Problem 1

Prove $\log(n!) = \Theta(n \log n)$

Problem 2 (20 points)

Resources used:
Collaborators:

The Fibonacci numbers $F_0, F_1, F_2, \ldots,$ are defined by the rule

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}.$$

a) Use induction to prove that $F_n \geq 2^{0.6n}$ for all $n \geq 13$.

b) Use induction to prove that $F_n \leq 2^{0.7n}$ for all $n \geq 0$.

(a) Base Case

$F_{13} = 233$ $\quad 2^{0.6(13)} = 222.86$ $\quad \boxed{233 \geq 222.86}$

$F_{14} = 377$ $\quad 2^{0.6(14)} = 337.794$ $\quad \boxed{377 \geq 337.794}$

Inductive Hyp

For all $n \geq 13$: $\quad F_n \geq 2^{0.6(n)}$

$\Rightarrow \quad F_{n-1} \geq 2^{0.6(n-1)}$

Inductive Step

$F_{n+1} \geq 2^{0.6(n+1)}$

$\Rightarrow F_n + F_{n-1} \geq 2^{0.6(n+1)}$

According to hyp $\rightarrow$ $F_n \geq 2^{0.6n}$ and $F_{n-1} \geq 2^{0.6(n-1)}$

The min value is $\geq$ min value of F_{n+1}

$\Rightarrow 2^{0.6n} + 2^{0.6(n-1)} \geq 2^{0.6(n+1)}$

$\Rightarrow \quad \geq 2^{0.6(n-}$

2 b

Problem 3 (10 Points)

Resources used:
Collaborators:

Use mathematical induction to show that if n is an exact power of 2 and if $T(n) = 2T(n/2) + n$ for $n > 2$ with $T(2) = 2$, then $T(n) = n \log_2 n$.

$T(n) = 2T(n/2) + n \qquad n > 2$

Base Case

$4 = 2^2$ ✓ $\qquad T(4) = 4 \log_2(2) = 4 \cdot 2 = 8$ $\boxed{✓}$

Inductive Hyp

$T(2n) = 2n \log_2(2n)$

Maybe $T(n) = n \log_2(n)$

Inductive Step

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

$= 2\,T(n) + n$

$= 2\left(n \log_2(n)\right) + n$

$= 2n \cdot \log_2(n^2) + n$

$= 2n$

Problem 4 Insertion sort.

Compare

Swap

7 6 5 4 3 2 1

6 7 5 4 3 2 1

6 5 7 4 3 2 1

5 6 7 4 3 2 1

5 6 4 7 3 2 1

5 4 6 7 3 2 1

4 5 6 7 3 2 1

4 5 6 3 7 2 1

4 5 3 6 7 2 1

4 3 5 6 7 2 1

3 4 5 6 7 2 1

3 4 5 6 2 7 1

3 4 5 2 6 7 1

3 4 2 5 6 7 1

3 2 4 5 6 7 1

2 3 4 5 6 7 1

2 3 4 5 6 1 7

2 3 4 5 1 6 7

2 3 4 1 5 6 7

2 3 1 4 5 6 7

2 1 3 4 5 6 7

$\boxed{1\ 2\ 3\ 4\ 5\ 6\ 7}$

(c)

7 6 5 4 3 2 1

1 7 6 5 4 3 2

1 2 7 6 5 4 3

1 2 3 7 6 5 4

1 2 3 4 7 6 5

1 2 3 4 5 7 6

1 2 3 4 5 6 7

Problem 4e

Compare

Swap

gap = 8

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
8	15	14	13	12	11	10	9	16	7	6	5	4	3	2	1
8	7	14	13	12	11	10	9	16	15	6	5	4	3	2	1
8	7	6	13	12	11	10	9	16	15	14	5	4	3	2	1
8	7	6	5	12	11	10	9	16	15	14	13	4	3	2	1
8	7	6	5	4	11	10	9	16	15	14	13	12	3	2	1
8	7	6	5	4	3	10	9	16	15	14	13	12	11	2	1
8	7	6	5	4	3	2	9	16	15	14	13	12	11	10	1

