CSCI 241

Scott Wehrwein

Hash Tables: Open Addressing

Goals

Know how to use open addressing with linear or quadratic probing for collision resolution.

Load Factor: Performance Implications

$$\text{Load factor } \lambda = \frac{\text{\# entries in table}}{\text{size of the array}}$$

If λ is large, runtime is slow.

If λ is small, memory is wasted.

If the memory's sitting there wasted... why not use it?

Open Addressing with Linear Probing

- Open Addressing - use empty buckets to store things that belong in other buckets.

- Which empty bucket? Using the next empty one is called Linear Probing

```
put(1, “dog”);
put(11, “auk”);
put(10, “bear”);
put(14, “cat”);
put(24, “ape”);
```

0	
1	
2	
3	
4	

```
put(key):
    h = hash(key);
    while A[h] is full:
        h = (h+1) % N
    A[h] = value
```

Open Addressing with Linear Probing

- Open Addressing - use empty buckets to store things that belong in other buckets.

- Which empty bucket? Using the next empty one is called Linear Probing

```
put(1, “dog”);
put(11, “auk”);
put(10, “bear”);
put(14, “cat”);
put(24, “ape”);
```

0	
1	(1, dog)
2	
3	
4	

```
put(key):
    h = hash(key);
    while A[h] is full:
        h = (h+1) % N
    A[h] = value
```

Open Addressing with Linear Probing

- Open Addressing - use empty buckets to store things that belong in other buckets.

- Which empty bucket? Using the next empty one is called Linear Probing

```
put(1, “dog”);
put(11, “auk”);
put(10, “bear”);
put(14, “cat”);
put(24, “ape”);
```

0	
1	(1, dog)
2	(11, auk)
3	
4	

```
put(key):
    h = hash(key);
    while A[h] is full:
        h = (h+1) % N
    A[h] = value
```

Open Addressing with Linear Probing

- Open Addressing - use empty buckets to store things that belong in other buckets.

- Which empty bucket? Using the next empty one is called Linear Probing

```
put(1, “dog”);
put(11, “auk”);
put(10, “bear”);
put(14, “cat”);
put(24, “ape”);
```

0	(10, bear)
1	(1, dog)
2	(11, auk)
3	
4	

```
put(key):
    h = hash(key);
    while A[h] is full:
        h = (h+1) % N
    A[h] = value
```

Open Addressing with Linear Probing

- Open Addressing - use empty buckets to store things that belong in other buckets.

- Which empty bucket? Using the next empty one is called Linear Probing

```
put(1, “dog”);
put(11, “auk”);
put(10, “bear”);
put(14, “cat”);
put(24, “ape”);
```

0	(10, bear)
1	(1, dog)
2	(11, auk)
3	
4	(14, cat)

```
put(key):
    h = hash(key);
    while A[h] is full:
        h = (h+1) % N
    A[h] = value
```

Open Addressing with Linear Probing

- Open Addressing - use empty buckets to store things that belong in other buckets.

- Which empty bucket? Using the next empty one is called Linear Probing

```
put(1, “dog”);
put(11, “auk”);
put(10, “bear”);
put(14, “cat”);
put(24, “ape”);
```

0	(10, bear)
1	(1, dog)
2	(11, auk)
3	(24, ape)
4	(14, cat)

```
put(key):
    h = hash(key);
    while A[h] is full:
        h = (h+1) % N
    A[h] = value
```

Open Addressing with Linear Probing

Problem: (e.g., 1, 1, 3, 2, 3, 4, 6, 4, 5)

Clustered hash values will result in a lot of searching.

```
put(1, “dog”);
put(11, “auk”);
put(10, “bear”);
put(14, “cat”);
put(24, “ape”);
```

0	(10, bear)
1	(1, dog)
2	(11, auk)
3	(24, ape)
4	(14, cat)

```
put(key):
    h = hash(key);
    while A[h] is full:
        h = (h+1) % N
    A[h] = value
```

Open Addressing with Quadratic Probing

Quadratic Probing: Jump further ahead to avoid clustering of full buckets.

Linear probing looks at H, H+1, H+2, H+3, H+4, ...

Quadratic probing looks at H, H+1, H+4, H+9, H+16, ...

```
put(1, “dog”);
put(11, “auk”);
put(10, “bear”);
put(14, “cat”);
put(24, “ape”);
```

0	(10, bear)
1	(1, dog)
2	(11, auk)
3	(24, ape)
4	(14, cat)

```
put(key):
    H = hash(key);
    i = 0;
    while A[h] is full:
        h = (H + i²) % N
        i++;
    A[h] = value
```

Open Addressing: Runtime

- May be faster, but may not be. Depends on keys.
- There's no free lunch: worst-case is always O(n).
- In practice, average-case is O(1) if you make good design decisions and insertions are not done by someone who wants to ruin your day.