NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CX1107 Data Structures and Algorithms **2020/21 Semester 2**

Solution 5: Searching

School of Computer Science and Engineering *Nanyang Technological University*

Q1 A sequence, $x_1, x_2, \ldots, x_n$, is said to be cyclically sorted if the smallest number in the sequence is x_i for some i, and the sequence, $x_i, x_{i+1}, \ldots, x_n, x_1, x_2, \ldots, x_{i-1}$ is sorted in increasing order. Give an algorithm to find the minimal element in the sequence in $\mathcal{O}(\log n)$ time.

S1 Let us see the following examples of cyclically sorted sequence:

1	2	3	4	5	6	7	8
6	7	8	1	2	3	4	5

1	2	3	4	5	6	7	8
3	4	5	6	7	8	1	2

Given sequences above, $x'_1, x'_2, \ldots, x'_{j-1}, x'_j, x'_{j+1}, \ldots, x'_{m-1}, x'_m$, there exist an index, j that

$$x'_j < x'_{j+1} < \ldots < x'_m < x'_1 < \ldots < x'_{j-1}$$

How can we find out the index, j?
Let us select the middle element of the sequence, x'_{mid}. If the $x'_{mid} < x_m$ (the last element of the sequence), then the minimum definitely is in the first half ($x'_1, x'_2, \ldots, x'_{j-1}, x'_j$) (including itself). Otherwise, the minimum will be in the second half ($x'_{j+1}, \ldots, x'_{m-1}, x'_m$).

1	2	3	4	5	6	7	8
3	4	5	6	7	8	1	2

first last

$$mid = \lfloor \tfrac{1+8}{2} \rfloor = 4$$

```
1  int minimum(int n, int m) \\ n and m are the indices of the 1st and last elements
     respectively
2  {
3    int mid;
4    if (m == n) return array[n];  \\ this is the min
5    else {
6     mid=(n+m)/2;
7     if (array[mid] < array[m])       \\ middle < last
8        return minimum(n, mid); \\ find min in 1st half
9     else
10       return minimum(mid + 1, m); \\ find min in 2nd half
11   }
12   }
```

Algorithm of Q1

Q2 The type of a hash table H under closed addressing is an array of list references, and under open addressing is an array of keys. Assume a key requires one "word" of memory and a linked list node

requires two words, one for the key and one for a list reference. Consider each of these load factors for closed addressing: 0.5, 1.0, 2.0.
Estimate the total space requirement, including space for lists, under closed addressing, and then, assuming that the same amount of space is used for an open addressing hash table, what are the corresponding load factors under open addressing?

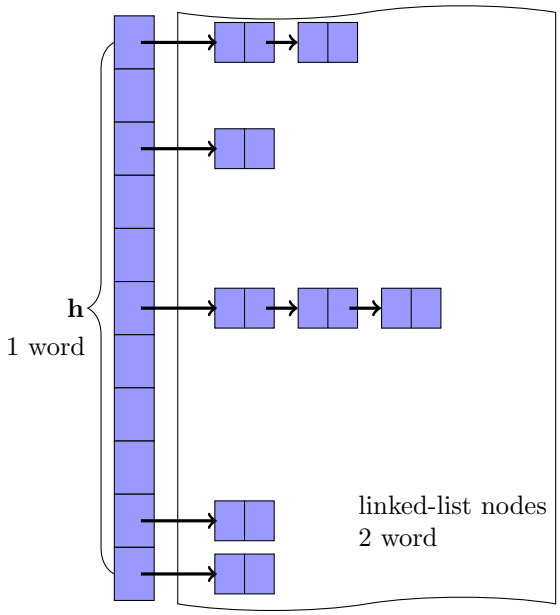

Figure 5.1: Closed Addressing Hash Table

S2 Under closed addressing, let h be hash table size. The space is h words for the list of references.

1. **Load Factor =0.5** implies that there are $0.5 \times h$ list nodes. Each nodes required 2 words. Total space is $2h$
2. **Load Factor =1** implies that there are $1 \times h$ list nodes. Each nodes required 2 words. Total space is $3h$
3. **Load Factor =2** implies that there are $2 \times h$ list nodes. Each nodes required 2 words. Total space is $5h$

The number of keys, n =Load Factor$\times h$.

1. **Load Factor =0.5** implies that there are $0.5 \times h$ keys. Total space is $2h$ for the list. The corresponding load factor under open addressing
$$\frac{0.5h}{2h} = 0.25$$
2. **Load Factor =1** implies that there are $1 \times h$ keys. Total space is $3h$ for the list. The corresponding load factor under open addressing
$$\frac{h}{3h} = \frac{1}{3} = 0.33$$
3. **Load Factor =2** implies that there are $2 \times h$ keys. Total space is $5h$ for the list. The corresponding load factor under open addressing
$$\frac{2h}{5h} = \frac{2}{5} = 0.4$$

Q3 Consider a hash table of size n using open address hashing and linear probing. Suppose that the hash table has a load factor of 0.5, describe with a diagram of the hash table, the best-case and the worst-case scenarios for the key distribution in the table.
For each of the two scenario, compute the average-case time complexity in terms of the number of key comparisons when inserting a new key. You may assume equal probability for the new key to be hashed into each of the n slots.
[**Note: Checking if a slot is empty is not a key comparison.**]

S3

1. **The best scenario:** the $\frac{n}{2}$ keys are hashed and distributed evenly into the n slots and no rehashing. Assuming that equal probability for a key to be hashed into each of the n slots, the average-case time complexity

$$\begin{aligned}\text{average-case time complexity} &= \frac{1}{n}(\sum_{i=1}^{n/2} 1) \\ &= \frac{1}{n}(n/2) \\ &= 0.5 = \Theta(1)\end{aligned}$$

2. **The worst scenario:**the $\frac{n}{2}$ keys are hashed in consecutive slots in the table. Each key always has to rehash and visit every key in the table. The i^{th} key is hashed and rehashed i times to get the slot.

$$\begin{aligned}\text{average-case time complexity} &= \frac{1}{n}(\sum_{i=1}^{n/2} i) \\ &= \frac{1}{n}\frac{n}{4}[1+\frac{n}{2}] \\ &= \frac{n}{8}+\frac{1}{4} = \Theta(n)\end{aligned}$$