Discrete Math Homework 13
Due Wednesday, April 19 at the beginning of class

General instructions:

- Use standard size paper (8.5 by 11).
- Answer each question in order using a single column.
- Be neat. If we cannot read your solution it is wrong.
- Show your work. If you just write an answer, you will get minimal credit even if the answer is correct.

A little more counting

Question A) How many 5-card hands (combinations) are there that are four of a kind. (Four cards of the same rank, and a fifth card that is a different rank.)
Question B) How many 5-card hands (combinations) are there that are two pair. (Two card of one rand, two cards of a second rank, and a fifth card that is a third rank.).

Question C) Fill in the following table to decide the probability that when you roll two dice the sum is a prime number. We are pretending that the dice are distinguishable and form a sequence of rolls. Check each square if the sum is prime. The probability will be the number of squares with a check divided by the total number of squares.

Value of Die 2

Value of Die 1	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

Rosen section 8.2

Question D) Rosen 8.2 Exercise 3b, c (p. 524)

Rosen section 8.3

Question E) Instead of doing a binary search on a sorted list, we can do a trinary search instead. Here is the algorithm:

T-Search *(List, target)*

Let n = length of the list
Let mid1 = n/3

Let mid2 = 2n/3
If the size of the list is 1
Return true if we found the target or false if not.

If target < List[mid1] return ***T-Search****(First third of List, target)*
Else if target < List[mid2] return ***T-Search****(Second third of List, target)*
Else return ***T-Search****(Last third of List, target)*

a) Argue why the recurrence relation for the work W(n) performed by this function is W(n) = W(n/3) + 2 with W(1) = 1

b) Use the Master theorem from page 582 to find the Big-O for W(n)

Question F) Consider the recursive Merge sort

Merge-Sort *(List)*
Let n = length of the list
If n is < 2 return List
Else
Let L1 = first half of the list
Let L2 = second half of the list
Let L1' = ***Merge-Sort****(L1)*
Let L2' = ***Merge-Sort****(L2)*
Let L' = Merge(L1', L2')
Return L'

The recurrence relation for Merge sort is M(n) = 2M(n/2) + cn

a) Use the Master theorem from page 582 to find the Big-O for M(n)

b) One tricky bit, is that the merge operation uses extra memory to copy the values from L1' and L2' into the result list L'. If we want to do the merge without using any extra memory, the time for the merge is no longer proportional to n. A naïve in-place merge would be n-squared giving a recurrence relation of

$$Q(n) = 2Q(n/2) + cn^2$$

Use the Master theorem from page 582 to find the Big-O for Q(n).

c) Suppose that we could do the in-place merge with time that is proportional to $n^{5/4}$, that would result in a recurrence relation of

$$R(n) = RQ(n/2) + cn^{5/4}$$

Use the Master theorem from page 582 to find the Big-O for R(n).

You may choose to solve one (and only one) of the following Extra Credit Problems. If you submit more than one, only the first will be graded.

Extra Credit 1) Compute the probability of being dealt a full house, four of a kind and two pair. (Divide by the number of combinations that could be dealt to you, which is C(52, 5).

Extra Credit 2) Rosen 8.2 Exercise 45 (p. 526)