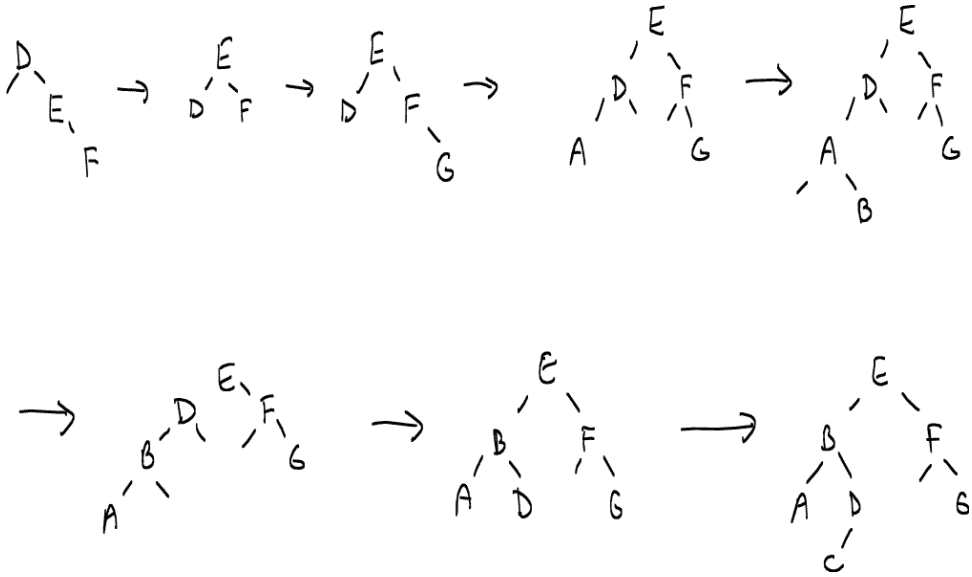
COMP 251 Algorithms and Data Structures
Winter 2014

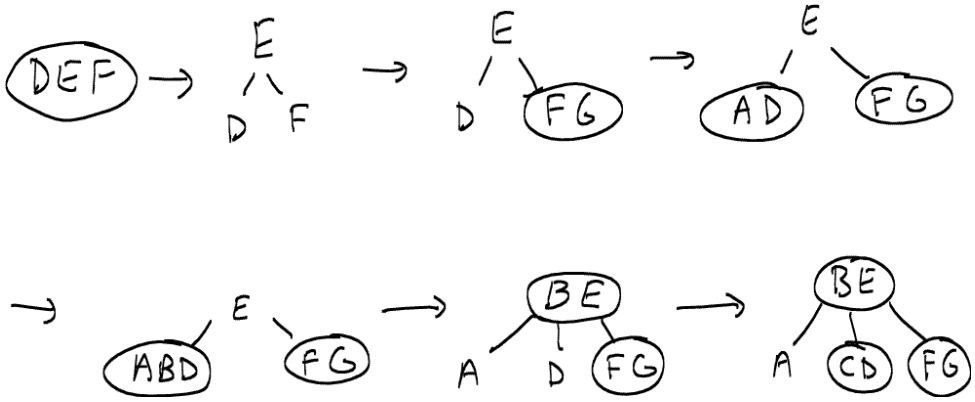
Final Exam Solutions

1)

a)



b)



Grading for (a) – 2 points: 0.5 for giving a binary search tree, 0.5 for inserting everything first and then balancing afterwards with rotations, 0.5 for balancing as you insert but making at least one mistake, 0.5 for getting it right.

Grading for (b) – 2 points: 0.5 for giving a 2-3 tree, 0.5 for inserting at leaves, 0.5 for splitting a 4 node and maintaining balance, and 0.5 for getting it right.

2)

a) The key and value are the element and the count, respectively. For each element in the list, if it is a key in the hash table then increment the count value, else add an entry into the hash table with that key and with count value 1. Next, scan all entries in the hash table and find the key with biggest count. The space and time required for the above are both O(n), i.e. hash table access is O(1) per element.
Note that the question doesn't say whether we are using open hashing (chaining) or closed hashing.

