Homework 3

Yuqiao Meng

October 13, 2021

1 Insertion Sort

Initial Array: [10, 7, 3, 8, 1, 9, 0]
Steps:

1. Pick up 7, it's less than 10, so it exchanges position with 10: [7, 10, 3, 8, 1, 9, 0]
2. Pick up 3, it's less than 10, so it exchanges position with 10: [7, 3, 10, 8, 1, 9, 0]
3. Pick up 3, it's less than 7, so it exchanges position with 7: [3, 7, 10, 8, 1, 9, 0]
4. Pick up 8, it's less than 10, so it exchanges position with 10: [3, 7, 8, 10, 1, 9, 0]
5. Pick up 8, it's bigger than 7, so it doesn't move: [3, 7, 8, 10, 1, 9, 0]
6. Pick up 1, it's less than 10, so it exchanges position with 10: [3, 7, 8, 1, 10, 9, 0]
7. Pick up 1, it's less than 8, so it exchanges position with 8: [3, 7, 1, 8, 10, 9, 0]
8. Pick up 1, it's less than 7, so it exchanges position with 7: [3, 1, 7, 8, 10, 9, 0]
9. Pick up 1, it's less than 3, so it exchanges position with 3: [1, 3, 7, 8, 10, 9, 0]

10. Pick up 9, it's less than 10, so it exchanges position with 10: [1, 3, 7, 8, 9, 10, 0]
11. Pick up 9, it's bigger than 8, so it doesn't move: [1, 3, 7, 8, 9, 10, 0]
12. Pick up 0, it's less than 10, so it exchanges position with 10: [1, 3, 7, 8, 9, 0, 10]
13. Pick up 0, it's less than 9, so it exchanges position with 9: [1, 3, 7, 8, 0, 9, 10]
14. Pick up 0, it's less than 8, so it exchanges position with 8: [1, 3, 7, 0, 8, 9, 10]
15. Pick up 0, it's less than 7, so it exchanges position with 7: [1, 3, 0, 7, 8, 9, 10]
16. Pick up 0, it's less than 3, so it exchanges position with 3: [1, 0, 3, 7, 8, 9, 10]
17. Pick up 0, it's less than 1, so it exchanges position with 1: [0, 1, 3, 7, 8, 9, 10]

2 Merge Sort

Initial Array: [13, 57, 39, 85, 70, 22, 64, 48]
Steps: p = 1, r = 8

1. q = (1+8)/2 = 4
    (a) Merge Sort(p,q) = (1,4), q = (1+4)/2 = 2
        i. Merge Sort(p,q) = (1,2), q = (1+2)/2 = 1
            A. Merge Sort(p,q) = (1,1) = 13, return
            B. Merge Sort(q+1,r) = (2,2) = 57, return
            C. Merge [13] and [57]: 13 is less than 57: [13, 57]
        ii. Merge Sort(q+1, r) = (3,4), q = (3+4)/2 = 3
            A. Merge Sort(p,q) = (3,3) = 39, return
            B. Merge Sort(q+1,r) = (4,4) = 85, return
            C. Merge [39] and [85]: 39 is less than 85: [39, 85]

        iii. Merge [13, 57] and [39, 85]:
            A. 13 is less than 39: [13]
            B. 39 is less than 57: [13, 39]
            C. 57 is less than 85: [13, 39, 57]
            D. one array is empty, add the rest items at the bottom: [13, 39, 57, 85]

(b) Merge Sort(q+1, r) = (5,8), q = (5+8)/2 = 6
    i. Merge Sort(p,q) = (5,6), q = (5+6)/2 = 5
        A. Merge Sort(p,q) = (5,5) = 70, return
        B. Merge Sort(q+1,r) = (6,6) = 22, return
        C. Merge [22] and [70]: 22 is less than 70: [22, 70]
    ii. Merge Sort(q+1, r) = (7,8), q = (7+8)/2 = 7
        A. Merge Sort(p,q) = (7,7) = 64, return
        B. Merge Sort(q+1,r) = (8,8) = 48, return
        C. Merge [48] and [64]: 48 is less than 64: [48, 64]
    iii. Merge [22, 70] and [48, 64]:
        A. 22 is less than 48: [22]
        B. 48 is less than 70: [22, 48]
        C. 64 is less than 70: [22, 48, 64]
        D. one array is empty, add the rest items at the bottom: [22, 48, 64, 70]

(c) Merge [13, 39, 57, 85] and [22, 48, 64, 70]:
    i. 13 is less than 22: [13]
    ii. 22 is less than 39: [13, 39]
    iii. 39 is less than 48: [13, 39, 39]
    iv. 48 is less than 57: [13, 39, 39, 48]
    v. 57 is less than 64: [13, 39, 39, 48, 57]
    vi. 64 is less than 85: [13, 39, 39, 48, 57, 64]
    vii. 70 is less than 85: [13, 39, 39, 48, 57, 64, 70]
    viii. one array is empty, add the rest items at the bottom: [13, 39, 39, 48, 57, 64, 70, 85]

3 Strassen's Method

A = $\begin{matrix} 2 & 3 \\ 1 & 2 \end{matrix}$ B = $\begin{matrix} 5 & 6 \\ 1 & 3 \end{matrix}$