Gap : 4

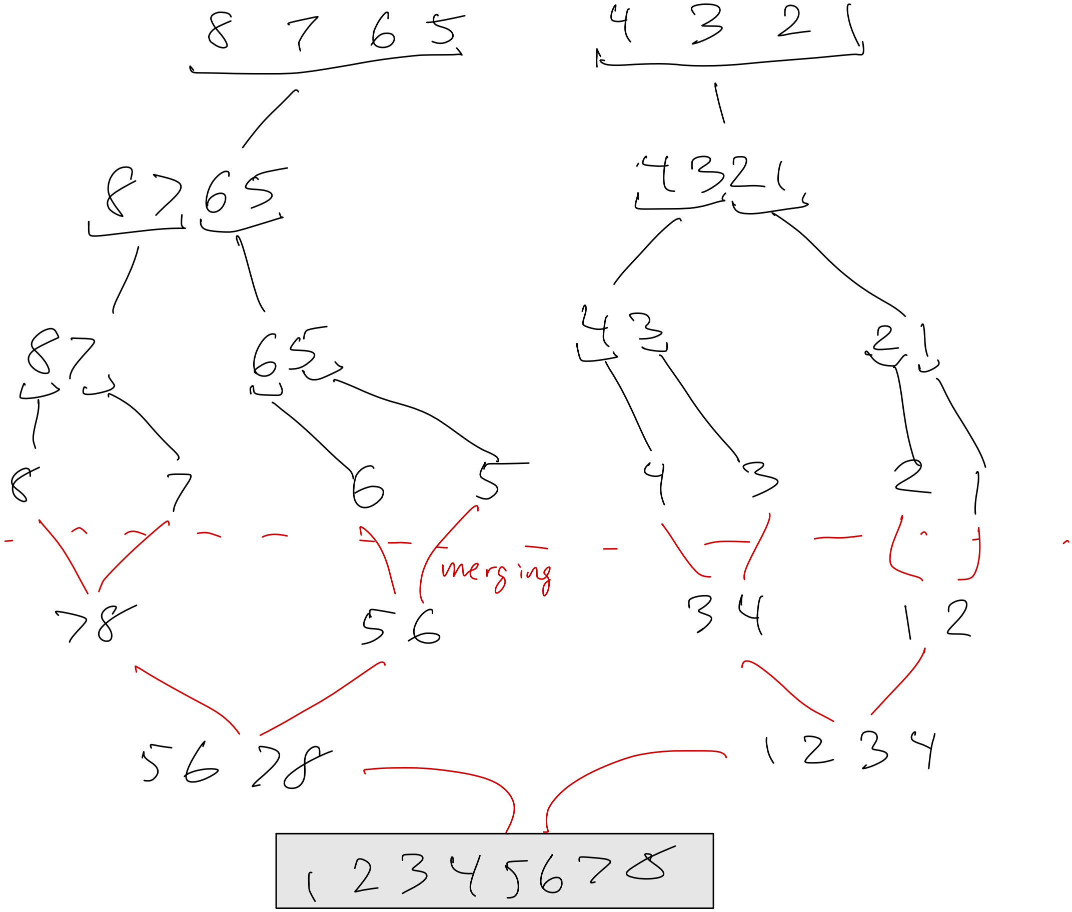
8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9
4	7	6	5	8	3	2	1	16	15	14	13	12	11	10	9
4	3	6	5	8	7	2	1	16	15	14	13	12	11	10	9
4	3	2	5	8	7	6	1	16	15	14	13	12	11	10	9
4	3	2	1	8	7	6	5	16	15	14	13	12	11	10	9
4	3	2	1	8	7	6	5	16	15	14	13	12	11	10	9
4	3	2	1	8	7	6	5	16	15	14	13	12	11	10	9
4	3	2	1	8	7	6	5	16	15	14	13	12	11	10	9
4	3	2	1	8	7	6	5	16	15	14	13	12	11	10	9
4	3	2	1	8	7	6	5	12	15	14	13	16	11	10	9
4	3	2	1	8	7	6	5	12	11	14	13	16	15	10	9
4	3	2	1	8	7	6	5	12	11	10	13	16	15	14	9

gap = 2

4	3	2	1	8	7	6	5	12	11	10	9	16	15	14	13
2	3	4	1	8	7	6	5	12	11	10	9	16	15	14	13
2	1	4	3	8	7	6	5	12	11	10	9	16	15	14	13
2	1	4	3	8	7	6	5	12	11	10	9	16	15	14	13