Grading for (a) – 2 points: We gave 0.5 if you noted space and time are O(n). If you just said that the integers are mapped to buckets and then you find the bucket with the most common integer, then we didn't give anything for that. You are not really using a hash table in this case since there is no "value" in your hash map. We gave 0.5 if you proposed a method like "counting sort", where each integer gets mapped to its own value i.e. the size of the hash table is the range of integers in the list. The problem with this solution there is that integers are typically 32 bits and so the range could be enormous. This is not practical.
We gave the full 2 points if you identified that the (key,value) are (number, count).

b) The correct answer is as follows. (Ignore the leftmost column with 0 to 9)

0	D	doug
1	E	edie
2	F	freda
3	B	bob
4	G	greg
5		
6		
7		
8	C	carla
9	A	abbie

We also accepted the following answer which is obtained if you resolve collisions with the sequence h(key), h(key)+1, h(key) + 1 + 4, h(key + 1 + 4 + 9}, etc. The reason we allow this is because in the lecture slides I had mistakenly given an illustration of quadratic probing which used this probe sequence. This mistake was only pointed out to me the day before the final exam.

0	D	Doug
1	E	Edie
2	F	Freda
3	B	Bob
4		
5	G	Greg
6		
7		
8	C	Carla
9	A	Abbie

Grading for (b) – 2 points: 1 point if you the solution was correct up to the point of the first collision (that is, A,B,C were correct). 2 points for either of the above solutions.

3)

a) {BDCGHF} is an SCC since it is a cycle. {A}, {E} are also SCCs.

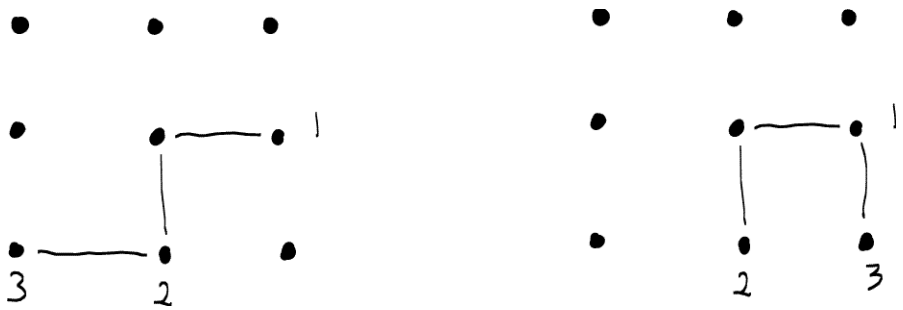
<u>Grading for (a) – 2 points:</u> We gave 1.5 if you identified (BDCG) as an SCC. If you didn't identify the single vertices as SCCs, you lost 0.5 points. i.e. we gave either 1.5 or 2 total for part (a).

b) The graph has a cycle, so there cannot be a topological ordering.

<u>Grading for (b) – 1 point:</u> If there is a cycle, then there cannot be a topological ordering (as proven in class). Alternatively, you can observe that all vertices have at least one outgoing edge.

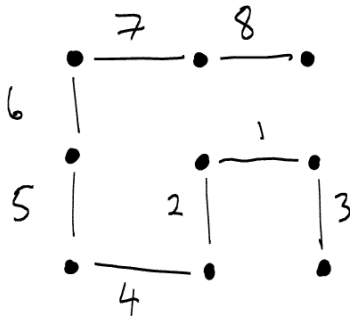
4)

a) There are two solutions. Either is fine.



<u>Grading for (a) – 2 points:</u> 1 point for drawing the edges, 1 point for the numbers.