Applying Strassen's Method: $A_{11} = 2, A_{12} = 3, A_{21} = 4, A_{22} = 5, B_{11} = 5, B_{12} = 6, B_{21} = 1, B_{22} = 3$

- $P_1 = A_{11}(B_{12} - B22) = 6$
- $P_2 = (A_{11} + A12)B_{22} = 15$
- $P_3 = (A_{21} + A22)B_{11} = 45$
- $P_4 = A_{22}(B_{21} - B11) = -20$
- $P_5 = (A_{11} + A22)(B_{11} + B22) = 56$
- $P_6 = (A_{12} - A22)(B_{21} + B22) = -8$
- $P_7 = (A_{11} - A21)(B_{11} - B12) = -22$
- $C_{11} = P_5 + P_4 - P_2 + P_6 = 13$
- $C_{11} = P_1 + P_2 = 21$
- $C_{11} = P_3 + P_4 = 25$
- $C_{11} = P_5 + P_1 - P_3 - P_7 = 39$

So the product is $\begin{matrix} 13 & 21 \\ 25 & 39 \end{matrix}$

4 Partition function in QuickSort

Initial Array: [13, 9, 5, 7, 3, 1, 10, 6, 11, 2]

1. p = 1, r = 10, i = p - 1 = 0, x = A[r] = 2
2. for j = p = 1 to r-1 = 9:
   (a) A[j] = A[1] = 13 > 2, return: [13, 9, 5, 7, 3, 1, 10, 6, 11, 2]
   (b) A[j] = A[2] = 9 > 2, return: [13, 9, 5, 7, 3, 1, 10, 6, 11, 2]

(c) A[j] = A[3] = 5 > 2, return: [13, 9, 5, 7, 3, 1, 10, 6, 11, 2]

(d) A[j] = A[4] = 7 > 2, return: [13, 9, 5, 7, 3, 1, 10, 6, 11, 2]

(e) A[j] = A[5] = 3 > 2, return: [13, 9, 5, 7, 3, 1, 10, 6, 11, 2]

(f) A[j] = A[6] = 1 < 2, i = i+1 = 1, exchange A[i] = A[1] with A[j] = A[6]: [1, 9, 5, 7, 3, 13, 10, 6, 11, 2]

(g) A[j] = A[7] = 10 > 2, return: [1, 9, 5, 7, 3, 13, 10, 6, 11, 2]

(h) A[j] = A[8] = 6 > 2, return: [1, 9, 5, 7, 3, 13, 10, 6, 11, 2]

(i) A[j] = A[9] = 11 > 2, return: [1, 9, 5, 7, 3, 13, 10, 6, 11, 2]

3. exchange A[i+1] = A[2] with A[r] = A[10]: [1, 2, 9, 5, 7, 3, 13, 10, 6, 11]

4. return i+1 = 2

5 Find Algorithm

Code: FindValue(A, p, r)

```
if p < r:
    q = (p+r)/2 // if the result is a decimal, just cut it.
    if A[q] = q:
        return q
    else if A[q] > q:
        return FindValue(A, p, q)
    else if A[q] < q:
        return FindValue(A, q+1, r)
else:
    if A[p] = p:
        return p
    else:
        return -1
```

Correctness:

- Loop Invariant: At the state of each iteration of the algorithm, if there is an index meets the requirement, it must be in the subarray A[p, ... , r]

- Initialization: At the start of the iteration, if A contains more than 2 items, p must be less than r, otherwise just return if the only item A[1] == 1
- Maintenance: if $p < r$, then let q = (p+r)/2, it's the middle item of the array, then check if A[q] == q
  - A[q] == q: find the index meets the requirement
  - A[q] > q: for every item, its index increases or decreases by 1, but its value can increase or decrease more quickly, and all the integers are distinct, so if there is an index meets the requirement, it can't be bigger than q. Because even though integers increase by 1, it can't equal to its index which increasing at the same speed. Thus FindValue(A, p, q)
  - A[q] < q: for every item, its index increases or decreases by 1, but its value can increase or decrease more quickly, and all the integers are distinct, so if there is an index meets the requirement, it can't be less than q. Because even though integers decrease by 1, it can't equal to its index which decreasing at the same speed. Thus FindValue(A, q+1, r)
  - Termination: At Termination, if there isn't an index meets the requirement, p will eventually equal to r, and return A[p] == p.

Running time Analysis: For every iteration, if the index has been found, just return; if not, exclude half of the array and keep finding in the other half array. So every iteration can reduce the n to $\frac{n}{2}$, $T(n) = 2T(n/2) + T(1)$, by applying master theorem, $T(n) = O(\log n)$

6 Loaded Die

Assume that all the default probability is p, and the die has six faces, then

- P(1) = $\frac{p}{1}$
- P(2) = $\frac{p}{2}$
- P(3) = $\frac{p}{3}$
- P(4) = $\frac{p}{4}$

- P(5) = $\frac{p}{5}$
- P(6) = $\frac{p}{6}$
- P(1)+P(2)+P(3)+P(4)+P(5)+P(6) = 1

We can solve that default probability p = $\frac{60}{147}$, so P(greater than 3) = P(4)+P(5)+P(6) = $\frac{37}{147}$