CS351 Sample Final Exam Questions

These questions are pulled from several previous finals. Your actual final will include fewer questions and may include different topics since we did not cover the same topics in previous classes. However, these problems should give you an idea of what type of questions you will be asked.

Short Answer

Give a short answer (i.e. 1 to 3 sentences) for the following questions:

a) Give two reasons why it may be preferable to implement a $O(n^2)$ algorithm over an O(n) algorithm that solves the same problem.

b) In the Build-Heap algorithm, the O(lgn) Heapify routine is invoked n times. This seems to indicate that Build-Heap is O(nlgn). Describe why the better bound for Build-Heap is really O(n).

c) Given a set of n numbers, we wish to find the i largest in sorted order. One solution is to use the expected running time of the Selection algorithm to find the ith largest number, partition around that number, and sort the i largest numbers (say, using Merge Sort). What is the running time in terms of n and i? Explain your answer.

d) Describe an O(n) algorithm to union two binary heaps together (not binomial heaps).

e) The algorithm below, "Fibsort", sorts an input array of numbers. Its parameters are the array and the starting/ending indices of the range within the array that we would like to sort.

Write a recurrence relation describing the runtime of the algorithm. Do not solve the recurrence.

```
Fibsort(long A[], int Start, int End)
    If End <= Start Then
       Return
    If A[Start] > A[End] Then
    Swap A[Start], A[End]
    Fibsort(A, Start + 1, End - 1)
    If A[Start] > A[Start + 1] Then
    Swap A[Start], A[Start+1]
    Fibsort(A, Start + 1, End)
```

Sample call for A[] = { 5, 4, 99, 21, 44 } :

```
Fibsort(A, 1, 5);
```

NP-Completeness

1) Show that the Subset Sum problem is NP-Complete.

Subset Sum: Given a set S of n numbers and a target number C, you want to know what subset of S sums to exactly C.

Assume that you know the Set Partition problem to be NP-Complete. In the Set Partition problem, you are given a set S of n numbers and you want to know if the set can be divided into two sets so that the sum of all elements in the first set is equal to the sum of all elements in the second set.

Note that this is backwards from the homework problem. In the homework, you were given that Subset Sum is NP-Complete and showed that Set Partition is NP-Complete. Here, you are given that Set Partition is NP-Complete and must show that Subset Sum is NP-Complete.

2) Show that the decision version of the longest path problem is NP-Complete.

The longest path problem is: Given an undirected graph G(V,E) of n vertices and a positive integer k, does G contain a simple cycle containing at least k vertices?

A simple cycle is a cycle that doesn't repeat any edges or vertices.

Show that the longest path problem is NP-Complete. You can use the knowledge that the Hamilton Circuit problem is NP-Complete.

Dynamic Programming

1) Planning a Party

Professor McKenzie is consulting for the president of A-B corporation, which is planning a company party. The company has a hierarchical structure; that is, the supervisor relation forms a tree rooted at the president. A person has only one supervisor.

The personnel office has ranked each employee with a conviviality rating, C[employee], which indicates how fun that person will be at the party. C[employee] is a real number. In order to maximize the party atmosphere, the president does not want both an employee and his or her immediate supervisor to attend.

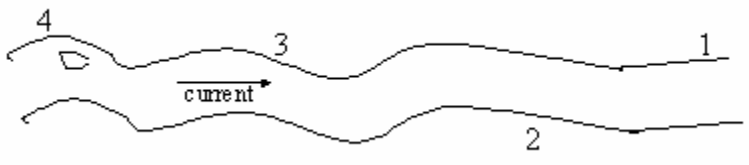
a) Describe an algorithm to make up the guest list. The goal should be to maximize the sum of the conviviality ratings of the guests.

b) How can the professor ensure that the president gets invited to his own party?

