CS146 Data Structures and Algorithms,

HW #1

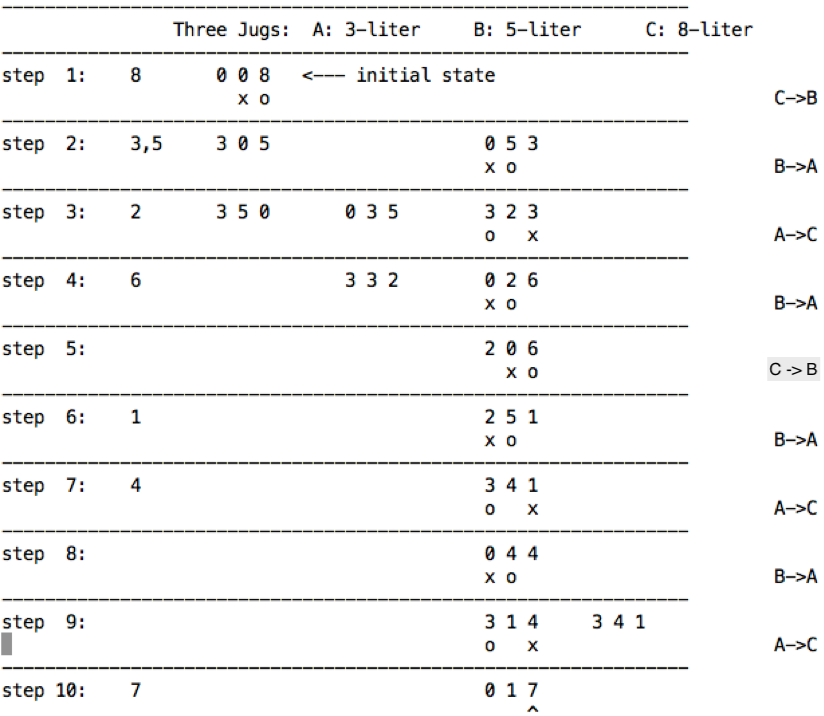
(Last updated on 9/7/2019)

* All questions are either multiple-choice or multiple-answer.
* All questions are equally weighted (2.5pt each)
* Highlight your chosen answers in red colors.
* Submit this document with your answers highlighted in red.

**The Three Jugs Problem**: You have an empty 3-gallon jug (label A) and an empty 5-gallon jug (labeled B). You also have an 8-gallon jug (labeled C) that is full of water.

You would like to give your friend exactly 7-liter of the water.

The following are the partial works to help you measure each amount of water.



Notations: o -- The jug to let out the water x -- The jug to receive the water

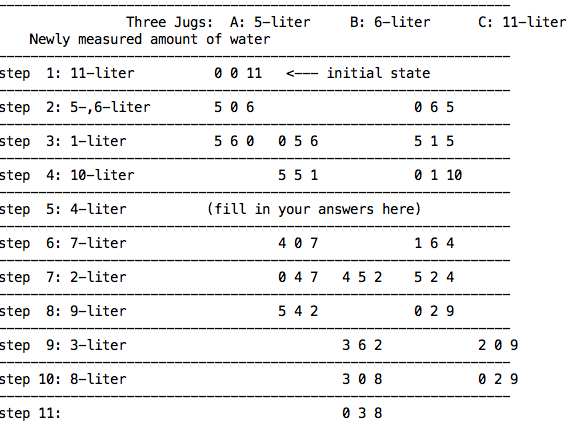
1. What is the sequence of the steps to measure 7-liter water?
2. C🡪B, B🡪A, A🡪C, B🡪A, C🡪B, B🡪A, A🡪C, B🡪A, A🡪C,
3. C🡪B, A🡪B, A🡪C, B🡪A, C🡪A, B🡪A, A🡪C, B🡪A, A->C,
4. C🡪B, A🡪B, A🡪C, B🡪A, C🡪A, B🡪A, C🡪A, B🡪A, A🡪C,
5. C🡪B, A🡪B, A🡪C, B🡪A, C->🡪, B🡪A, C🡪A, B🡪A, A🡪B,
6. none of the above

