

General instructions. First read [Lecture Note 4](#), then answer the problems given in the rest of this document. Edit this LaTeX template to add your answers, as described in the file named `00_README.txt`, then compile it to produce [homework.2.pdf](#).

Don't know? Say so. If you don't know the answer to a problem or any part of the problem, we encourage you to simply write “*I don't know*” instead of giving an answer. (If you write “I don't know” you can also add any questions you might have.)

Gentle reminder: homework is for learning. As noted in [Lecture Note 1](#), engaging fully with the homeworks is the best way to learn the material. *The primary role of homeworks is to help you learn the material, and the primary role of feedback on homeworks is to help you improve your ideas.*

Please try to explain your ideas as clearly as you can. The clearer your explanations are, the more useful the feedback on them can be.

Collaboration and citation policy. If you are doing this homework as part of a course, follow the course's collaboration and citation policy. In any case, at the end of each answer, list the people you collaborated with, and cite every source that your answer uses ideas from (other than the lecture notes and the sources listed in the lecture notes). If there are none, write “none”.

Problem 1. (Section 1.4) Reorder this sequence so each function is big-O of the following function:

$$n^2 \quad 6^{\log_2 n} \quad n! \quad 11^n \quad \log_2(n!) \quad 2^{3n} \quad 10n^{3/2} \quad 100n \quad 3^{\log_4 n} \quad \log_2^4 n$$

No justification is required. Give the reordered sequence below. For grading, order the functions vertically, one on each line.

Problem 1 answer

$$n^2$$

$$6^{\log_2 n}$$

$$n!$$

$$11^n$$

$$\log_2(n!)$$

$$2^{3n}$$

$$10 n^{3/2}$$

$$100 n$$

$$3^{\log_4 n}$$

$$\log_2^4 n$$

Problem 1 collaboration and citations I collaborated on this problem with the following people:

• ★ ⋯ ★

My answers use ideas from the following sources (other than the lecture notes and textbooks):

• ★ ⋯ ★

Problem 2. (Section 4.1) Use the so-called naive upper and lower bounds, and the rule for geometric sums, to determine the big- Θ value of each of the following sums. You don't have to show your work, but you can if you want feedback on it (and it may help you get partial credit if your answer is wrong).

(a) $\sum_{i=1}^{\log_2 n} i(\log_3 i)^2$

(b) $\sum_{i=1}^n i^2/3^i$

(c) $\sum_{i=1}^{2^n} (\log_2 i)^2$

(d) $\sum_{i=1}^n 1.2^i/i$

Problem 2 answer

(a) $\sum_{i=1}^{\log_2 n} i(\log_3 i)^2$ is $\Theta(\star \cdots \star)$

(b) $\sum_{i=1}^n i^2/3^i$ is $\Theta(\star \cdots \star)$

(c) $\sum_{i=1}^{2^n} (\log_2 i)^2$ is $\Theta(\star \cdots \star)$

(d) $\sum_{i=1}^n 1.2^i/i$ is $\Theta(\star \cdots \star)$

Problem 2 collaboration and citations I collaborated on this problem with the following people:

• $\star \cdots \star$

My answers use ideas from the following sources (other than the lecture notes and textbooks):

• $\star \cdots \star$

Problem 3. (*Recursion-tree method*) Repeat Exercise 4.2 for each of the algorithms below. Give your answers in the same format as the solution in Lecture Note 4 for that exercise. To make it easier, ignore floors in your calculations. That is, for (a) and (c) assume n is a power of 3. For (b) assume n is a power of 4.

