CS320 Algorithms, Fall 2017 Test 1 – Version A – KEY
Friday, October 6, 2017, 50 minutes maximum

<table>
<thead>
<tr>
<th>Category</th>
<th>Total Possible Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group I</strong>, questions 1-2 (you must answer at least 1 of 1-2; the other is optional to make 100 points)</td>
<td>Total for category: 40 points</td>
<td></td>
</tr>
<tr>
<td><strong>Group II</strong>, questions 3-5 (you must answer at least 1 of 3-5; rest are optional to make 100 points)</td>
<td>Total for category: 20 points</td>
<td></td>
</tr>
<tr>
<td><strong>Group III</strong>, questions 6-10 (you must answer at least 1 of 8-10; rest are optional to make 100 points)</td>
<td>Total for category: 25 points</td>
<td></td>
</tr>
<tr>
<td><strong>Group IV</strong>, questions 11-13 (you must answer at least 1 of 11-13; rest are optional to make 100 points)</td>
<td>Total for category: 15 points</td>
<td></td>
</tr>
<tr>
<td><strong>Group V</strong>, questions 14-16 (you must answer at least 1 of 14-16; rest are optional to make 100 points)</td>
<td>Total for category: 27 points</td>
<td></td>
</tr>
<tr>
<td><strong>Group VI</strong>, questions 17-26 (optional to make 100 points)</td>
<td>Total for category: 38 points</td>
<td></td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>Total points on exam: <strong>165 points</strong></td>
<td></td>
</tr>
</tbody>
</table>

You must answer questions to make at least 100 points. If you only answer 1 question out of Groups I-V, you will not have 100 points; you must answer additional questions to get to 100 points.

You may answer more questions if you want to increase your chance of making a better score on this part of Test 1, but you will not receive credit for more than 100 points total on Test 1.

- This exam has **26 questions**.
- Be specific and clear in your answers.
- There are several questions that have room for up to 2 sentences of justification. We will **not consider more than the first 2 sentences** when we are grading.

This exam is closed book, closed notes EXCEPT for a single 8.5”x11” page with your choice of notes on it. Handwritten, typed, double-sided OK. You **MUST turn in this notes sheet with your exam**.

Name: ____________________________________________
GROUP I QUESTIONS 1 – 2
(You must answer at least 1 of questions 1-2 in this section. You may answer other questions for additional points.)

1. [20 pt] Boats/Ships and People
This problem is an exploration of the Gale-Shapley algorithm and potential adaptations to new situations. For the problem, consider the 2 sets of participants to be a set of boats and a set of groups. These are described below.
There are 3 different boats/ships with different sizes and activities they support. The boat preferences are listed in curly braces, with preferred activities at the beginning of the list. (Some boats have fewer items in their preference lists because the boat cannot support other activities.)
   A. 5-person sailboat: {fishing, photography}
   B. 5-passenger diving boat: {diving}
   C. 15-person Catamaran: {snorkeling, photography}

There are 5 groups of 5 people each with different preferences. The group preferences are listed in curly braces, with preferred activities at the beginning of the list. (Some groups have fewer items in their preference lists because members of the group refuse to engage in other activities.)
   1. Group 1: {diving, fishing, snorkeling}
   2. Group 2: {photography}  
   3. Group 3: {snorkeling, diving, fishing}
   4. Group 4: {fishing, photography}
   5. Group 5: {photography, fishing, diving, snorkeling}

Questions:
   a. [1] Can we use recursion to solve this problem?
      ___ Yes    ___ X. No
      Optional: In 2 sentences or less, explain your answer (e.g. why or why not)

   b. [2] Can we use G-S directly for this problem?
      ___ Yes    ___ X. No
      Optional: In 2 sentences or less, explain your answer (e.g. why or why not)

   c. [3] Are there adaptations to the problem we can make to G-S to use it with this problem?
      ___ X. Yes    ___ No
      If so, what are they? (Choose all that apply)
      □ X convert the group activity desires to preference lists of boats, adding boats that don’t have these activities at the end
      □ X convert the boat activities into preference lists of groups, ranked by the position of the activities the boat provides in the group’s list of desired activities, and adding groups that don’t want any of the activities provided by the boat at the end
      □ only create a boat’s preference list to include the groups who want to do the activities it supports
X create 3 of the ‘C’ boat so that each copy provides space for 5 people
X make sure there are the same number of boats and groups of people
X after the algorithm has created a matching, remove any matching of boats/groups that are not allowed (e.g. the diving boat having the group that wants to only do photography), even if this means not all boats or groups are matched
X give priority to the groups who only want to do 1 activity; match them first and never break the match
other
Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer

