ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2019 Prof. R. Wattenhofer

Computer Engineering II

Exercise Sheet Chapter 8

We categorize questions into four different categories:

Quiz Short questions which we will solve rather interactively at the start of the exercise sessions.

Basic Improve the basic understanding of the lecture material.

Advanced Test your ability to work with the lecture content. This is the typical style of questions which appear in the exam.

Questions marked with $^{(g)}$ may need some research on Google.

Quiz

1 Quiz

a) What happens if a hash function is biased to favor some buckets?

i) The number of collisions stays the same, it just spreads to the favored buckets.

ii) The number of collisions goes down since more buckets will be empty.

iii) The number of collisions goes up.

b) What do we need to take into account to analyze the time complexity of using a hash table that picks hash functions from a universal family?

i) Number of keys

ii) Distribution of keys

iii) Size of hash table

iv) Similarities between keys

v) Method for resolving collisions

c) Is hashing a good idea if you need every single insert/delete/search to be fast? Consider what the worst-case scenario for e.g. an insert operation can be.

i) Yes

ii) No

Basic

2 Trying out hashing

Let $N = \{10, 22, 31, 4, 15, 28, 17, 88, 59\}$ and $m = 11$. Let $h(k) = k \mod m$; now build three hash tables: one for linear probing with $c = 1$, one for quadratic probing with $c = 1$ and $d = 3$, and one for double hashing with $h'(k) = 1 + (k \mod (m-1))$. Reminder:

- Linear probing: $h_i(k) \equiv h(k) + ci \mod m$
- Quadratic probing: $h_i(k) \equiv h(k) + ci + di^2 \mod m$
- Double hashing: $h_i(k) \equiv h(k) + ih'(k) \mod m$

Note: You can play around with hashing with probing on `https://www.cs.usfca.edu/~galles/visualization/ClosedHash.html`. It allows inserting, deleting, and searching in a hash table with probing for some example parametrized hash functions. Note that the possibility of inserting the same key into the table multiple times – which the implementation on that website allows – may not be the same in all implementations. The Java standard library `HashMap` for example will instead replace the old entry with the new one.

Furthermore, you can just do half the exercise in class and the rest at home since it is somewhat time consuming. Also, don't give up if a probing sequence seems to go on for too long!

3 Using hash tables

Assume you are given two sets of integers, $S = \{s_1, \ldots, s_q\}$ and $T = \{t_1, \ldots, t_r\}$, and you want to check whether $S \subseteq T$.

a) Give an efficient algorithm that uses hash tables.

b) What is the time complexity of your algorithm? Is it preferable to a simple algorithm that sorts the sets and compares them?

Advanced

4 r-independent hashing

Given a family of hash functions $\mathcal{H} \subseteq \{U \to M\}$, we say that $\mathcal{H}$ is *r-independent* if for every r distinct keys $\langle x_1, \ldots, x_r \rangle$ and h sampled uniformly from $\mathcal{H}$, the vector $\langle h(x_1), \ldots, h(x_r) \rangle$ is equally likely to be any element of M^r.

a) Show that if $\mathcal{H}$ is 2-independent, then it is universal. Hint: use that $\mathcal{H}$ is universal if and only if $\Pr[h(k) = h(l)] \leq \frac{1}{m}$ for keys $k \neq l$.

b) Show that the universal family $\mathcal{H}$ defined in the script (Theorem 8.10) is not 2-independent.

5 Not quite universal hashing

Remember the universal family from the script: $\mathcal{H} := \{h_a : a \in [m]^{r+1}\}$ where $h_a(k_0, \ldots, k_r) = \sum_{i=0}^{r} a_i \cdot k_i \mod m$ for some prime m. Show that if we restrict the a_i to be nonzero, then $\mathcal{H}$ is no longer a universal family if $r \geq 1$ and $m \geq 3$.

Hint: Find two keys with a collision probability of more than $\frac{1}{m}$!

6 Obfuscated quadratic probing

Consider Algorithm 1 with $m = 2^p$ for some integer p.

Algorithm 1 Obfuscated quadratic probing: search

Input: key k to search for

```
1: i := h(k)
2: if M[i] = k then
3:    return M[i]
4: end if
5: j := 0
6: for l ∈ {0, ..., m − 1} do
7:    j := j + 1
8:    i := (i + j) mod m
9:    if M[i] = k then
10:      return M[i]
11:   end if
12: end for
13: return ⊥
```

a) Show that this is an instance of quadratic probing by giving the constants c and d for a hash function $h_i(k) \equiv h(k) + ci + di^2 \mod m$.

b) Prove that the probing sequence of every key covers the whole table. Do this in two steps:

- Show that $h_s(k) \equiv h_r(k) \mod m$ for $r < s$ if and only if $(s - r)(s + r + 1) = t2^{p+1}$ for some integer t.
- Show that only one of $(s - r)$ and $(s + r + 1)$ can be even, then show that $(s - r)(s + r + 1) = t2^{p+1}$ has no solutions if $r < s$ and $r, s < m$.

7 Robin Hood hashing

In hashing with probing we notice that some objects can be placed in buckets early in their probing sequence while others may travel longer in their probing sequence until they find an empty bucket. Robin Hood hashing is trying to rectify this unfair situation by taking from the "short" and giving to the "long". Specifically, we make the following modification in the insertion process of hashing with linear probing: If the bucket is already occupied, then the object that has traveled longer in its probing sequence remains in the bucket.

a) How does this change affect the expected value of the probe sequence length?

b) How does this change the expected value of the longest probe sequence length?

c) How does this change affect the variance of the probe sequence length?

d) Describe the performance issues in searching if we follow the naive delete operation (we find the object and delete it from the hash table leaving the bucket empty). Suggest a modification to address the problem.
Note: This is a general problem of linear probing and independent of the Robin Hood modification, so it might help playing around with the delete operation `https://www.cs.usfca.edu/~galles/visualization/ClosedHash.html`!

e) After multiple delete operations (imagine an almost empty hash table), we notice that the hash table is performing poorly in searching. The Sheriff of Nottingham tries to reverse the effects of Robin Hood (hashing) and comes up with a delete operation that improves the search time significantly. Can you explain the problem to the people of the Nottinghamshire and describe the solution of the Sheriff?
Hint: Think about shifting some objects.