2) Trading Posts

There are n trading posts along a river numbered n, n -1, … 3, 2, 1.. At any of the posts you can rent a canoe to be returned at any other post downstream. (It is impossible to paddle against the river since the water is moving too quickly). For each possible departure point i and each possible arrival point j (< i), the cost of a rental from i to j is known. It is C[i,j]. However, it can happen that the cost of renting from i to j is higher than the total costs of a series of shorter rentals. In this case you can return the first canoe at some post k between i and j and continue your journey in a second (and, maybe, third, fourth ...) canoe. There is no extra charge for changing canoes in this way.

Give a dynamic programming algorithm to determine the minimum cost of a trip by canoe from each possible departure point i to each possible arrival point j. Analyze the running time of your algorithm in terms of n. For your dynamic programming solution, focus on computing the minimum cost of a trip from trading post n to trading post 1, using up to each intermediate trading post.



3. Coin Changer

In the year 2012, the US Congress passes an amendment allowing foreign-born citizens to become president and Arnold Schwarzenegger is elected president of the United States. To commemorate his achievement a new 20 cent coin is created in his honor:

While a great honor to president Schwarzenegger, the new coin means that we can no longer use the greedy strategy to return change with the fewest number of coins. The greedy strategy is to simply dole out the largest of each coin possible. For example, to give 40 cents in change, the greedy strategy would return 1 quarter, 1 dime, and 1 nickel. However, we could instead dole out 2 Schwarzeneggers which would be the fewest number of coins possible.

Devise a dynamic programming formulation that determines the fewest number of coins to return as change for coin denominations D_0, D_1, D_2, D_3 and D_4. D_0 is always one cent. D_1 through D_4 could be arbitrary amounts (e.g. perhaps D_1=5, D_2=10, D_3=20, and D_4=25 cents).

To get you started, the minimum number of coins to return is denoted by C[N], where N is the amount of change:

C[N] = 0 if N = 0
; if no change to return, minimum number of
; coins is zero

You complete the dynamic programming formulation for N > 0:

C[N] = if N > 0

Recurrence Relations and Algorithm Design

1) Min/Max Finding

Given an array A of n real numbers, a(1), a(2) … a(n) , the algorithm below is to find the maximum and minimum of these n numbers. It is invoked via:
FindMinMax(A,1,n,min,max). It works as follows:

```
FindMinMax(A, lowindex, highindex, &min, &max)
    If lowindex=highindex then
        Max=A(lowindex)
        Min=A(lowindex)
        Return
    mid ← ⌊(lowindex + highindex)/2⌋
    FindMinMax(A, lowindex, mid, min1, max1)
    FindMinMax(A, mid+1, highindex, min2, max2)
    Min ← Minimum(min1,min2)
    Max ← Maximum(max1,max2)
```

Let T(n) be the number of comparisons needed using this algorithm. Find a recursive equation for T(n) and solve it using any method you like.

2) Nuts and Bolts

You have a bucket that contains exactly *n* nuts and *n* bolts. For each nut in the bucket, there is exactly one bolt in the bucket that fits it, and vice versa. You want to find the nut that fits each bolt. More formally, the nuts may be defined as N[1..n] and the bolts as B[1..n].

The only technique that you may use to assess the relative sizes of the nuts (and bolts) is to try to screw a particular nut N[i] onto a particular bolt B[j]. If the nut is too big for the bolt it will go on loosely, and you can tell that. If the nut is too small for the bolt it won't go on at all, and you can tell that. And if the nut exactly fits the bolt, you can tell that, too. In other words, you can see if a nut is <, =, or > a bolt, but you can't compare nuts to nuts or bolts to bolts.

Describe an expected (i.e. average case) O(nlgn) algorithm to solve the Nuts and Bolts problem**. Justify that your solution is O(nlgn) by providing a recurrence relation and solving it using any method.**

If you cannot think of a O(nlgn) solution you can get half credit by describing a $O(n^2)$ solution.

3) Sorting

Below are the contents of an array as various algorithms are sorting it. Each line does not necessarily represent the next step in the algorithm, but only some later step during the course of the algorithm's execution. Key elements are depicted in **bold** or underline. Identify each algorithm and provide a brief rationale for each answer to make it clear that you did not just guess at a sorting algorithm.