___________________
___________________

d. [3] What would the measure of progress be? (Choose all that apply)
   
   X the number of boats that are matched
   X the number of groups that get their highest preference activity
   X the number of “proposals” that are made
   other
Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer

___________________


e. [3] If we can use some form of G-S, what are the proposers and acceptors? What would a stable matching look like? First part: 1 pt, multiple choice: 2 pts for correct, 1 pt for both correct and additional answers
   “proposers” ________ really doesn’t matter – groups is fine
   “acceptors” _______ really doesn’t matter – boats is fine

A stable matching would be when: (Choose all that apply)
   X there is no (g1, b1) and (g2, b2) where g1 prefers b2 over b1 and b2 prefers g2 over g1
   X all boats are matched to a group, even if the group doesn’t want to do the activities provided by the boat
   X there is no (g1, b1) and (g2, b2) where g1 prefers b2 over b1 and g2 prefers g1 over g2
   X all groups and boats are matched
   X none of these; it cannot return a stable matching
Optional: In 2 sentences or less, explain your answer (e.g. why or why not)

___________________


f. [3] If we make adaptations will the algorithm still return a perfect matching? (Choose all that apply)
   X the algorithm will never return a perfect matching
   X only if groups are forced to go on boats that may not provide the activities they want
   X the algorithm will always return a perfect matching
   X the algorithm will sometimes return a perfect matching, but we cannot state any particular conditions when this might occur
   X only if boats are forced to provide the activities people want


g. [2] How would we measure the size of the problem? (Choose one)
the number of passengers/people a boat can hold
- the total number of people
- the number of boats
- X a number representing the number of boats (after we replicate boats as needed to get the same number as the number of groups; this is the same as the number of groups
- the number of boats + the number of groups
- other

Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer

______________________________________________________________________
______________________________________________________________________

h. [3] How do each of the above questions (a, b, c, d, e, f, and g) fit into the approach we said we’d use to study algorithms (first day of class)? Note that the question may not completely fulfill the requirements of the approach step, but it does provide some of the work. Write the letter associated with the question in the blank. If a question doesn’t fit one of these steps in the approach then leave it out. If no question fits one of the steps in the approach then write N/A or none or put a dash (‘---’) beside the step. Under 5 correct: 1pt, >5 2 pts, all correct: 3pts

1. Formulate problem: __ c, d __
2. Design algorithm: __ c, d, first part of e __
3. Prove correctness: __ e, f __
4. Analyze complexity: __ g __
5. Implement: __ N/A __

2. [20 pt] Convex Hull:
The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior. Imagine a bunch of pegs standing in a group, if you stretched a rubber band around them, the shape of the rubber band is the convex hull of the points/pegs.

Finding the convex hull for a given set of points can be solved recursively. It involves partitioning the points. (Hint: think about first ordering the points, maybe by their x coordinate values.)

a. [3] What is one way you can think of to divide the points so that we can use recursion for this problem? (Choose one.)
- X separate them into 2 parts, divided by some vertical line that is about the midpoint of all the x coordinate values
- separate them into clusters (e.g., (p6, p11) and (p12, p2, p4, p5), (p0), (p10), (p3))
Separate them into points on the border and points inside (e.g., (p₀, p₁₂, p₁₀, p₃, p₁) and (p₂, p₄, p₁₁))

Other

Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer

b. [2] How many subproblems are there? 2

c. [2] Roughly how large are the subproblems? (Choose one.)

- 1 point and all but 1 of the points
- As big as the biggest cluster
- X about half the points
- About a third of the points and the other 2/3 of the points
- Other

Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer

d. [2] In the recurrence equation, what would represent the work done for the subproblems? (Choose one.)

