CH-231-A

Algorithms and Data Structures

ADS

Lecture 28

Dr. Kinga Lipskoch

Spring 2020

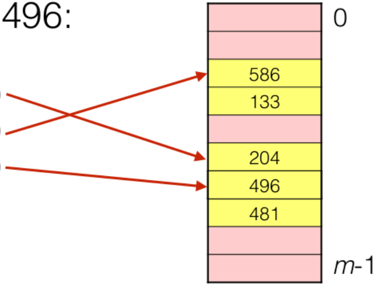
Resolving Collisions by Open Addressing

- No additional storage is used.
- Only store one element per slot.
- Insertion probes the table systematically until an empty slot is found.
- The hash function depends on the key and the probe number, i.e., $h : U \times \{0, 1, ..., m-1\} \rightarrow \{0, 1, ..., m-1\}$.
- The probe sequence $< h(k,0), h(k,1), ..., h(k,m-1) >$ should be a permutation of $\{0, 1, ..., m-1\}$.

Insert Example

- Insert key k = 496:

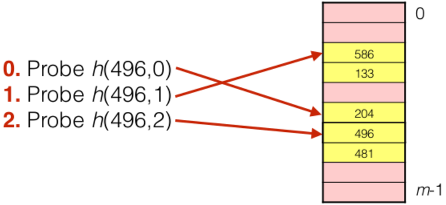
0. Probe $h(496,0)$
1. Probe $h(496,1)$
2. Probe $h(496,2)$




```
HASH-INSERT(T, k)
i = 0
repeat
    j = h(k, i)
    if T[j] == NIL
        T[j] = k
        return j
    else i = i + 1
until i == m
error "hash table overflow"
```

Search Example

```
HASH-SEARCH(T, k)
i = 0
repeat
    j = h(k, i)
    if T[j] == k
        return j
    i = i + 1
until T[j] == NIL or i == m
return NIL
```



- Search key $k = 496$
  - Search uses the same probe sequence, terminating successfully if it finds the key and unsuccessfully if it encounters an empty slot (or made it all the way through the list)
  - Search times no longer depend on load factor α
- What about delete?
  - Have a special node type: DELETED
  - Chaining more commonly used when keys must also be deleted

Probing Strategies (1)

Linear probing:

- Given an ordinary hash function $h'(k)$, linear probing uses the hash function $h(k, i) = (h'(k) + i) \bmod m$.
- This is a simple computation.
- However, it may suffer from primary clustering, where long runs of occupied slots build up and tend to get longer.
  - empty slot preceded by i full slots gets filled next with probability $(i + 1)/m$

Probing Strategies (2)

Quadratic probing:

- Quadratic probing uses the hash function $h(k, i) = (h'(k) + c_1 \cdot i + c_2 \cdot i^2) \bmod m$.
- Offset by amount that depends on quadratic manner, works much better than linear probing
- But, it may still suffer from secondary clustering: If two keys have initially the same value, then they also have the same probe sequence
- In addition c_1, c_2, and m need to be constrained to make full use of the hash table

Probing Strategies (3)

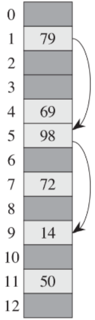
Double hashing:

- Given two ordinary hash functions $h_1(k)$ and $h_2(k)$, double hashing uses the hash function $h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod m$.
- The initial probe goes to position $T[h_1(k)]$; successive probe positions are offset by $h_2(k)$ $\rightarrow$ the initial probe position, the offset, or both, may vary
- This method generates excellent results, if $h_2(k)$ is relatively prime to the hash-table size m,

Probing Strategies (4)

Double hashing (continue):

- e.g., by making m a power of 2 and design $h_2(k)$ to only produce odd numbers.
- or let m be prime and design h_2 such that it always returns a positive integer less than m, e.g. let m' be slightly less than m:
  $h_1(k) = k \bmod m$
  $h_2(k) = 1 + (k \bmod m')$



$h_1(k)$ = k mod 13
$h_2(k)$ = 1 + (k mod 11)

—> k=14; $h_1(k)$=1, $h_2(k)$=4

—> k=27; $h_1(k)$=1, $h_2(k)$=6

Analysis of Open Addressing (1)

Theorem:

- Assume uniform hashing, i.e., each key is likely to have any one of the $m!$ permutations as its probe sequence.
- Given an open-addressed hash table with load factor $\alpha = n/m < 1$.
- The expected number of probes in an unsuccessful search is, at most $\dfrac{1}{1-\alpha}$.

Analysis of Open Addressing (2)

Proof:

- At least, one probe is always necessary.
- With probability n/m, the first probe hits an occupied slot, i.e., a second probe is necessary.
- With probability $(n-1)/(m-1)$, the second probe hits an occupied slot, i.e., a third probe is necessary.
- With probability $(n-2)/(m-2)$, the third probe hits an occupied slot, i.e., a fourth probe is necessary.
- ...

Analysis of Open Addressing (3)

Given that $\dfrac{n-i}{m-i} < \dfrac{n}{m} = \alpha$ for $i = 1, 2, \ldots, n$.

$$
\begin{aligned}
&1+\frac{n}{m}\left(1+\frac{n-1}{m-1}\left(1+\frac{n-2}{m-2}\left(\cdots\left(1+\frac{1}{m-n+1}\right)\cdots\right)\right)\right)\\
&\le 1+\alpha(1+\alpha(1+\alpha(\cdots(1+\alpha)\cdots)))\\
&\le 1+\alpha+\alpha^2+\alpha^3+\cdots\\
&= \sum_{i=0}^{\infty}\alpha^i\\
&= \frac{1}{1-\alpha}.
\end{aligned}
$$

Analysis of Open Addressing (4)

- The successful search takes less number of probes $\left[\text{expected number is at most } \dfrac{1}{\alpha} \ln \dfrac{1}{1-\alpha}\right]$.
- We conclude that if α is constant, then accessing an open-addressed hash table takes constant time.
- For example, if the table is half full, the expected number of probes is $1/(1-0.5) = 2$.
- Or, if the table is 90% full, the expected number of probes is $1/(1-0.9) = 10$.

Summary

- Dynamic sets with queries and modifying operations.
- Array: Random access, search in $O(\lg n)$, but modifying operations $O(n)$.
- Stack: LIFO only. Operations in $O(1)$.
- Queue: FIFO only. Operations in $O(1)$.
- Linked list: Modifying operations in $O(1)$, but search $O(n)$.
- BST: All operations in $O(h)$.
- Red-black trees: All operations in $O(\lg n)$.
- Heap: All operations in $O(\lg n)$.
- Hash tables: Operations in $O(1)$, but additional storage space.