

Problem Set 7
Due: Tuesday, December 4

Reading: Handouts 29 (Greediness), 31 (Compression), 32 (Priority Queues), 34 (Amortization); *CLRS* Chapter 6 (Heapsort), Chapter 16.1–16.3 (Greediness), Chapter 17 (Amortized Analysis); *CLR* Chapter 7 (Heapsort), Chapter 17.1–17.3 (Greediness), Chapter 18 (Amortized Analysis)

Suggested Problems: *CLRS* 6.1-4, 6.1-5, 6.2-1, 6.3-1, 6.3-3, 6.4-1, 6.4-3, 6.5-6, 6.5-8; 12.1-2; 16.1-1, 16.1-4, 16.2-2, 16.2-5, 16.2-6, 16.3-1, 16.3-7; *CLR* 7.1-4, 7.1-5, 7.2-1, 7.3-1, 7.3-3, 7.4-1, 7.4-2, 7.5-3, 7.5-6; 13.1-2, 17.1-1, 17.1-3, 17.2-2, 17.2-3, 17.2-5, 17.2-6, 17.3-1, 17.3-7;

Required Problems: You should write up and turn in the solutions to the problems listed below: The points awarded per problem are given in brackets.

Problem 1 [10]: More Snake Oil Snake Oil Software’s CEO Kim Priest has just announced an amazing compression program that is guaranteed to compress every binary file by a factor of two, effectively doubling the size of your computer’s memory. (A length- n binary file is just a sequence of n 0s and 1s.) You have been asked to review this program.

- a. [5] Based on a simple counting argument, explain why the program cannot possibly work as advertised. *Hint:* How many length- n binary files are there?
- b. [3] Extend the reasoning in part a to explain why any compression technique that shrinks some files must increase the size others.
- c. [2] Given the problems raised by parts a and b, explain why compression techniques are still valuable in practice.

Problem 2 [15]: Professor Midas Do *CLRS* Exercise 16.2-4/*CLR* Exercise 17.2-4. Formally prove that your algorithm is optimal. As shown in Handout 29, *CLRS* Chapter 16, and *CLR* Chapter 17 there are two parts to a proof that a greedy algorithm is optimal:

- a. [7] Show that the algorithm exhibits the **greedy choice property**: an optimal solution can begin with the greedy choice. The strategy for proving this is as follows. Assume you are given an optimal solution S . Then show that (1) S begins with the greedy choice or (2) if S does not begin with the greedy choice, then replacing whatever it begins with by the greedy choice leads to another optimal solution. (*Note:* there may be many optimal solutions.)
- b. [8] Show that the problem exhibits the **optimal substructure property**: the optimal solution for the whole problem can be derived from the optimal solution for the parts. This is usually shown as follows. Assume that you are given an optimal solution S for the whole problem and show that you can extract the optimal solution S_i for subproblem i from the optimal solution to the whole problem. The proof proceeds by contradiction: assume that there is a better solution S'_i to subproblem i and show that you could construct a better whole solution S' using S'_i . But that contradicts your original assumption about the optimality of S . So it must be the case that your other assumption is incorrect: S'_i cannot be better than S_i . That is, each S_i is optimal.

Problem 3 [15]: When a Queue Made of Two Lists Seems So New To You – That’s Amortization!

Below is an implementation of a mutable first-in-first-out (FIFO) queue that represents the queue as a pair of two lists, labeled `front` and `back`:

```
Empty-Queue()
  return new record {front = EmptyList(), back = EmptyList()}

Empty-Queue?(Q)
  return EmptyList?(front[Q]) and EmptyList?(back[Q])

Enq(x, Q)
  back[Q] ← Prepend(x, back[Q])

Deq(Q)
  if EmptyList?(front[Q]) then
    front([Q]) ← Reverse(back[Q]) ▷ Assume a linear-time Reverse
    back([Q]) ← EmptyList()
  if EmptyList?(front[Q]) then
    error "empty queue"
  else
    let h = Head(front[Q])
    in front[Q] ← Tail(front[Q])
    return h
```

a. [5] Show the state of the queue (i.e., the `front` and `back` lists) after each of the operations in the following sequence:

```
q ← Empty-Queue()
Enq(1,q)
Enq(2,q)
Enq(3,q)
Deq(q)
Enq(4,q)
Enq(5,q)
Deq(q)
Deq(q)
Deq(q)
Enq(6,q)
Deq(q)
Deq(q)
```

b. [5] Using the banker’s (accounting) method, argue that the amortized cost of `Enq` and `Deq` are each $O(1)$.

c. [5] Using the physicist’s (potential) method, argue that the amortized cost of `Enq` and `Deq` are each $O(1)$.