Algorithm 1

23	9	2	12	77	39
12	9	2	23	77	39
12	9	2	23	77	39
2	9	12	23	77	39
2	9	12	23	77	39
2	9	12	23	77	39
2	9	12	**23**	77	39
2	9	12	23	77	39
2	9	12	23	**77**	39
2	9	12	23	39	77

Algorithm 2

12	39	2	94	23	77	52	9
12	**39**	2	94	23	77	52	9
2	12	39	94	23	77	52	9
2	12	39	**94**	23	77	52	9
2	12	**23**	39	94	77	52	9
2	12	23	39	**77**	94	52	9
2	12	23	39	**52**	77	94	9
2	**9**	12	23	39	52	77	94

Algorithm 3

12	39	2	94	23	77	52	9
94	39	77	12	23	2	52	9
77	39	52	12	23	2	9	**94**
52	39	9	12	23	2	**77**	94
39	23	9	12	2	**52**	77	94
23	12	9	2	**39**	52	77	94
12	2	9	**23**	39	52	77	94
9	2	**12**	23	39	52	77	94
2	**9**	12	23	39	52	77	94

Algorithm 4

12	39	02	94	23	77	52	09
1**2**	0**2**	5**2**	2**3**	9**4**	7**7**	3**9**	0**9**
02	**0**9	**1**2	**2**3	**3**9	**5**2	**7**7	**9**4

4) Recurrence relations

Give asymptotically tight Big-O bounds for the following recurrence relations. Justify your solution with the indicated method for solving the recurrence relation.

a) $T(n) = T(n-2) + 1$ Use iterative substitution method
$T(n) = 1 \quad \text{for} \quad n \leq 2$

b) $T(n) = 2T\left(\frac{n}{2}\right) + 1.$ Use induction. Guess T(n)=O(n).
$T(1) = 1$

c) $T(n) = 9T\left(\frac{n}{4}\right) + n$ Use the master theorem.
$T(1) = 1$

5) Lab Section Scheduling

100 students have registered for CS201. Each student must attend one of 5 lab sessions. A lab session cannot hold more than 25 students. Each student has specified three choices for their lab section. A "happiness" rating is assigned to each choice. A student assigned to their first choice is worth 10 points, the second choice is worth 5 points, and the third choice is worth 2 points. An assignment outside the top three is worth 0 points.

Describe how students can be assigned to lab sections so that the happiness of all students is maximized while not exceeding the capacity of any individual lab section.

6) String Matching

Give the pattern "abbab" and the text "abbacabbab":

Initially the pattern and text would line up as follows:

```
Text:       abbacabbab
Pattern:    abbab
```

Using the naïve algorithm, there would be a mismatch at the characters in **bold**:

```
Text:       abba**c**abbab
Pattern:    abba**b**
```

Next, the naïve algorithm would shift the pattern over one character and resume comparisons at the underlined characters:

```
Text:       abbacabbab
Pattern:     abbab
```

a) For the same initial lineup of the pattern and text, show where the mismatch would occur and then how the pattern would be shifted and what characters would be compared next using the Knuth-Morris-Pratt algorithm.

b) For the same initial lineup of the pattern and text, show where the mismatch would occur and then how the pattern would be shifted and what characters would be compared next using the Boyer Moore algorithm.

7) Gossip-Net

Through years of meticulous observation you have determined how many days it takes on average for gossip to travel between members of your family and friends. This data has been compiled into the graph below. For example, Mom spreads gossip daily to your sister, yourself, and Aunt Kiki. She only gossips to Uncle Ned every three days and to Uncle Jed every 42 days.

Use Dijkstra's Algorithm to compute how long it will take gossip to travel from you to everyone else in your social network.

As you run through the algorithm, put a number next to each vertex in the order that you extract them from the min-queue. Next to each vertex you should also give the final distance/cost to the source vertex (You), and the parent vertex that is necessary to recreate the actual shortest path.