1. Which jug has 7-liter water?
2. Jug C.
3. Jug B
4. Jug A
5. none of the above

**The Three Jugs Problem** (again):

You have an empty 5-gallon jug (labeled A) and an empty 6-gallon jug (labeled B). You also have an 11-gallon jug (labeled C) that is full of water. The goal is to exactly measure each amount of the water with minimum number of steps.

The following are the partial works to help you measure each amount of water.



1. What are the newly generated states at step 5?
2. 4 6 1 1 0 10
3. 0 5 6 4 6 1 1 0 10
4. 0 5 6 4 6 1 0 0 11
5. none of the above
6. What is the run time complexity for Insertion sort in the average case?
7. O(1)
8. O(n)
9. O(n2)
10. none of the above
11. Which of the following **are** correct about run time complexity for Insertion sort in the best case? (Choose two)?
    1. The best case run time is O(1)
    2. The best case run time is O(n)
    3. The best case run time is O(n2)
    4. The best case happens when the numbers in the array are in the sorted order.
    5. The best case happens when the numbers in the array are in the reversely sorted order.
    6. none of the above
12. Which of the following is correct about run time complexity for selection sort in the best case?
13. The best case run time is O(1)
14. The best case run time is O(n)
15. The best case run time is O(n2)
16. The best case happens when the numbers in the array are in the sorted order.
17. The best case happens when the numbers in the array are in the reversely sorted order.
18. none of the above
19. For the average case, insertion sort could have more swapping operation than the selection sort.
20. True.
21. False.
22. The following is the correct operation sequence of the \_\_\_\_\_\_\_\_\_\_\_\_\_ sort.
23. Insertion
24. Selection
25. Quick
26. Merge
27. none of the above

Input: 3 4 5 1 6 2

step1: 1 4 5 3 6 2

step2: 1 2 5 3 6 4

step3: 1 2 3 5 6 4

step4: 1 2 3 4 6 5

step5: 1 2 3 4 5 6

1. Is the following the correct operation sequence of the insertion sort?

**3 4 5 1 6 2**

**3 4 5 1 6 2**

**3 4 5 1 6 2**

**3 4 5 1 6 2**

**1 3 4 5 6 2**

**1 3 4 5 6 2**

1. True
2. False

Shown in the following are operation sequences that correspond to the selection sort or the insertion sort.

M1:

Input : 5 4 3

step 1: 4 5 3

step 2: 3 4 5

M2:

Input : 5 4 3

step 1: 3 4 5

step 2: 3 4 5

M3:

Input : 2 3 1

step 1: 2 3 1

step 2: 1 2 3

M4:

Input : 2 3 1

step 1: 1 3 2

step 2: 1 2 3

1. Choose the cases which are insertion sorts.
2. M1
3. M2
4. M3
5. M4
6. none of the above
7. The sum of the geometric series 20 + 21 + 22 + … + 2h-1 is
8. 2h - 1
9. 2h + 1
10. 2h
11. 2h+1
12. none of the above
13. If N = 21 + 22 + … + 2h, then h =
14. O(log2 N)
15. O(N log2 N)
16. O(N)
17. none of the above
18. Prove n ∈ O(n2) by choosing some c>0 and n0 such that n ≤ cn2 for n≥ n0. Which of the following is NOT the correct choices of c and n0?
19. c=1, n0=2
20. c= 1/4, n0= 3
21. c= 10, n0= 2
22. c= 1/20, n0= 20
23. c=1, n0=1
24. none of the above
25. Prove 1000\*n ∈ O(n2) by choosing some c>0 and n0 such that 1000\*n ≤ c n2 for n ≥ n0. Which of the following is NOT the correct choice of c and n0?
26. c=1, n0=1000
27. c=1000, n0=1
28. c=1000, n0=1000
29. c=1, n0=1
30. none of the above
31. Prove n2+10n + 5 ∈ O(n2) by choosing some c>0 and n0 such that

n2+10n + 5 ≤ c \* n2 for n≥ n0. Which of the following is NOT the correct choices of c and n0?