Problem 4 [25]: A Heap o' Heaps

- a. [5] Draw as trees the sequence of *complete* heaps H_0, H_1, \dots, H_{17} that results from inserting the following letters into the empty heap H_0 :

T H E Q U I C K B R O W N Y A M S

For example, H_1 should be the result of inserting T into the empty heap H_0 ; H_2 should be the result of inserting H into H_1 ; and so on. Assume that letters appearing earlier in the alphabet have a *higher* priority than those appearing later. For example, the root node of H_{17} should be A. You need only show the final tree that results from each insertion; you need not show the individual "bubble up" steps that lead to the final tree.

- b. [2] Show how the complete heap H_{17} would be represented as an array.

- c. [4] Draw as trees the sequence of *complete* heaps $H_{18}, H_{19}, H_{20}, H_{21}$ that result from extracting and deleting the *four* highest-priority items of H_{17} in order of priority (from high to low). You need only show the final tree that results from each deletion; you need not show the individual "bubble down" steps that lead to the final tree.

- d. [4] Draw as trees the sequence of *leftist* heaps LH_0, LH_1, \dots, LH_8 that results from inserting the following letters into the empty heap LH_0 :

T H E Q U I C K

- e. [4] Draw as trees the sequence of *leftist* heaps $LH_{20}, LH_{21}, \dots, LH_{29}$ that results from inserting the following letters into the empty heap LH_{20} :

B R O W N Y A M S

You need not show the intermediate results of individual merge steps.

- f. [2] Draw as a tree the *leftist* heap LH_0 that results from merging the two heaps LH_8 and LH_9 .

- g. [4] Draw as trees the sequence of *leftist* heaps LH_1, LH_2, LH_3, LH_4 that results from extracting and deleting the *four* highest-priority items of LH_0 in order of priority (from high to low). You need only show the final tree that results from each deletion; you need not show the individual merge steps that lead to the final tree.

Problem 5 [35]: Grn eggs nd Hm Here we consider compressing the first few lines of Dr. Seuss's timeless classic, *Green Eggs and Ham*:

```
i_am_sam
i_am_sam
sam_i_am
that_sam-i-am
that_sam-i-am
i_do_not_like_that_sam-i-am
do_you_like_green_eggs_and_ham
i_do_not_like_them_sam-i-am
i_do_not_like_green_eggs_and_ham
```

For simplicity, all letters are lower case, and all punctuation has been omitted except for dashes (written -) spaces (written $_$), and carriage returns (written \downarrow).

We will refer to the above 9 lines as “the poem”. We will consider the number of bits needed to represent the poem under various compression schemes.

Here are some useful facts about the poem:

- There are 19 distinct characters in the poem (counting dash, space, and carriage return).
- Character frequency table:

a	d	e	g	h	i	k	l	m	n	o	r	s	t	u	y	-	$_$	\downarrow
21	6	11	6	6	14	4	4	17	7	8	2	9	10	1	1	8	31	8

- Total number of characters: 174
- There are 16 distinct words in the poem (counting individual dashes, spaces and carriage returns as words).
- Word frequency table:

am	and	do	eggs	green	ham	like	not	i	sam	that	them	you	-	$_$	\downarrow
7	2	4	2	2	2	4	3	10	7	3	1	1	8	31	8

- Total number of words: 95

a. [1]

What is the number of bits that it would take to represent the poem if each character is represented by an 8-bit ASCII character?

b. [4]

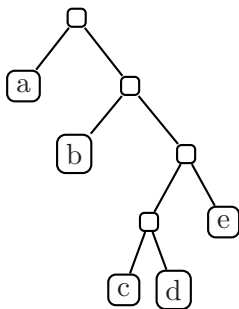
1. What is the smallest number of bits k that it would take to represent one character of the poem in a uniform-length binary code?
2. What is the number of bits it would take to represent a table mapping 7-bit ASCII characters to their equivalent in the binary code? Assume that the table has the following format:
 - (a) 3 bits to represent k .
 - (b) 7 bits to represent the number n of table entries.
 - (c) n table entries, where each entry is the k -bit encoding of a character followed by its 7-bit ASCII character representation.
3. How many bits total would it take to represent the table followed by the k -bit representations of the contents of the poem?

c. [10]

1. Using the character frequency table, derive a Huffman code tree for this example, where each symbol is a character.
2. Make a table showing the codes for the 19 individual characters.
3. Ignoring the space for encoding the tree/table, how many bits would it take to encode the poem using your table?

4. Some encoding of the tree/table must be sent along with the encoded message to allow it to be decoded. There are many ways to encode the tree. How many bits would be required to represent the tree using the following encoding?
- 16 bits indicating the number of leaves in the tree.
 - A sequence of 8-bit bytes representing a post-order traversal of the tree, where:
 - A byte with leading bit 0 stands for a leaf denoting a character represented by the remaining 7 bits.
 - A byte with leading bit 1 stands for n “node” constructors, where n is the number represented by the remaining 7 bits.

