SC1007 Data Structures and Algorithms **2022/23 Semester 2**

Solution 5: Hash Table and Graph Representation

School of Computer Science and Engineering *Nanyang Technological University*

Q1 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?

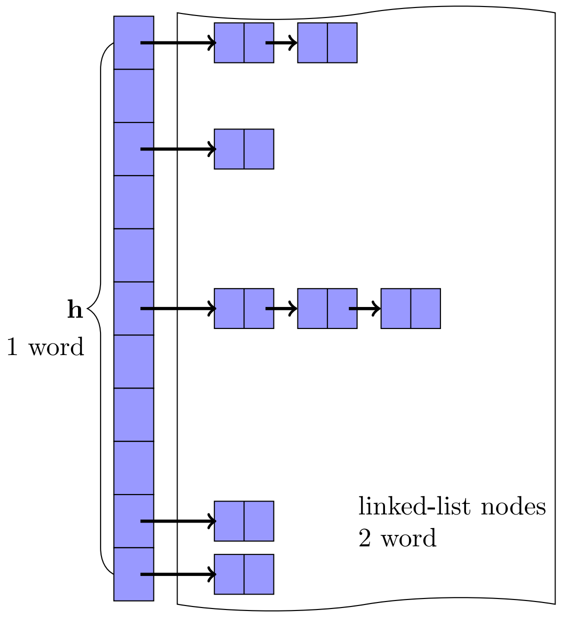

Figure 5.1: Closed Addressing Hash Table

S1 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

$$\frac{2h}{5h} = \frac{2}{5} = 0.4$$

Q2 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.]

S2
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.

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

Q3 Manually execute breadth-first search on the undirected graph in Figure 5.2, starting from vertex s. Then, use it as an example to illustrate the following properties:

(a) The results of breadth-first search may depend on the order in which the neighbours of a given vertex are visited.

(b) With different orders of visiting the neighbours, although the BFS tree may be different, the distance from starting vertex s to each vertex will be the same.

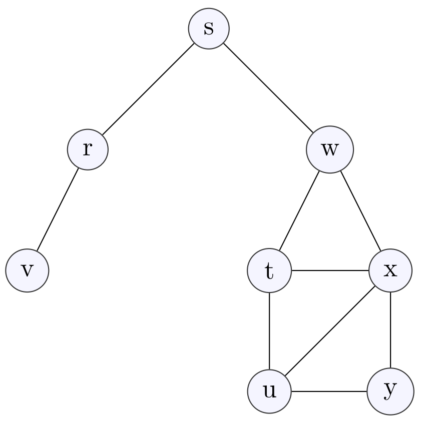

Figure 5.2: The Graph for Q3

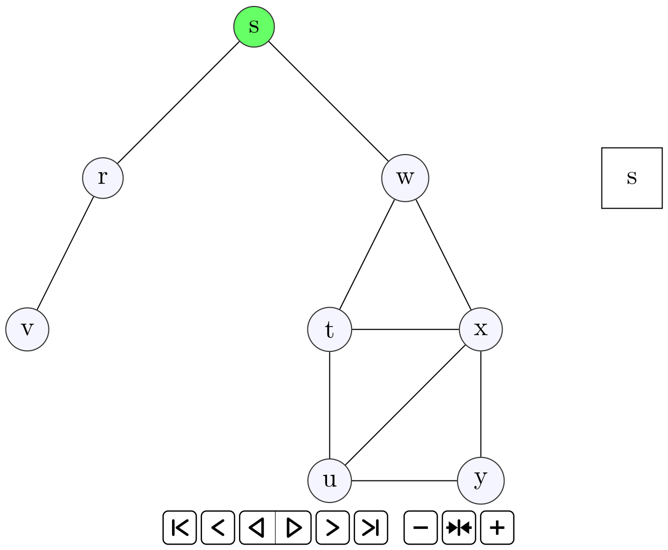

Figure 5.3: Graph for S1

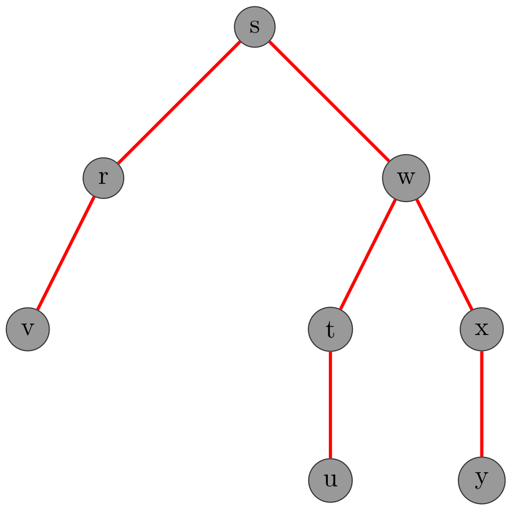

Figure 5.4: Visiting neighbors in alphabetical order

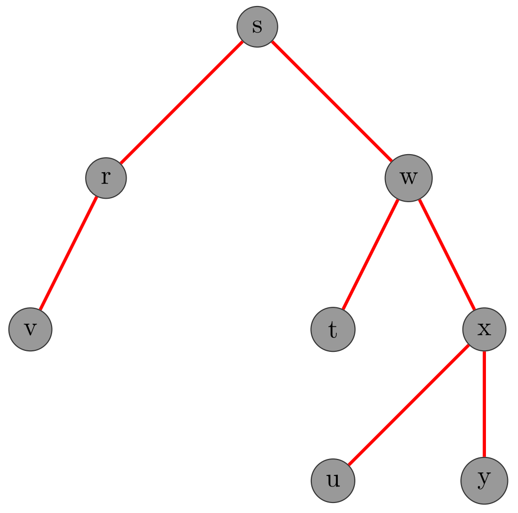

Figure 5.5: Visiting neighbors in reverse alphabetical order

S3

- When the queue is empty, the BFS is finished.
- The edges of BFS tree are shown in red.
- Likewise, a BFS tree can be constructed if the neighbors are visited in the reverse alphabetical order (an exercise for the students).
- The two trees differ in that vertex u is adjacent with t in the left tree, but adjacent with x in the right tree.
- The distance from starting vertex s to each other vertex is equal in the two trees.