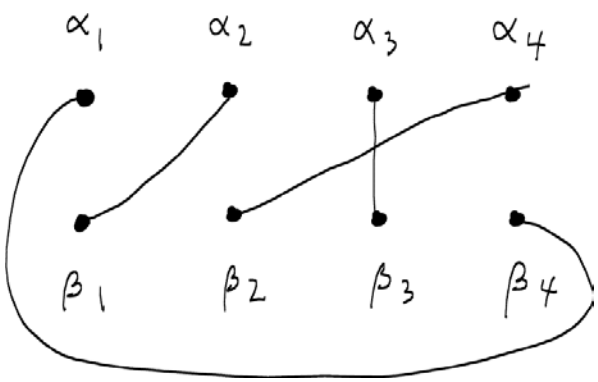
b) Again there are two solutions, namely the order of the 2nd and 3rd edges found can be swapped in the figure below.



<u>Grading for (b) – 2 points:</u> 1 point for drawing the edges, 1 point for the numbers.

5)

a)



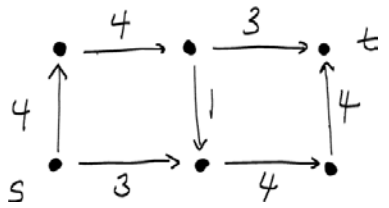
<u>Grading for (a) – 2 points:</u> 1 point for any matching, 2 points for a correct matching.

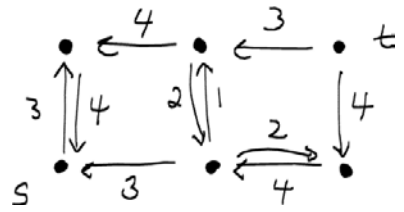
b) We know that if A chooses then A gets its best valid match and B gets their worst valid match, whereas if B chooses then B gets its best valid match and A gets its worst valid match. If the same solution arises when A chooses as when B chooses, then A's best valid matches must be the same as the A's worst valid matches which means that there is only one valid match for the A's and B's.

Another way to answer the question is to note that when the A's choose they all get their best valid match and when the B's choose they all gets their best valid match. So, if the same match arises when the A's choose as when the B's choose, then both As and Bs must getg their best valid matches. So, there can't there be another valid match because, in that case, there would be an a in A and b in B that were not matched but were each other's best valid matches, and hence preferred each other over their current match.

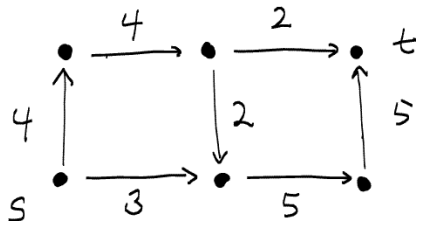
<u>Grading for (b) – 2 points:</u> This one was a bit tricky to grade because it depended on how clearly you argued your point. Sometimes a judgment call was used.

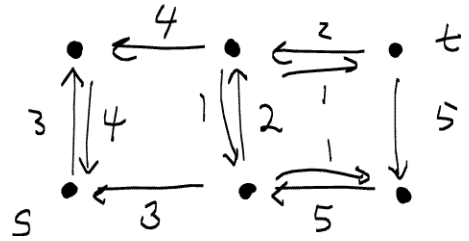
6) There were two solutions to this question. The first one which was much more common was as follows. The max flow and residual graphs are, respectively:



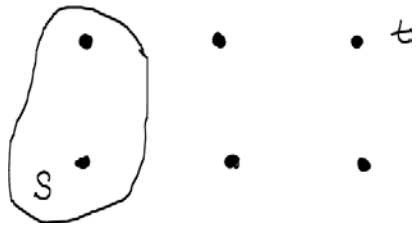


The second solution is:





In both cases, the cut is



<u>Grading - 4 points:</u> 2 points for the max flow, 1 point for residual graph, 1 for the min cut.

7) There are two solutions possible.

For the first solution: $f(n, k) = f(n-2, k-1) + f(n - 1, k)$.

The two terms are as follows:

- $f(n-2, k-1)$, the first segment has list elements 1 and 2 only;
- $f(n-1, k)$, the first segment has list elements 1, 2, 3 and maybe more. In this case, we can group list elements 1 and 2, reducing the problem size by 1.