1. C=20, n0=1
2. C=3, n0=1000
3. C=18, n0=1
4. C=4, n0=1000
5. C=2, n0=3
6. none of the above
7. Which of the following is NOT true?
8. (n2+ n) ∈ O(n2)
9. (n2/1000 + n) ∈ O(n2)
10. n2/1000 ∈ o(n2) (o is the small Oh)
11. 15n ∈ o(n2)
12. none of the above
13. Prove 1000\*n ∈ o(n2) by choosing n0 such that for any given c> 0, there exits n0 such that 0 ≤ 1000\*n < c\*n2 for all n >= n0. Which of the following is the correct choices of c and n0?
14. n0 > 1000/c, for all c >0
15. n0 > 100/c, for all c >0
16. n0 > 1000, for all c >0
17. n0 > 2000, for all c >0
18. none of the above
19. Which of the following is false?
20. n/10 ∈ Ω(n)
21. n/10 ∈ O(n)
22. n/10 ∈ θ(n)
23. n/10 ∈ ω(n)
24. none of the above
25. Which of the following is (are) NOT true?
26. n/1000 ∈ θ(n)
27. n2+ n ∈ Θ (n2+1000n)
28. 100n2+ n ∈ Θ (n2)
29. log (n3) ∈ Θ (n3)
30. log(n3) + ∈ Θ ()
31. all are true

1. We can prove that 1000n ω(n) as follows:

For some values of c, there exists no n0  such that 1000n > c\*n, n ≥ n0.

What value of c can be used for this proof?

1. c=1
2. c=10
3. c=100
4. c=1000
5. none of the above
6. Which of the following is true?
7. log2n ∈ o(log10 n )
8. 2n ∈ Θ(3n)
9. 3n ∈ Θ (n!)
10. ∈ ω( log10 n )
11. none of the above

1. Choose the correct statements.
2. 15n3 log n + 10n2 + 50 ∈ O(n3 log n)
3. 3n2− 12n + 2 ∈ Ω(n3)
4. 2n+1  ∈ ω (2n)
5. 2n+1  ∈ Θ(2n)
6. None of the above
7. Which of the following is (are) false?
8. 22n ∈ O(2n)
9. log(n!) ∈ O(n log n)
10. 2n+1 ∈ Θ(2n)
11. 22n ∈ Θ (2n)
12. 9n3+ 12n ∈ o(2n)
13. none of the above
14. Which of the following is true?
15. 10n2 + 50 ∈ O(n log n)
16. 3n2 + 12n + 2 ∈ Ω(n3)
17. If f(n) =  (g(n)), then g(n) =  (f(n)).
18. none of the above
19. Which of the following is true?
20. If f(n) = O(g(n)), 🡺 f(n) = (g(n)).
21. If f(n) = (g(n)), 🡺 f(n) = (g(n)).
22. If f(n) = (g(n)), 🡺 g(n) = (f(n)).
23. If f(n) = (g(n)), 🡺 g(n) = (f(n)).
24. none of the above
25. Which of the following is NOT true?
26. n2 ∈ Θ(n2)
27. n2 + n ∈ Θ(n2)
28. 100n2 + n ∈ Θ(n2)
29. 100n2 + lg n ∈ Θ(n2)
30. n lg n ∈ Θ(n2)
31. none of the above
32. Which of the following is NOT true?
33. if f(n) = Θ (g(n)), 🡺 f(n) = O(g(n)) and f(n) = Ω(g(n))
34. if f(n) = O(g(n)) and f(n) = Ω(g(n)), 🡺 f(n) = Θ (g(n))
35. if f(n) = o(g(n), 🡺 f(n) = O(g(n))
36. if f(n) = O(g(n)), 🡺 g(n) = Ω(f(n))
37. if f(n) = O(g(n), 🡺 f(n) = o(g(n))
38. none of the above
39. Which of the following are true? (multiple answers)
40. log2n =  (log10n)
41. log2n = Θ (log10n)
42. 3n =  (2n)
43. 3n = Θ (2n)
44. n! =  (3n)
45. n! = Θ (3n)
46. n0.5 =  (log10n)
47. n0.5 = Θ (log10n)
48. none of the above