	2	1	4	3	8	7	6	5	12	11	10	9	16	15	14	13
	2	1	4	3	6	7	8	5	12	11	10	9	16	15	14	13
	2	1	4	3	6	5	8	7	12	11	10	9	16	15	14	13
	2	1	4	3	6	5	8	7	12	11	10	9	16	15	14	13
	2	1	4	3	6	5	8	7	12	11	10	9	16	15	14	13
	2	1	4	3	6	5	8	7	10	11	12	9	16	15	14	13
	2	1	4	3	6	5	8	7	10	9	12	11	16	15	14	13
	2	1	4	3	6	5	8	7	10	9	12	11	16	15	14	13
	2	1	4	3	6	5	8	7	10	9	12	11	16	15	14	13
	2	1	4	3	6	5	8	7	10	9	12	11	14	15	16	13
gap = 1	2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15
	1	2	4	3	6	5	8	7	10	9	12	11	14	13	16	15
	1	2	4	3	6	5	8	7	10	9	12	11	14	13	16	15
	1	2	3	4	6	5	8	7	10	9	12	11	14	13	16	15
	1	2	3	4	6	5	8	7	10	9	12	11	14	13	16	15
	1	2	3	4	5	6	8	7	10	9	12	11	14	13	16	15
	1	2	3	4	5	6	8	7	10	9	12	11	14	13	16	15
	1	2	3	4	5	6	7	8	10	9	12	11	14	13	16	15
	1	2	3	4	5	6	7	8	10	9	12	11	14	13	16	15
	1	2	3	4	5	6	7	8	9	10	12	11	14	13	16	15
	1	2	3	4	5	6	7	8	9	10	12	11	14	13	16	15
	1	2	3	4	5	6	7	8	9	10	11	12	14	13	16	15
	1	2	3	4	5	6	7	8	9	10	11	12	14	13	16	15
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16	15
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16	15
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

4 (f)


8 7 6 5
4 3 2 1
8765
4321
87
65
43
21
8
7
6
5
4
3
2
1
merging
78
56
34
12
5678
1234
12345678

5 (a)

3	4	9	7	2
0	1	2	3	4

(0, 4)
(1, 4)
(2, 3)
(2, 4)
(3, 4)

(b) Reverse array {n, n-1, n-2 ... 2, 1}

- This has n elements
- 1 none
  2 1 inv
  3 2 inv

1 + 2 + ... n - 1 inversion

$$\sum_{n=1}^{n} \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$$

(c) As the inversion count increases, the runtime for insertion sort increase. Insertion sort is based on swapping a previous larger item with a smaller item after. (inversion) The more inversions, the more swaps insertion sort would have to make.

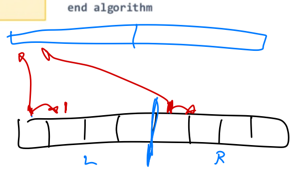
Merge Sort Algorithm

```
algorithm MergeSort(A, l, r)

   if (l < r)
      m = (l + r) / 2
      MergeSort(A, l, m)
      MergeSort(A, m + 1, r)
      merge(A, l, m, r)
   end if

   return A
end algorithm
```

- **Time Complexity:** $T(n) \in \Theta(n \log_2(n))$
- **Space Complexity:** $\Theta(n)$

```
algorithm merge(A, l, m, r)
   n1 = m - l + 1
   n2 = r - m
   let L be an array of size n1 + 1
   let R be an array of size n2 + 1

   for (i = 0 to n1 - 1)
      L[i] = A[l + i]
   end for
   for (j = 0 to n2 - 1)
      R[j] = A[m + j + 1]
   end for

   L[n1] = ∞, R[n2] = ∞
   i = 0, j = 0

   for (k = l to r - 1)
      if (L[i] <= R[j])
         A[k] = L[i]
         i = i + 1
      else
         A[k] = R[j]
         j = j + 1
      end if
   end for

   return A
end algorithm
```

3



5 d