You will submit your solution to this assignment to the Curator System (as `HW03`). Your solution must be either a plain text file (e.g., NotePad) or a typed MS Word document; submissions in other formats will not be graded.

Except as noted, credit will only be given if you show relevant work.

---

1. [7 points each] Assume that we have a hash table of 10 slots, and we use the hash function h(k) = k mod 10. Show the results of inserting the following series of records in the table using the various collision resolution methods: 11, 31, 32, 56, 46, 47, 66.

(a) Linear probing with step size 1

0	1	2	3	4	5	6	7	8	9
	11	31	32			56	46	47	66

11 goes to its home slot, 1;
31 collides with 11 and probes to slot 2;
32 collides with 31 and probes to slot 3;
56 goes to its home slot, 6;
46 collides with 56 and probes to slot 7;
47 collides with 46 and probes to slot 8;
66 collides with 56 and then probes past slots 7 and 8, going into slot 9

(b) Linear probing with step size 3

0	1	2	3	4	5	6	7	8	9
	11	32		31	66	56	47		46

11 goes to its home slot, 1;
31 collides with 11 and probes to slot 4;
32 goes to its home slot;
56 goes to its home slot, 6;
46 collides with 56 and probes to slot 9;
47 goes to its home slot, 7;
66 collides with 56 and probes past slots 9 and 2, going into slot 5

(c) Quadratic probing with k^2

0	1	2	3	4	5	6	7	8	9
66	11	31	32			56	46	47	

11 goes to its home slot, 1;
31 collides with 11 and probes to slot 2;
32 collides with 31 and probes to slot 3;
56 goes to its home slot, 6;
46 collides with 56 and probes to slot 7;
47 collides with 46 and probes to slot 8;
66 collides with 56 and probes past slot 7, going into slot 0

(d) Double hashing with second hash function h2(k) = k/10.

0	1	2	3	4	5	6	7	8	9
46	11	32		31		56	47	66	

11 goes to its home slot, 1;
31 collides with 11, h2(31) = 3 so 31 probes to slot 4;
32 goes to its home slot, 2;
56 goes to its home slot, 6;
46 collides with 56, h2(46) = 4 so 46 probes to slot 0;
47 goes to its home slot 7;
66 collides with 56, h2(66) = 6 so 66 probes to slot 2 and then to slot 8

2. [8 points each] The designer of a hash table decides to use some sort of probing strategy to resolve collisions. The exact probing strategy is not important to this question. The designer also decides to never allow the table to become completely full, by resizing/rehashing before that happens.

The designer decides to avoid the use of tombstones when deletions are performed. Instead, he proposes to use only two slot states, empty and full, and apply the following approach when a deletion is performed:

> Suppose that the element to be deleted is in slot L; use the probing strategy, taking one (more) probe step from slot L, to see if there is an element that was pushed beyond slot L (i.e., the one due to a collision with the now-deleted element in slot L). If so, move that element into slot L and delete it from its original slot, applying this same strategy from that position in the table. Stop when probing reaches a slot that is empty or that contains and element that did not collide with the element being deleted.

The designer argues that this will not only simplify the design of the table, but it will make searching more efficient, in some cases, since it will sometimes move elements "closer" to their home slot.

a) Explain how the designer can determine whether the element that is "one more probe step from slot L" has been "pushed beyond slot L" due to a collision with the element that was formerly in slot L.

The problem statement was incorrect and there was really no correct answer to the questions as posed.

So, we're giving full credit for any answer.

b) Explain why the strategy will not yield a working hash table, at least in some situations. Efficiency is not the issue here; logical correctness is.

3. [7 points each] Assume a system uses a hard drive with the following physical characteristics:

total capacity	64 GB
# of platters	4
# of tracks per surface	16384
# of sectors per track	2048
cluster size	4 KB
spindle speed	14400 RPM
head start time	0.1 ms
track to track seek time	0.01 ms

In answering the following questions, express all final time values to the nearest thousandth of a millisecond (e.g, 8.333 ms). For full credit, you must show and explain all of your calculations.

a) What is the average random head seek time for this drive?

From the notes, the average random seek time is the sum of the head start time and the product of the track-to-track seek time and 1/3 the number of tracks per surface:

Avg seek time = 0.1 + (0.01 * 16384)/3
= 0.1 + 54.613
= 54.713ms

b) What is the average rotational latency for this drive?

From the notes, the average rotational latency is the time for half a rotation of the platters:

Avg latency = 0.5 * (60000/14400)
= 2.083ms

c) What is the average total time required to read one randomly-chosen sector from this drive?

The average read time for a sector would be the average seek time plus the average latency plus the time required for a sector to rotate past the read head. The latter time would be:

Transfer time = (1/2048) * 4.167
= 0.002ms

So the average total time to read a sector would be:

Avg sector read time = 54.713 + 2.083 + .002
= 56.798ms

d) What is the average total time required to read one randomly-chosen cluster from this drive?

A cluster is 4KB or 8 sectors (since a sector is 512 bytes). So, the transfer time for a cluster would be:

Transfer time = (8/2048) * 4.167
= 0.016ms

So the average total time to read a sector would be:

Avg sector read time = 54.713 + 2.083 + .016
= 56.812ms

e) What is the average total time required to read a file of 20 MB from this drive if the clusters are stored as efficiently as possible on the drive?

A single track stores 1MB of data, so a 20MB file would occupy 20 tracks. The total time to seek to and read the first track would be:

Time for first track = 54.713 + 2.083 + .4.167
= 60.963ms

The time to read each subsequent track would be:

Time for first track = 0.1 + 0.01 + 2.083 + .4.167
= 6.360ms

So, the total time to read the file would be:

Total time = 60.963 + 19 * 6.36
= 181.803ms

(Actually, the file could be stored slightly more efficiently by storing it on 8 tracks in one cylinder, 8 tracks on an adjacent cylinder, and 4 tracks on one more adjacent cylinder.)

f) What is the average total time required to read a file of 20 MB from this drive if the clusters are randomly scattered on the drive?

If the file is randomly scattered on the drive, then we must read each cluster as a separate operation. A 20MB file would occupy 5120 clusters (since 1MB takes 256 4KB clusters). So the total read time would be:

Total time = 5120 * 56.812
= 290877.440ms

4. [7 points each] Assume each of the changes described below were made to the design of the drive described in the preceding question, and that no other physical changes were made. Indicate the effect of the change on the average random head seek time and the average rotational latency time for the drive. The parts are independent. Justify your conclusions if you want credit.

a) doubling the rotational speed, and modifying the read/write heads so that they can keep up with the increased transfer rate

Doubling the rotational speed would cut the average latency in half, but it would have no effect whatsoever on the seek time.

b) reducing the platter diameter by half and doubling the number of tracks per surface; the track-to-track seek time is not changed

Reducing the platter diameter and doubling the number of tracks per surface would increase the average seek time, since that depends only on the number of tracks per surface and the track-to-track seek time, which has not been changed. This would not have any effect whatsoever on the latency.