

4-1 Hash functions & load

a. I imagine that an algorithm requires us to hash things containing English phrases knowing that strings are stored as sequences of characters. APH decides to simply use the sum of those character values (modulo the size of her hash table) as the string's hash. Will the performance of her implementation match the expected value shown in lecture?

1. Yes, ^{the} sum operation will space out strings nicely by length
2. ... by the characters they use

③ No, reordering words in a string will not produce a different hash
X. No, because the independence condition of the simple uniform hashing assumption is violated

Not 4, hashing does not depend on where other keys are hashed.

6. A implements both collision resolution & dynamic resizing for her hash table. However, she doesn't want to ~~use~~ both. Dynamic resizing can avoid collisions, collisions don't cause correctness issues if collision resolution is implemented. Which statement about these 2 properties is true?

1. DR alone will preserve both properties
2. DR will preserve correctness, not performance
3. Collision resolution will preserve performance, not correctness

④ Both are needed

DR for performance, collision resolution for correctness

c. A implements resizing. ~~Any~~ Enlarging a table of size m into a table of size m' , containing n elements. What is the best time complexity she can achieve?

1. $\Theta(m)$ 2. $\Theta(n)$ ~~3. $\Theta(n)$~~ 4. $\Theta(m')$ 5. $\Theta(m+n)$ 6. $\Theta(m+n)$ 7. $\Theta(m'+n)$

Lecture 09 states resizing has $\Theta(m+n)$ time ~~cost~~ $\Theta(m)$

In this case the size of the final table is m' so plug $m=m'$ into the above formula

d. In lecture, we discussed doubling the size of our hash table. IHC implements ($m'=2m$) but stops when it occurs she might be able to avoid wasting half of the memory the table occupies on empty space by letting $m'=m+k$ instead, where k is some constant. Does this work? If so why don't we do it. Theoretical & practical reasoning required.

Theoretically: From Lec 09 Notes

Q1) amortized cost for rebuilding at $m'=2m$: $\Theta(\sum_{i=1}^{\infty} 2^i) = \Theta(m)$

Otherwise, just adding in values would take $\Theta(n)$ amortized cost to find the appropriate location: $\Theta(m+m+k+\dots+2m) = \Theta(m^2)$

Practically:

It is very simple to double size, and do so at powers of 2 instead of some constant

Think about the asymptotic cost over many resizings.

4-2. v. Let's examine the "membership falling" use case. Which statement accurately describes it?

1. Many insertions right after creation, and then mostly lookups
2. " " " " " " only " "
3. A workload of evenly-mixed insertions/deletions & lookups
4. Alternating rounds of insertions/deletions & lookups

It is described as "create once then rarely changes" \Rightarrow many insertions, mostly at creation

(c) Many cells to hash-key \Rightarrow many lookups.

b. Now imagine you have to place a host function, size, collision resolution strategy, and so forth in order to make a hash table perfectly suited to this use case alone. After the statement that best describes the choices you might make.

1. A large min size on a growth rate of 2.
2. A small min size & growth rate of 2
3. large min size & . 4
4. small . . 4

The document recommends a low $\lambda \Rightarrow$ many slots per key \Rightarrow large table size
 \Rightarrow faster growth rate

Many keys inserted initially \Rightarrow start w/ a bigger table