- T(n/3) + T(2n/3)
- T(n/2)
- T(n/2), where z is the number of clusters
- X 2T(n/2)
- 2T(n/3)
- Other

Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer

e. [2] If we can combine the results of the subproblems into a convex hull for the original problem in O(n) complexity (n original points), then what is the recurrence equation for the work done to solve this problem? (Choose one.)

- X T(n) = 2T(n/2) + O(n)
- T(n) = 2T(n/2) − O(n)
- T(n) = T(n/3) + T(2n/3)
- T(n) = 3T(n/3) + O(log n)
- Other

Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer

f. [3] What is the big-O solution for this recurrence equation? (Choose one.)

- We cannot determine this from the Master Theorem
- O(n²)
- O(2/3 log n)
- O(n log n)
- X O(n log n)
g. [3] What strategy/technique have you used to think about the solution to this problem? (Choose one.)
   - divide and conquer
   - recursion
   - greedy
   - dynamic programming
   - other

Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer

______________________________________________________________________
______________________________________________________________________

h. [3] How do the steps in the questions (a, b, c, d, e, f, and g) above relate to the steps of the strategy/technique you chose for the previous question? Note that the question may not completely fulfill the requirements of the approach step, but it does provide some of the work. Write the letter associated with the question in the blank. If a question doesn’t fit one of these steps in the approach then leave it out. If no question fits one of the steps then write N/A or none or put a dash (‘--’) beside the step. (Only fill in the steps for the technique you chose in part g of this question.)

Divide and Conquer:
- divide the problem into roughly equal sized subproblems
- solve the subproblems recursively
- combine the subproblem solutions into a complete solution

Dynamic Programming:
- define subproblems(characterize structure and show it is optimal
- recursively define the optimum
- compute the optimum solutions to subproblems (usually bottom-up)
- construct overall optimum from stored data

Greedy:
- determine optimal subproblem structure
- develop recursive solution
- show that if we make the greedy choice we are left with only 1 subproblem
- prove it is always safe to make the greedy choice
- develop a recursive algorithm for the greedy strategy
- convert the recursive algorithm to an iterative algorithm

Additional:
- define recursion
- define base case

GROUP II QUESTIONS 3-5
[20 pt] Logic:
Assume we have the following statement: (we have overlapping subproblems) \( \Rightarrow \) (we can solve this problem using dynamic programming)

3. [5] What can you say about the statement if you are told that (we have overlapping subproblems) is false?
   - [ ] X the statement is true
   - [ ] the statement is false
   - [ ] we cannot say whether the statement is true or false

4. [5] What can you say about the statement if you are told that (we can solve this problem using dynamic programming) is false?
   - [ ] the statement is true
   - [ ] the statement is false
   - [ ] X we cannot say whether the statement is true or false

5. [10] Given the following set of statements about a proof, choose which statements are part of a proof regarding shortest paths, and put them in the correct order.

   1. The number of edges in p equals the number of edges in p \(_1\) plus the number of edges in p \(_2\).
   2. We can splice together any shortest path \( u \overset{p_1}{\rightarrow} w \) and any shortest path \( w \overset{p_3}{\rightarrow} v \).
   3. This contradicts our statement that p is a shortest path from u to v.
   4. Assume vertices u and v where u \( \neq \) v. Then any path p from u to v must contain an intermediate vertex w.
   5. If there were another path, \( p_1' \), from u to w with fewer edges than \( p_1 \), then we can cut out \( p_1 \) and paste in \( p_1' \) to produce a path \( u \overset{p_1'}{\rightarrow} w \overset{p_3}{\rightarrow} v \) with fewer edges than p.
   6. Because we use vertices u and w in one subproblem solution, we cannot use them in another.
   7. Therefore we can decompose the path \( u \overset{p}{\rightarrow} v \) into subpaths \( u \overset{p_1}{\rightarrow} w \overset{p_3}{\rightarrow} v \).
   8. If p is a shortest path from u to v, then \( p_1 \) must be a shortest path from u to w.

   - [ ] 1, 2, 3, 4, 5, 6, 7, 8
   - [ ] 3, 2, 6, 7, 8, 3
   - [ ] X 4, 7, 1, 8, 5, 3
   - [ ] 1, 2, 5, 6, 8, 3

GROUP III QUESTIONS 6-10
(You must answer at least 1 of questions 6-10 in this section. You may answer other questions for additional points.)