======= Merge Sort =======

The key step of merge sort is to merge two sorted arrays in linear time.

Consider following example of merging two sorted arrays.

Step 0:

Array 1: 1 3 5 7 (The number underlined is the next one to be compared.)

Array 2: 2 4 6 8

Output: 1,

Step 1:

Array 1: 3 5 7

Array 2: 2 4 6 8

Output: 1, 2

Step 2:

Array 1: 3 5 7

Array 2: 4 6 8

Output: 1, 2, 3

…

1. What will be the two numbers to be compared at step 3?
   1. 4 and 5
   2. 3 and 6
   3. 4 and 3
   4. 5 and 6
   5. none of the above
2. What will be the two numbers to be compared at the last step 6?
   1. 7 and 8
   2. 7 and 6
   3. 5 and 8
   4. 5 and 6
3. none of the above
4. How many comparisons will be performed for two sorted arrays with n and m numbers respectively?
   1. n+m-1
   2. n
   3. m
   4. n\*m
   5. none of the above
5. Given the following recurrence function: use the iteration method to get T(n).

T(n)= T(n-1) +n/2, n>=1

T(0) = 0

T(n) = T(n-1) +n/2 = T(n-2) + (n-1)/2 + n/2 = T(n-3) + (n-2)/2 + (n-1)/2 + n/2

= T(n-k-1) + (n-k)/2 + … + (n-2)/2 + (n-1)/2 + n/2

= T(0) + ½ + 2/2 + 3/2 + … + n/2

Based on the above work, what is the T(n)?

1. T(n) = Θ (n2)
2. T(n) = Θ (n/2)
3. T(n) = Θ (nlog n)
4. T(n) = Θ (log n)
5. none of the above
6. Given recurrence function: T(n)=4T(n/2) +n, use the iteration method to get T(n) as follow

T(n)= 4T(n/2) +n

= 4 (4T(n/4) + n/2) +n = 16T(n/4) + 3n = 42(n/22) + (22-1)n

= 16(4T(n/8) + n/4) +3n = 64T(n/8) + 7n = 43(n/23) + (23-1)n

= .... = 44(n/24) + (24-1)n

= .... = 4k(n/2k) + (2k-1)n

Let n =2k, then k = log2 n, and we have T(n)=

1. n2 + (n-1)n
2. n + (n-1)n
3. 2n2 + (n-1)n
4. 2n + (n-1)n
5. none of the above
6. Given the following recurrence function, use iteration method to get T(n).

T(n) = T(n-1) + 1, for n > 0

= 0, for n=0.

T(n)=T(n-1)+1 = T(n-2) + 1 + 1 = T(n-3) + 1 + 1 + 1 = T(n-k) + k, k = 1,2,3,…, n

Based on the above work, what is the order of T(n)?

1. T(n) = Θ (n2 )
2. T(n) = Θ (n)
3. T(n) = Θ (n1/2)
4. T(n) = Θ (n log n)
5. T(n) = Θ (log n)
6. none of the above
7. Given the following recurrence function, use iteration method to get T(n).

T(n) = T(n/2) +1, if n > 1

= 0 , if n = 1

T(n) = T(n/2) + 1

= T(n/4) + 1 + 1

= T(n/8) + 1 + 1 + 1

= T(n/16) + 1 + 1 + 1 + 1

...

