418$ movies. Therefore, if we organize movies in chunks so that 418 chunks from differnt movies fit into 8 tracks and all movies occupy consecutive cylinders, we can play as many as 418 movies with an initial delay of 100 ms.

Section 13.8

Exercise 13.8.8

1. Performance

    - No need to calculate the size of each tuple

2. Better disk space (and main memory) utilization

    - Often, varying-length tuples require allocation of storage big enough to handle the maximum possible tuples size. However, many tuples may be much smaller, and so storage could be wasted

3. Simpler processing (algorithms)

- Usually, fixed-length tuples require less amount of processing since certain assumptions could be made based on the constant tuple length (e.g. start of next tuple, number of tuples in a block, etc.)