[25 pt] Bounding:
Using the picture below, fill in the blanks with the proper bounds for the functions \( f_1(n) \) and \( f_2(n) \).
Fill in the blank with the proper bound using these letters: \( O, \Omega, \Theta \)
6. If \( g(n) = n^2 \), then \( f_1(n) = \_ \_ \_ \Omega \_ \_ \_ g(n) \), if \( g(n) = \log n \) then \( f_1(n) = \_ \_ \_ \Omega \_ \_ \_ g(n) \), and if \( g(n) = n \) then \( f_2(n) = \_ \_ \_ \Omega \_ \_ \_ g(n) \)

7. What is a function that has a \( \Theta() \) relation with \( f_1(n) \)? (Choose one.)
   - \( n \log n \)
   - \( \sqrt{n} \)
   - \( 1 \)
   - \( X f_1(n) \)
   - None of the above

8. If \( f(n) = n^2 \), and \( g(n) = f_1(n) \), then \( f(n) = \_ \_ \_ \Omega \_ \_ \_ g(n) \), if \( g(n) = 2^n \) then \( f(n) = \_ \_ \_ \Omega \_ \_ \_ g(n) \), if \( g(n) = n! \) then \( f(n) = \_ \_ \_ \Omega \_ \_ \_ g(n) \), and if \( g(n) = n \) then \( f(n) = \_ \_ \_ \Omega \_ \_ \_ g(n) \)

Use the ratio of \( f(n)/g(n) \) to determine the proper relation between the functions. Fill in the blanks with the ratio and then the proper bound using the letters: \( \Theta, o, \) or \( \omega \). The following facts may be useful:

- \( x^y/x^z = x^{y-z} \) and \( x^y x^z = x^{y+z} \)
- \( c^n/a^n = (c/a)^n \)

For 9,10: ratio correct: 3pts, relation correct: 2pts
9. If \( f(n) = (\log_2 n)^2 \) and \( g(n) = n^{0.001 \log_2 n} \) then \( f(n)/g(n) = \frac{\log_2 n}{n^{0.001}} \), which means that \( f(n) = \_ \_ \_ o \_ \_ \_ (g(n)) \)

10. If \( f(n) = n! \) and \( g(n) = (n + 1)! \), then \( f(n)/g(n) = \frac{1}{(n+1)} \), which means that \( f(n) = \_ \_ \_ o \_ \_ \_ (g(n)) \)

**GROUP IV QUESTIONS 11-13**
(You must answer at least 1 of questions 13-13 in this section. You may answer other questions for additional points.)
[15 pt] Master Theorem:

11. [5] When we built a min-heap using an array to represent the binary tree, we inserted an item at the end of the array and then called a procedure to restore the heap invariant. What was the heap invariant? (Choose one.)
   - □ the heap has to be a complete binary tree
   - X all children of a node must have larger values than their parent node
   - □ a node must have a larger value than its children
   - □ other

Optional: Especially if you answered ‘other’, in 2 sentences or less, explain your answer
______________________________________________________________________
______________________________________________________________________

12. [5] We can sort a list of items by recursively sorting the 2 halves and then merging the resulting sorted halves together. The recurrence equation for work done by this procedure is: (Choose one.)
   - □ T(n) = nT(1) + log n
   - X T(n) = 2T(n/2) + n
   - □ other

Optional: Especially if you answered ‘other’, in 2 sentences or less, explain your answer
______________________________________________________________________
______________________________________________________________________

13. [5] What is the solution to this recurrence equation? n log n
GROUP V QUESTIONS 14-16
(You must answer at least 1 of questions 14-16 in this section. You may answer other questions for additional points.)

[27 pt] Strategies/Techniques:
Partial credit given if answer to 14 was dynamic programming and answer to 15 was greedy
You are in a technical interview for a company you’d really like to join, and they give you the dice throwing problem: given n dice each with m faces numbered 1 to m, how many ways are there to get a particular value X where X is the total sum of the values on each of the dice faces when the dice are thrown. (Hint: This is similar to several of the problems we studied. Think about the total, and how it changes as you look at each dice, and how the number of dice changes as you work through the problem.)