For the second solution: $f(n, k) = f(n-2, k-1) + f(n - 3, k - 1) + f(n-2, k)$.

The three terms are as follows:

- $f(n - 2, k - 1)$, the first segment has list elements 1 and 2 only;
- $f(n - 3, k - 1)$, the first segment has list elements 1, 2, 3 only;
- $f(n - 2, k)$, the first segment has list elements 1,2,3,4 and possibly more

Notice that the second solution is the same as the first: take the first solution and backsubstitute into the second term in the recurrence!

<u>Grading – 3 points:</u>

We gave 2 points if you had one of the two solutions above, if there was no explanation. If you had part of the solution above but no explanation or the explanation was incorrect, we made a judgment call. If one of the two solutions above was given and there was an explanation, then we only gave the 3rd point if the explanation was correct.

8)

a) opt(n) = max_{ j in 1 to n}(opt(n – j) + v_j), and base case opt(0) = 0.

<u>Grading for (a) – 3 points:</u>

1 point for the max over all the j's, 1 point for opt(n – j) and 1 point for v_ j . Many students wrote a recurrence with two variables opt(n, j) and had extra terms. We penalized you 1 point for this, assuming your solution contained the right side of the above recurrence as one of the terms

b) There are many ways to write the code here, recursive or not. Here is an example of what we were looking for.

```
Initialize a counter.    For each j = 1 to n,  count[ j ] = 0.
k = n
while k > 0{
        for i = 1 to k
            if opt[k] = opt[k – i] + v_i
                    count[i] ++
                     k = k - i
                      break  // for loop
    }
```

<u>Grading for (b) – 1 point:</u> We were looking for two elements for 0.5 points each. First, the opt(n- j) from the recurrence was needed i.e. using the solution from the smaller problem size. Second, some mechanism for computing a count of the number of pieces of each length.

9)

a)

This was one of the more challenging questions on the exam. The question asks for an upper bound, i.e. big O. What makes it challenging is that it asks for an upper bound on a minimal value (shortest distance). The concept of a "max of a min" can be confusing. But it did come up a few times in the course and it is important beyond this course too.

There is a trivial upper bound, namely O(1) but we want a non-trivial upper bound.

The way to approach the problem is to notice that this upper bound on closest pair occurs if the elements roughly form a square grid. Since there are n points, the grid would be of dimension sqrt(n) x sqrt(n). Since the size of the square is 1x1, the smallest distance between points would be about 1/sqrt(n). So the solution is **O(1/ sqrt(n)).**

<u>Grading for (a) – 2 points:</u> If you wrote O(1/n) and/or expressed the idea that the solution is a decreasing function of n. We gave 0.5 for each of those.

b) The area of a vertical strip of width 2*delta and of height 1 is 2*delta. The probability that any point lands in this vertical strip is p = 2*delta. Let X_j be a random variable that takes value 1 when point j lands in the vertical strip and 0 when it doesn't. The number of points that land in the vertical strip is the sum over j of X_ j. By linearity of expectation, the expected number of points in the strip is n * 2 * delta. Using delta = c/ sqrt(n) from (a), we get the solution for (b) is O(n/ sqrt(n)), i.e. **O(sqrt(n))**.

<u>Grading for (b) – 1 point:</u> If you gave a linearity of expectation argument and observed that the number of points grows as O(delta *n), then we gave the point here.

10)

a) Partition the list of n numbers into two lists, each with n/2 numbers. Compute the sum of each list (recursively). Then add together the two sums.

<u>Grading for (a):</u> 1 point. Many students misunderstood the question and thought it was analogous to Karatsuba multiplication. Some other students suggested dividing the problem into columns.

b) T(n) = 2 T(n/2) + 1. The "1" is for the addition of the two sums.

For the Master Method, we have a = 2, b=2, d=0, so it's a > b^d (the third one), which gives O(n).

