Erica Filgueras

Exercise 1.4.5

Asked: Give the tilde approximation and order of growth of the following:

|  |  |  |
| --- | --- | --- |
|  | Tilde Approximation | Order of Growth/  Theta Complexity |
| 1. n+1 | ~n |  |
| 1. 1+ | ~1 |  |
| 1. (1+ )\*(1+) | ~1 |  |
|  | ~ |  |
|  | ~1 |  |
|  | ~2 |  |
|  | ~0 |  |

**Exercise 1.4.5 QUICK FIND**

Show the following:

1. Contents of id[] array
2. Number of times the array is accessed

Sequence: 9-0, 3-4, 5-8, 7-2, 2-1, 5-7, 0-3, 4-2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| p | q |  |  |  |  |  |  |  |  |  |  |  | Array Access  (using quick find algorithm) |
|  |  | **i** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |  |
|  |  | **id []** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |  |
| 9 | 0 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 0 | find method: 2  for loop: 10  value change: 1  **TOTAL: 13** |
| 3 | 4 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 4 | 4 | 5 | 6 | 7 | 8 | 0 | find method: 2  for loop: 10  value change: 1  **TOTAL: 13** |
| 5 | 8 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 4 | 4 | 8 | 6 | 7 | 8 | 0 | find method: 2  for loop: 10  value change: 1  **TOTAL: 13** |
| 7 | 2 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 4 | 4 | 8 | 6 | 2 | 8 | 0 | find method: 2  for loop: 10  value change: 1  **TOTAL: 13** |
| 2 | 1 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 1 | 4 | 4 | 8 | 6 | 1 | 8 | 0 | find method: 2  for loop: 10  value change: 2  **TOTAL: 14** |
| 5 | 7 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 1 | 4 | 4 | 1 | 6 | 1 | 1 | 0 | find method: 2  for loop: 10  value change: 2  **TOTAL: 14** |
| 0 | 3 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 4 | 1 | 1 | 4 | 4 | 1 | 6 | 1 | 1 | 4 | find method: 2  for loop: 10  value change: 2  **TOTAL: 14** |
| 4 | 2 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 1 | 1 | 1 | find method: 2  for loop: 10  value change: 4  **TOTAL: 16** |

**Exercise 1.5.2 QUICK UNION**

Sequence: 9-0, 3-4, 5-8, 7-2, 2-1, 5-7, 0-3, 4-2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| p | q |  |  |  |  |  |  |  |  |  |  |  | Array Access  (using quick find algorithm) |
|  |  | **i** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |  |
|  |  | **id []** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |  |
| 9 | 0 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 3 | 4 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 4 | 4 | 5 | 6 | 7 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 5 | 8 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 4 | 4 | 8 | 6 | 7 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 7 | 2 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 4 | 4 | 8 | 6 | 2 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 2 | 1 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 1 | 4 | 4 | 8 | 6 | 2 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 5 | 7 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 1 | 4 | 4 | 8 | 6 | 2 | 1 | 0 | find method: 5  value change: 1  **TOTAL: 6** |
| 0 | 3 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 4 | 1 | 1 | 4 | 4 | 8 | 6 | 2 | 1 | 0 | find method: 3  value change: 1  **TOTAL:** |
| 4 | 2 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 4 | 1 | 1 | 4 | 1 | 8 | 6 | 2 | 1 | 0 | find method: 3  value change: 1  **TOTAL:** |

Exercise 1.5.3 WEIGHTED QUICK UNION

Sequence: 9-0, 3-4, 5-8, 7-2, 2-1, 5-7, 0-3, 4-2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| p | q |  |  |  |  |  |  |  |  |  |  |  | Array Access  (using quick find algorithm) |
|  |  | **i** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |  |
|  |  | **id []** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |  |
| 9 | 0 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 3 | 4 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 4 | 4 | 5 | 6 | 7 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 5 | 8 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 4 | 4 | 8 | 6 | 7 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 7 | 2 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 1 | 2 | 4 | 4 | 8 | 6 | 2 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 2 | 1 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 2 | 2 | 4 | 4 | 8 | 6 | 2 | 8 | 0 | find method: 2  value change: 1  **TOTAL: 3** |
| 5 | 7 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 2 | 2 | 4 | 4 | 8 | 6 | 2 | 2 | 0 | find method: 4  value change: 1  **TOTAL: 5** |
| 0 | 3 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 0 | 2 | 2 | 4 | 0 | 8 | 6 | 2 | 2 | 0 | find method: 3  value change: 1  **TOTAL: 4** |
| 4 | 2 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |
|  |  | id[] | 2 | 2 | 2 | 4 | 0 | 8 | 6 | 2 | 2 | 0 | find method: 3  value change: 1  **TOTAL: 4** |

**Extra Credit:**

1. Describe an algorithm to create a **copy** of a stack, using only the **push** and **pop** operations (1 point)

For every element in the first stack, pop every element until the stack is empty. All the popped elements in the first stack should be stored in a new array.

For every element in the new array, push every element to a new stack starting from n-1 to a new stack.

1. What is the Theta-complexity of the algorithm you have described in (a)? (1 point)

The theta complexity of this algorithm would be n. Although two for loops are used, these loops are not nested and hence, the tilde approximation of this algorithm would be 2n with an order of growth/ tilde approximation of .