14. [10] What strategy/technique could you use to solve this problem? (Choose one.)
   - searching
   - divide and conquer
   - greedy
   - recursion
   - dynamic programming
   - sorting
   - other

Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer
______________________________________________________________________
______________________________________________________________________

15. [5] If you think it would be useful to look at one or more other strategies/techniques, which would you explore? (Choose all that apply.)
   - searching
   - divide and conquer
   - greedy
   - recursion
   - dynamic programming
   - sorting
   - nothing else needs to be explored
   - other

Optional: Especially if you chose ‘other’, in 2 sentences or less, explain your answer
______________________________________________________________________

16. [12] How do divide and conquer, dynamic programming, and greedy strategy relate? Fill in the blanks below using ONLY the words in the following word bank. Words can be used more than once if needed, and not all words in the bank are needed.

   Word bank: [general, sorting, searching, recursion, subproblems, resources, overlapping, optimal, greedy, dependent, Gale-Shapley]

   Divide and Conquer is very ___general___ and useful whenever ___recursion___ can be used to solve ___subproblems___ that don’t share ___resources_____. Dynamic programming is used when there are ___overlapping___ ___subproblems___ and ___optimal___ substructure.
Greedy is used if the answer to a subproblem is not dependent on the answers to later subproblems.

GROUP VI QUESTIONS 17-26
(You may answer questions in this group for additional points.)

[15 pt] Fun problem:
If tree incorrect, but subsequent questions correct, then partial credit given.
Recall the Huffman code algorithm: sort characters by their frequencies in ascending order (minimum frequency character first) to create a “sorted list”, then take the 2 minimum items and create a node that has the smallest as its left child and the larger as its right child. Use the sum of the 2 frequencies as the name of this node, and label the left side 0 and the right side 1. Then put this little tree back into the “sorted list” in its proper position. Now take the next 2 minimum items in this “list” and create a node that has the smallest as its left child and the larger as its right child. Use the sum of these as the name of the new node, and label the left and right sides 0 and 1, respectively. Repeat until there is just 1 node left in the “list”.
17. [5] Draw a Huffman code tree for these expected relative frequencies {a:6, b:12, c:3, d:10, e:7}.
Answer:

18. [3] Complete this table of the codes for each character.

<table>
<thead>
<tr>
<th>char</th>
<th>code</th>
<th>#bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Answer:

<table>
<thead>
<tr>
<th>char</th>
<th>code</th>
<th>#bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>010</td>
<td>3</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>00</td>
<td>2</td>
</tr>
</tbody>
</table>

20. [2] How many bits does it take to encode the string? Compare to a UTF-16 code that takes 2 bytes/character.

Bits to encode “cabbed” using Huffman coding: ___14___ = ___2___ bytes
Bytes to encode “cabbed” using UTF-16 coding: ___12___ bytes

[15 pt] Python:
21. [4] Which data structure(s) support fast (constant time) membership testing? (Choose all that apply.)
   - X dictionary
   - X set
   - list
   - string

22. [3] Use the list constructor to take this string ‘today’ and produce this list ['t', 'o', 'd', 'a', 'y']. ONLY write the constructor, don’t include any variable assignments for the list.
   ___list('today')___

23. [3] In Python3, what is the result of the last expression? (Choose one.)
   - y = list()
   - 0 if not y else y[-1]
   - index out of bounds exception raised
   - X 0
   - -1

24. [5] [x**2 for x in range(4) if x % 2 == 0] is an example of a ___list comprehension__ and produces [___0, 4_____________].

[8 pt] Gale-Shapley:
Assume we ran the Gale-Shapley algorithm with the following preferences:
   w1: {m1, m2}  m1: {w1, w2}
   w2: {m2, m1}  m2: {w2, w1}
Assume the w’s do the “proposing” and that the algorithm produces the following matching:
   (w1, m2), (w2, m1)
25. [5] Is this matching stable? _____Yes  X No
   Optional: In 2 sentences or less, explain your answer
   _______w1 prefers m1 and m1 prefers w1

26. [3] Is it perfect? X Yes  _______No
   Optional: In 2 sentences or less, explain your answer
   ____________________________________________________________