60 marks **CS 218 Quiz 3** **8:30-9:25, 18/3/16**

There will be no clarifications. In case of doubts please make the most reasonable assumptions about what the intent of the question is, state them, and proceed.

Problem 1: Your are given a matrix A where a_{ij} represents the distance from node i to node j of a graph G. You are also given as inputs vertices u, v of G. You are to find the shortest path from u to v passing through every vertex of G exactly once.

(a)[5 marks] State what decisions you need to make to solve the problem. Your answer should be of the form: "The ith decision is: ...". No justification should be necessary; but if you feel the need to justify, please give only 2-3 lines.

The ith decision is: what is the ith point on the path.

(b)[10 marks] Give a recursive function to find the shortest path in a brute force manner. The function should be described at a high level, but the arguments to it must be clearly stated. This should use the answer of part (a) and no justification need be given. You can assume appropriate data structures to represent sets etc. State how you make the initial call to your function.

HP(start vertex x, end vertex y, candidate intermediate vertices W){

for every w in W: P(w) = w || HP(w, y, W - w);

Return that P(w) which has shortest length.

}

(c)[10 marks] State how many distinct calls will be made to your recursive function. Justify your answer. What is the total number of calls that could be made (including making the same call in several parts of the recursion)? No memoization is to be done. Only answer what is asked.

The start vertex x can take at most $n-1$ different values. The end vertex will always be v. The set W will be some subset of the set of vertices $V - \{u, v\}$. There are $n-1$ ways to choose x, and 2^{n-2} ways to choose W. Thus overall there will be at most $(n-1)2^{n-2}$ different argument choices. Hence at most that many distinct calls.

5 marks

The number of different leaf level calls will be $n-2!$, because every permutation of all vertices other than u, v will get considered in some call.

5 marks

The point to note here is that the brute force solution is directly memoizable. But to see this, you cannot have any global variables to keep track of which vertices have been selected. On the other hand global variables which are read only are perfectly fine. This is also a good programming practice – the specific information you want to pass to a call is more readably passed as an argument.

Problem 2: Suppose you are given a set S of character strings. You are also given another string T. Give an algorithm to decide whether T can be constructed by concatenating the strings in S. For example, if S contains the strings "abc", "abcde", "defg", and T is "abcdefg" the answer is YES. However if S only contains "abcde" and "defg" and T is "abcdefg" the answer is NO. You should also state which strings in S form T in which order. Answer all this through the parts (a)–(d) given below. For simplicity you can assume that S is held in a data structure which given a string t of any length determines in time $O(1)$ whether t is present in S.

(a)[5 marks] What decisions do you need to make? What would be the first decision? Clearly state the possible outcomes of each decision. Only a 4-5 line answer is expected.

What is the first word? The choices would be the set of words in S which are also prefixes of T. The decision would be to pick one of those. Given the choice for the first, the second decision would be to pick the second word and so on.

Note that the question "what decision" is used in the same sense as we have been using in class. "What decision" should not be mixed with what is a good decision and what is a bad decision – that comes later. First figure out what freedom you have and list out your options clearly. If you start making value judgements too early, you might get carried away and not realize what choices you really have.

Also, it might help to interpret the question "what decisions" in the following sense. You are required to find a certain "solution", which could be a path, or a set of words. "The decisions" are about how you choose the PARTS of the solution. In order to construct the solution you need to break it down into parts, and you need to decide what possible candidates are for each part.

(b)[10 marks] Give a recurrence the solution of which says whether T can be formed using the strings in S. Clearly explain the interpretation of the terms in the recurrence. Justify the recurrence in 4-5 lines max.

I am giving a solution on the assumption that words in S can be repeated. Some people did not allow repetition; that is fine too, and you get marks for that too.

Valid(i) will denote whether T[i..n] can be built by concatenating words in S.

Valid(i) = $OR_{j=i..n}$ T(i..j) is a word and Valid(j+1)

Base case Valid(n+1..n)=true.

We want the value of Valid(1).

(c)[5 marks] What table will you use and in what order will you fill it? How much time will it take?

The table is one dimensional and to be filled from the end, i.e. in decreasing order of i. To fill the ith element, we may need to examine all $j \geq i$, hence $O(n)$ time per entry, for a total of $O(n^2)$.

(d)[5 marks] State how you will recover the strings that make up T from your table.

Start from $i = 1$, and find smallest j s.t. that $T[i..j]$ is a word and Valid(j+1), then we can emit T[i..j]

as a constituent string. We then recurse with $i = j + 1$. Note that if Valid(1)=true, then such a j must exist.