<u>Grading for (b):</u> 1 point. Many students gave the recurrence with a "cn" term instead of a "1", which is the same as mergesort. Perhaps you were thinking that the partitioning takes time proportional to n. But if the list is an array (as in typical mergesort Implementations) then the partition takes O(1) time, not O(n) times. In mergesort, the "cn" term is for the merging step.

c) T(n) = 2 T(n/2) + c

= 2 (2 T(n/4) + c) + c

= 4 T(n/4) + 2c + c

= 4(2 T(n/8) + c) + 2c + c

= 8 T(n/8) + 4 c + 2 c + c

= n T(n/n) + c * sum_{i = 0}^ {(log n) – 1} 2^i

= n + c 2^ log n

= n + cn

= n (1 + c)

<u>Grading for (c):</u> 2 points. The first point is for solving the recurrence by backsubstitution. If you put a "c n" instead of "1" or "c" in (b), then we looked at whether you solved the recurrence in (b) correctly.
The second point was to observe that the usual way to add n integers takes time O(n) namely constant time for each integer added to the accumulated sum. The mergesort recurrence would give O(n log n) which is slower than O(n).

11)

a) Following the same pattern for the median of median of 5's gives the recurrence:

$t(n) = t(n/7) + t(11 n / 14) + c n.$

You weren't required to explain why, but here is the explanation: The $t(n/7)$ term is needed to recursively find the median of the median of 7's. Once that median (call it P for pivot) is obtained, we can examine each 7 tuple. (There are n/7 of them.) If the median of the 7-tuple is less than P then the smallest three elements of the 7-tuple will be less than P, whereas if the median of the 7-tuple is greater than P then the largest three elements of the 7-tuple will be greater than P. Half of the 7-tuples will be of the former type and half of the latter. This means that the pivot will reduce the size of the list to be at most $n*(1 - (3/7)/2) = n*(11/14)$. That accounts for the $t(11 n / 14)$ term. The linear term $c n$ is there for computing the partition, once the pivot P has been chosen.

<u>Grading for (a) - 3 points:</u> namely the three terms in the recurrence. We gave 0.5 instead of 1 for the second term if you wrote $t(10 n/ 14)$ or something else close.

b) Suppose $t(k) < b k$ for some b and for all **$k < n$**. That is the induction hypothesis. The induction step is then to show that $t(k) < b k$ for **$n=k$**. Following the pattern of the median of median of 5's case that was done in the lecture, let $b = 14 c$ where c is from the recurrence. Then,

$$
\begin{aligned}
t(n) &< b n/7 + 11 b n / 14 + c n \\
&= 14 c n/ 7 + 14 c*11 n / 14 + c n \\
&= (2 + 11 + 1) c n \\
&= b n. \quad \text{Thus, } t(n) < b n \text{ as required.}
\end{aligned}
$$

The above few lines is what I wanted. It is not a complete proof by induction because it only gives the induction step. A full proof would require showing the induction hypothesis is also true. This can be done by choosing the constants c and b to be sufficiently. Details omitted.

<u>Grading for (b):</u> 1 point for the substitution $b = 14 c$. We gave 0.5 if you substituted $b k$ for $t(k)$ and the other 0.5 points for giving an argument that combines the b and c constants.

12) Let X_i j be 1 if there is a collision between i and j where i < j and i and j go from 1 to n. There are n(n-1)/2 pairs (i, j) where i < j. The number of collisions is the sum of X_ i j over i and j where i < j. By linearity of expectation, the expected number of collisions is the sum of E{ X_ i j} . But E{ X_ i j} = 1/m. Thus, the answer is n*(n-1)/(2m).

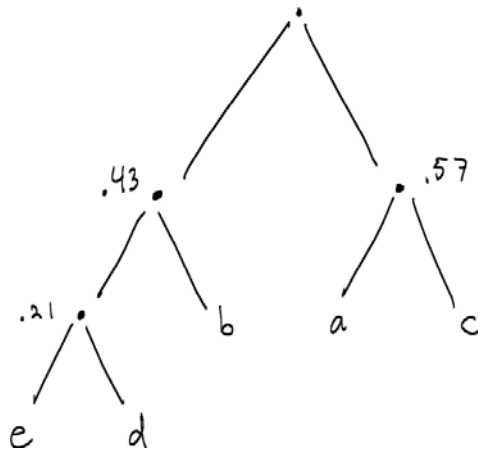
<u>Grading scheme - 3 points:</u> 1 point for noting there are n(n-1)/2 pairs.
1 point for noting the expected value of the number of collisions for one pair of entries is 1/m.
1 point for putting those together (applying linearity of expectation).