- (a) def PrintAs(n): *Assume n is a power of 3.*
 1. for $i = 1$ to $3n$: print("A") *... print $3n$ A's*
 2. if $n > 1$:
 3. for $j = 1$ to 4: PrintAs($\lfloor n/3 \rfloor$) *... make 4 recursive calls, each with input $\lfloor n/3 \rfloor$*
- (b) def PrintBs(n): *Assume n is a power of 4.*
 1. for $i = 1$ to n^2 : print("B") *... print n^2 B's*
 2. if $n > 1$:
 3. for $j = 1$ to 16: PrintBs($\lfloor n/4 \rfloor$) *... make 16 recursive calls, each with input $\lfloor n/4 \rfloor$*
- (c) def PrintCs(n): *Assume n is a power of 3.*
 1. for $i = 1$ to $5n$: print("C") *... print $5n$ C's*
 2. if $n > 1$:
 3. for $j = 1$ to 2: PrintCs($\lfloor n/3 \rfloor$) *... make 2 recursive calls, each with input $\lfloor n/3 \rfloor$*

(continued)

Problem 3(a) answer

- (i) Define $A(n)$ to be the number of letters printed by PRINTAS given input $n \geq 1$.

Then $A(1) = \star \cdots \star$

For $n > 1$, $A(n) = \star \cdots \star$

- (ii) Within level i :

The number of nodes is $\star \cdots \star$

The problem size is $\star \cdots \star$

The number of letters printed by each call is $\star \cdots \star$

- (iii) The number of levels is $\star \cdots \star$

- (iv) The total number of letters printed is

$$A(n) = \star \cdots \star.$$

- (v) Using $\star \cdots \star$, the big- Θ -value of $A(n)$ is

$$\star \cdots \star.$$

(continued)

Problem 3(b) answer

- (i) Define $B(n)$ to be the number of letters printed by PRINTBs given input $n \geq 1$.

Then $B(1) = \star \cdots \star$

For $n > 1$, $B(n) = \star \cdots \star$

- (ii) Within level i :

The number of nodes is $\star \cdots \star$

The problem size is $\star \cdots \star$

The number of letters printed by each call is $\star \cdots \star$

- (iii) The number of levels is $\star \cdots \star$

- (iv) The total number of letters printed is

$$B(n) = \star \cdots \star.$$

- (v) Using $\star \cdots \star$, the big- Θ value of $B(n)$ is

$$\star \cdots \star.$$

(continued)

Problem 3(c) answer

- (i) Define $C(n)$ to be the number of letters printed by PRINTCs given input $n \geq 1$.

Then $C(1) = \star \cdots \star$

For $n > 1$, $C(n) = \star \cdots \star$

- (ii) Within level i :

The number of nodes is $\star \cdots \star$

The problem size is $\star \cdots \star$

The number of letters printed by each call is $\star \cdots \star$

- (iii) The number of levels is $\star \cdots \star$

- (iv) The total number of letters printed is

$$C(n) = \star \cdots \star.$$

- (v) Using $\star \cdots \star$, the big- Θ value of $C(n)$ is

$$\star \cdots \star.$$

Problem 3 collaboration and citations I collaborated on this problem with the following people:

- $\star \cdots \star$

My answers use ideas from the following sources (other than the lecture notes and textbooks):

- $\star \cdots \star$

Problem 4. (*recursion trees*) Consider a variant of mergesort that, given an array of size N , recurses as usual unless the subproblem to be solved is of size $\sqrt{N}$ or less, in which case it solves the subproblem without recursing, instead using insertion sort (a quadratic-time algorithm) on the subproblem. Use a recursion-tree analysis to determine the asymptotic worst-case running time of this algorithm on arrays of size N . Show your work as described for Problem 3.

You may assume N is an even power of two. That is $N = 2^{2k}$ for some integer k , so $\sqrt{N} = 2^k$.

Problem 4 answer

Fix the input size N .

- (i) For $n \leq N$, define $T(n)$ to be run-time on sub-arrays of size n .