T(n) = T(n/2 ) + 1

= T(n/22) + 2

= T(n/23) + 3

= ...

= T(n/2k) + k

When n= 2k, we have T(n)=T(n/2 log2n) + log2n

Based on the above work, what is the order of T(n)?

1. T(n) = Θ ( n2 )
2. T(n) = Θ ( n)
3. T(n) = Θ ( n1/2)
4. T(n) = Θ ( n log n)
5. T(n) = Θ ( log n)
6. none of the above
7. Given the following recurrence function, use iteration method to get T(n).

T(n) = T(n/2) + n, if n > 0

= 0 , if n = 0

*T(n) = T(n/2) + n=T*(*n/22* )*+ n/2 + n=T*(*n/2k* )*+n/2k-1  + … + n/2 + n*

Let *n=2k ,* then we have *T(n)=T(1) + 2 + 4 + … +2k*

Based on the above work, what is the order of T(n)?

1. T(n) = Θ ( n2 )
2. T(n) = Θ ( n)
3. T(n) = Θ ( n1/2)
4. T(n) = Θ ( n log n)
5. T(n) = Θ ( log n)
6. none of the above

**Master theorem**

Recall that for the recurrence function *T*(*n*) = *a T*(*n*/*b*) + *f* (*n*), the master theorem states that

Case 1: f (n) = O(nlogba –ε) 🡺 T(n) = Θ(nlogba)

Case 2: f (n) = Θ (nlogba) 🡺T(n) = Θ(nlogba log n)

Case 3: f (n) = Ω (nlogba + ε) and a \*f (n/b) < c f (n) 🡺 T(n) = Θ( f (n))

1. Given the following recurrence function, use Master method solve the following.

*T(n) = T(n/2) + n1/2*

Based on the above work, which case of Master Theorem applies here?

1. Case I
2. Case 2
3. Case 3
4. Master theorem does not apply here.
5. Given the following recurrence function, use Master method solve the following.

T(n) = 9T(n/3) + n2

Based on the above work, which case of Master Theorem applies here?

1. Case I
2. Case 2
3. Case 3
4. Master theorem does not apply here.
5. Given the following recurrence function, use Master Theorem to solve the following.

T(n) = 6T(n/4) + n

Based on the above work, which case of Master Theorem applies here?

1. Case I
2. Case 2
3. Case 3
4. Master theorem does not apply here.
5. Given recurrence function, use Master theorem Case 1 to solve

T(n)=4T(n/2) +1

Note that a=4, b=2, so we get logba = 2

f(n) = 1 <= n2-e, for some 0<e<=2. So, we have T(n) = Θ (nlogba) .

Based on the above work, what is the T(n)?

1. T(n) = Θ ( n2 )
2. T(n) = Θ ( n/2)
3. T(n) = Θ (log n)
4. T(n) = Θ (nlog n)
5. none of the above
6. (Bonus) How do you feel this assignment? (5% on top of your grade.)

(Challenging questions for your amusements.) (0pt)

1. (0pt) Given the following recurrence function, use Master method to get T(n).

T(n) = 2T(n-2) + n, for n > 0

= 0, for n=0

Let n = log m, i.e., m = 2n. Then we have

T(n) = 2T(n-2) + 1 => T(log m) = 2\*T(log m -2) + 1 = 2\*T(log (m/4) ) + 1

Let S(m) = T(log m) = T(n)

=> S(m) = 2 \* S(m/4) + 1

=> S(m) = Θ ( m1/2 ) 🡨 (Master method Case I)

=> T(n) = S(m) = ?

Based on the above work, what is the T(n)?

1. T(n) = Θ ( 2n/2 )
2. T(n) = Θ ( 2n)
3. T(n) = Θ ( 22n)
4. T(n) = Θ ( 2n+1)