Many students didn't seem to understand the basics of linearity of expectation or didn't understand the question. Many made arguments along the following lines. The number of points for any of the three is given below.

- (1 point) "The probability of a collision is 1/m and there are n entries to be added. So by linearity of expectations, the expected number of collisions is n/m." The problem with this argument is that a collision is defined by two elements and the answer says nothing about the number of pairs. Rather, what the argument is really saying is that the expected number of entries in each bucket is n/m. But that's different from saying what the number of collisions is.
- (1 point) A slight more sophisticated argument that many students gave, but which is also wrong, went like this: "The expected number of entries in each bucket is n/m. But the first entry isn't a collision, so the expected number of collisions in each bucket is n/m – 1. Summing over m buckets gives m*(n/m – 1) = n – m. " The problem with this argument is that the expected number of collision in each bucket is not n/m – 1. Collisions are pairwise, so in fact the number of collisions in each bucket is the number of *pairs* of entries in each bucket.
- (2 points) An even more sophisticated argument went like this. "The probability of a collision when you add the ith entry is i/m, since the more entries you have in the table, the greater the probability. Summing over I gives the answer, which is n(n-1)/2 / m." The problem with this answer is that it says i/m is the probability of a collision. But that's wrong. Rather, i/m is the *expected value of the number of collisions of the ith element with elements 1, …, i-1*.

13)

a)



The question did not specify that the left branch is 0 and the right branch is 1. It turns out not to matter. The important thing is that the lower probability event is labelled 0 and the higher probability event is labelled 1.

<u>Grading scheme for (a) - 2 points:</u> I started with 2 points if you seemed to know what a Huffman code is (e.g. no letters at internal nodes) and then I took off 0.5 points for every small error you made.

b) 10 10 01 000 001 11 gives "aabedc".

<u>Grading scheme for (b) - 1 point:</u> If you gave the wrong Huffman tree for (a), then we examined whether you decoded the bit string according to the tree *you gave* for (a).

14) This was perhaps the most challenging question on the exam. I didn't expect a complete solution. I was just looking for the main idea:

For each of the two initial lists A, B, select the median that is, i = n/2. Then compare these two medians. Without loss of generality, *suppose the median of A is less than the median of B*. In that case, we know that all the n/2-1 elements that are smaller than the median of A are also smaller than the median of B. Thus, the n/2 elements that are less than or equal to the median of A are smaller than all elements that are greater than the median of A (n/2) and the elements that are greater than or equal to the median of B (n/2+1). *Thus, the n/2 elements in A that are less than or equal to the median of A cannot contain the median of {A,B}*. By similar reasoning, *the n/2 elements of B that are greater than or equal to the median of B cannot contain the median of {A,B}.*

If you gave the above argument, you got full points.

That's not a complete solution of course: we still need to find the median of {A,B}. To do so, we just repeat what we did above, but we consider only the elements of A that are greater than the median of A and the elements of B that are less than median of B. We select the median of these two lists, namely select(A, 3n/4) and select(B, n/4), and we compare these two medians. We can rule out half the remaining elements, depending on which of the two medians is smaller. Essentially, we are doing a binary search on the two lists, cutting the size of the list by a factor 2 each time.

<u>Grading scheme - 3 points:</u>

1 point - select the median for each of the two lists.
1 point for realizing it is useful to compare the two medians and see which is larger,
1 point for observing that half the elements in each list can be ruled out using some idea such as above.

If you wrote other things which indicated that you didn't understand the question, then you might have lost marks from the above.