

## Question 9.1.1:

Given the sequence  $\langle 3, 10, 2, 4 \rangle$ , use double hashing

Solution

$$\text{Hash}_1(3) = 3 \bmod 5 = 3$$

$$\text{Hash}_1(10) = 10 \bmod 5 = 0$$

$$\text{Hash}_1(2) = 2 \bmod 5 = 2$$

$$\text{Hash}_1(4) = 4 \bmod 5 = 4$$

0	10
1	
2	2
3	3
4	4

Considering that the first hash has the more priority than the second hash

another solution: considering the two hashes have the same priority.

$$\text{Hash}(i) = ((i \bmod 5) + (3i \bmod 8)) \bmod 5$$

$$\begin{aligned} \text{Hash}(3) &= ((3 \bmod 5) + ((7 \times 3) \bmod 8)) \bmod 5 \\ &= (3 + 5) \bmod 5 = 3 \end{aligned}$$

$$\begin{aligned} \text{Hash}(10) &= ((10 \bmod 5) + ((10 \times 3) \bmod 8)) \bmod 5 \\ &= (0 + 6) \bmod 5 = 1 \end{aligned}$$

$$\begin{aligned} \text{Hash}(2) &= ((2 \bmod 5) + ((2 \times 3) \bmod 8)) \bmod 5 \\ &= (2 + 6) \bmod 5 = 3 \quad (\text{Collision!}) \end{aligned}$$

$$\text{Hash}_1(2) = 2 \bmod 5 = 2$$

$$\text{Hash}(4) = ((4 \bmod 5) + (3 \times 4 \bmod 8)) \bmod 5 = 3 \quad (\text{collision})$$

$$\text{Hash}_1(4) = 4 \bmod 5 = 4$$

0	
1	10
2	2
3	3
4	4

Question 9.1.2: reference : the slides

The reason why I chose my hash function to be  $(\text{hash} = \text{key} \bmod \text{maxSize})$  is that

For each key, there will always exist a unique index less than the maximum size of the array indicating the index of the inserted value.

Question 9.2.1

9.2

let's prove it by contradiction

Suppose we are selecting the shortest distance between 3 points among 7 points from the origin by comparing the relative distance

the greedy path choice

the optimal solution

the distance  $(a, b, c) > (c, d, f)$

but according to the greedy choice algorithm, the relative ~~paths~~ distances comparisons led to path different from the global optimum one.

the greedy choice  $(a, b, c)$  while the global optimum is  $(c, d, f)$

Question 9.2 .2

**Greedy-select ( set A)**

**set B = set A**

**While A != NULL**

**// searching for the latest start time event;**

**Do:**

**if (x2.starting\_time > x1.starting\_time)**

**// x1 and x2 are elements representing activities**

**The\_latest\_start = x2;**

**B.Remove(x1)**

**While (there is more than one event in B )**

**Remove x2 and all overlapping elements from A**

**End while**

**Return B**

For the entered unsorted set, the algorithm will copy the elements to another set first; and after that, the algorithm will remove the elements with the earlier time one by one till the set B has only one element. And the