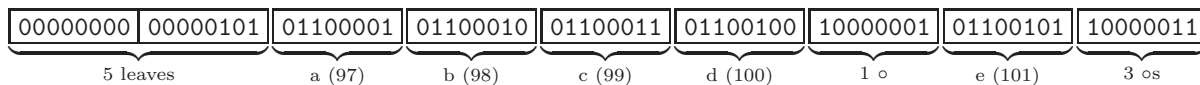
For example, the tree



has the post-order representation

a b c d ◦ e ◦ ◦ ◦

(where ◦ denotes the post-order node constructor) and yields a table with the following bytes:



5. What is the total number of bits (table + contents) needed to represent the poem using this encoding?

d. [10] Repeat part c, except use the words of the poem (rather than its characters) as symbols in the Huffman code. Assume that the Huffman tree is represented as in part c except that the leaves (which are now strings, not just characters) are represented as (1) an 8-bit byte with leading bit 0 and 7 bits encoding the number of characters n in the string and (2) $7n$ bits representing the n 7-bit ASCII encodings of the characters in the string.

e. [10] Use *one* of (1) the LZ77 encoding or (2) the LZ78 encoding (you choose!) described in Handout #31 to encode the poem, and indicate the total number of bits taken by your representation.

- The LZ77 encoding should consist of triples of the form (1) back (2) length and (3) character. Assume that the encoding begins with 8 bits indicating the maximum number of bits b needed to encode a single back/length number. Each triple can then be represented by $2 \cdot b + 7$ bits.
- The LZ78 encoding should consist of pairs of the form (1) codeword index and (2) character. Assume that the encoding begins with 8 bits indicating the maximum number of bits b needed to encode a single codeword (index). Each pair can then be represented by $b + 7$ bits.

Extra Credit 1 [40]: More Compression

The following problems extend Problem 5:

- f. [15] Revise part e of Problem 5 to use Huffman coding to compress the back/length numbers and characters. Remember that it is necessary to transmit a representation of the Huffman tree in this case. Does this approach decrease or increase the size of your representation?
- g. [25] Learn about the LZW compression algorithm. Use it to encode the poem. Using reasonable assumptions about the bit representation of components, how many bits does it take to represent the poem using LZW?

Extra Credit 2 [30]: Get Rich Quick with Huffman Coding

The following game was invented by Mads Tofte¹. In his own words:

[There are] two players, A and B. A rolls two dice, thus producing a sum between 2 and 12. B has to guess the sum. B can ask A any question he likes, but A must answer only yes or no. B pays one unit for each question he poses, until he knows the correct answer. Then the roles are switched. This cycle is repeated 10 times (to give statistics a chance) and the points are added up. The one who has asked the fewest questions altogether wins. The optimal strategy turns out to require roughly 33 questions for 10 rounds. A reasonable opponent uses around 36 questions, in my experience, which makes the opponent want to try again. Excellent pub game (that is what I invented it for), as long as one can remember the Huffman tree (and re-arrangements of it) by heart.

Construct the Huffman tree implied by the problem, and show why it offers the competitive advantage mentioned in Mads's description.

Extra Credit 3 [30]

Do *CLRS* Exercise 16.1-3/*CLR* Exercise 17.1-2. Prove that your algorithm is optimal. *Warning:* many "obvious" algorithms are not optimal!

Extra Credit 4 [30]

Do *CLRS* Problem 16-1/*CLR* Problem 17-1. (The *CLRS* version has an extra part (d) that you can do for even more credit.)

Extra Credit 5 [30]

The implementation of complete heaps in *CLRS* Ch. 6/*CLR* Ch. 7/Handout #32 is for a mutable specification of priority queues. Develop an implementation of complete heaps for an immutable specification of priority queues. Your implementation should be purely functional (no side effects!) and should represent the complete heap as a tree rather than an array.

¹Known for his work on the semantics of the ML programming language, Mads Tofte is now the director of the IT University of Copenhagen. He is also my "twin": people have mistaken Mads Tofte for me (and vice versa) at conferences. To see why, check out the picture at <http://www.diku.dk/users/tofte/>.

Problem Set Header Page
Please make this the first page of your hardcopy submission.

CS231 Problem Set 7

Due Tuesday, December 4

Name:

Date & Time Submitted:

Collaborators (*anyone you worked with on the problem set*):

*In the **Time** column, please estimate the time you spend on the parts of this problem set. Please try to be as accurate as possible; this information will help me design future problem sets. I will fill out the **Score** column when grading your problem set.*

Part	Time	Score
General Reading		
Problem 1 [10]		
Problem 2 [15]		
Problem 3 [15]		
Problem 4 [25]		
Problem 5 [35]		
Extra Credit [135]		
Total		