Then, for $n \leq \sqrt{N}$, we have $T(n) = \star \cdots \star$

For $n > \sqrt{N}$, we have $T(n) = \star \cdots \star$

- (ii) Within each level i :

The number of nodes is $\star \cdots \star$

The problem size is $\star \cdots \star$

If i is not the bottom level, the work done per subproblem is $\star \cdots \star$

If i is the bottom level, the work done per subproblem is $\star \cdots \star$

- (iii) The number of levels is $\star \cdots \star$, because $\star \cdots \star$.

- (iv) Summing over the levels, the total work is

$$T(N) = \star \cdots \star.$$

- (v) The big- Θ value of this sum is

$$\star \cdots \star.$$

Problem 4 collaboration and citations

I collaborated on this problem with the following people:

- $\star \cdots \star$

My answers use ideas from the following sources (other than the lecture notes and textbooks):

- $\star \cdots \star$

Problem 5. (*Exercise 3.13*) **NOTE: Do this problem or Problem 6, but not both.**

Prove or disprove the following theorem:

Theorem 1. *In every execution of the hospitals-propose Stable Matching algorithm, there is at most one hospital that makes offers to every doctor.*

(Note: we mean the algorithm described in the lecture notes—in each iteration, one hospital makes an offer to one doctor.) If the theorem is false, describe a counter-example: a particular instance, and an execution of the algorithm on that instance, in which at least two hospitals make offers to every doctor. If the theorem is true, give a long-form proof. In the proof, you may reuse without proof any observations, lemmas, or theorems from the lecture notes, but cite the observation, lemma, or theorem by number when you do, to let the reader know.

Problem 5 answer: the theorem is false ¹

The assertion is false. Consider the following problem instance:

$X : A \ B \ C$	$A : X \ Y \ Z$
$Y : B \ C \ A$	$B : Y \ Z \ X$
$Z : C \ A \ B$	$C : Z \ X \ Y$

(Please replace by your own instance and delete this comment.)

Now consider an execution of the algorithm on this instance that proceeds as follows:

★...★

The two hospitals ★...★ and ★...★ each make offers to every doctor.

Problem 5 answer: the theorem is true

Proof (long form).

1. Consider any execution of the algorithm.
2. ★...★
3. ★...★

□

Problem 5 collaboration and citations I collaborated on this problem with the following people:

- ★...★

My answers use ideas from the following sources (other than the lecture notes and textbooks):

- ★...★

¹MAKE SURE YOU READ THE COMMENTS in the file `problem_5_answer.tex`!

Problem 6. (*Exercise 3.11*) **NOTE: Do this problem or Problem 5, but not both.**

Prove or disprove the following theorem:

Theorem 2. *In any execution of the hospitals-propose Stable Matching algorithm there is at least one doctor that receives no offer before the final iteration.*

Note: we mean the algorithm described in the lecture notes—in each iteration, one hospital makes an offer to one doctor.) If the theorem is false, describe a counter-example: a particular instance, and an execution of the algorithm on that instance, in which every doctor receives an offer before the final iteration. If the theorem is true, give a long-form proof. In the proof, you may reuse without proof any observations, lemmas, or theorems from the lecture notes, but cite the observation, lemma, or theorem by number when you do, to let the reader know.

Problem 6 answer: the theorem is false ²

The assertion is false. Consider the following problem instance:

$X : A \ B \ C$	$A : X \ Y \ Z$
$Y : B \ C \ A$	$B : Y \ Z \ X$
$Z : C \ A \ B$	$C : Z \ X \ Y$

(Please replace by your own instance and delete this comment.)

Now consider an execution of the algorithm on this instance that proceeds as follows:

★...★

So every doctor receives an offer before the final iteration of the algorithm.

Problem 6 answer: the theorem is true

Proof (long form).

1. Consider any execution of the algorithm.

2. ★...★

3. ★...★

□

Problem 6 collaboration and citations I collaborated on this problem with the following people:

● ★...★

My answers use ideas from the following sources (other than the lecture notes and textbooks):

● ★...★

²MAKE SURE YOU READ THE COMMENTS in the file `problem_6_answer.tex`!