

## Solutions to Sample Problems

### Problem 1: TRUE/FALSE Questions with Justifications

For each of the following statements, answer whether it is true or false, and justify your choice in *at most* 3 sentences. Answers exceeding 3 sentences *may not* be read. Answers without justifications will *not* receive full credit. A justification need *not* be a full proof, and is for demonstration of your understanding only.

1. The partition subroutine of Quick Sort can be modified so that Quick Sort runs in  $O(n \log n)$  time in the *worst case*.

**Answer: TRUE.** Just use the deterministic  $O(n)$  time selection algorithm to find the median and use the median as the pivot.

2. BFS can be used to decide whether a given *undirected* graph  $G = (V, E)$  has a cycle of odd length in time  $O(V + E)$ .

**Answer: TRUE.** It can be shown that  $G$  has a cycle of odd length if and only if  $G$  is *not* 2-colorable, and we saw in HW6 that BFS can be used to decide whether  $G$  is 2-colorable.

3. Let  $T_e$  be the time for each extract-min operation and  $T_d$  that for each decrease-key operation of a give priority queue. Then using this priority queue Dijkstra's algorithm for computing single-source shortest paths runs in time  $O(E \cdot T_e + V \cdot T_d)$ .

**Answer: FALSE.** Should be  $O(V \cdot T_e + E \cdot T_d)$ .

4. To show that a language  $L$  is **NP-Complete**, it suffices to show that (1)  $L \in \mathbf{NP}$ , and (2)  $L$  reduces to 3SAT in polynomial time.

**Answer: FALSE.** (2) should be 3SAT reduces to  $L$  in polynomial time.

5. Define the language 3CLIQUE as

$$3\text{CLIQUE} = \{ \langle G \rangle : G \text{ has a clique of size } 3 \}.$$

Then 3CLIQUE is **NP-Complete**.

**Answer: False unless  $\mathbf{P} = \mathbf{NP}$ .** The following is a  $O(n^3)$  time algorithm for deciding 3CLIQUE: For each triple  $(u, v, w)$  of vertices, check whether  $(u, v, w)$  forms a clique. We know that if an **NP-Complete** problem is in  $\mathbf{P}$ , then  $\mathbf{P} = \mathbf{NP}$ .

### Problem 2: Questions with Short Answers

1. Suppose there are  $n$  files  $F_1, F_2, \dots, F_n$  with lengths  $l_1, l_2, \dots, l_n$  respectively, and we would like to concatenate them into a single file  $F = F_1 \circ F_2 \circ \dots \circ F_n$  of length  $l = l_1 + l_2 + \dots + l_n$ . The only primitive operation available is a binary concatenation operation that concatenates *two* given files  $F$  and  $F'$  of lengths  $l$  and  $l'$  respectively, into a single file of length  $l + l'$ . The cost of this operation on two input files of lengths  $l$  and  $l'$  is given by a cost function  $C(l, l')$ . Give an  $O(n^3)$  time algorithm that finds an optimal sequence of binary concatenations.

**Answer:** The idea is exactly the same as that for matrix chain multiplication. Let  $M[i, j]$  denote the optimal cost for concatenating  $F_i, \dots, F_j$ . Then  $M[i, i] = 0$  for each  $i$ , and for  $i < j$ ,

$$M[i, j] = \min_{i \leq k < j} \left\{ M[i, k] + M[k + 1, j] + C \left( \sum_{t=i}^k l_t, \sum_{t=k+1}^j l_t \right) \right\}.$$

This gives an  $O(n^3)$  algorithm just as the one for matrix chain multiplication.

2. Give an efficient algorithm that given a weighted directed graph  $G = (V, E)$  with *positive integral* edge weights, and a start vertex  $s$ , finds for each vertex  $v \in V$  a *shortest path* from  $s$  to  $v$  with the *smallest number of edges*.

**Answer:** There are a number of ways to do this. The following one is probably the simplest that takes advantage of the fact that the given edge weights are *integers*.

Given  $G = (V, E)$  and an *integer* weight function  $w$ , define a new weight function  $w'$  on  $G$  as follows: for each  $(u, v) \in E$ , let  $w'(u, v) = w(u, v) + 1/n$ , where  $n = |V|$ . Then just run any algorithm for single-source shortest paths, say Dijkstra's algorithm, on  $(G, w')$ .

We now show that this is correct. (**Note:** On the exam you need *not* prove correctness unless you are asked.)

For a simple path  $P$  in  $G$ , and let  $l(P)$  denote its length (i.e. number of edges in  $P$ ). Note that (1)  $w'(P) = w(P) + l(P)/n$ , and (2)  $l(P) \leq n - 1 < n$ . Thus,  $w'(P) < w(P) + 1$ .

Let  $P_1$  and  $P_2$  be two paths. Thus if  $w(P_1) = w(P_2)$  and  $l(P_1) < l(P_2)$ , then  $w'(P_1) = w(P_1) + l(P_1)/n < w(P_2) + l(P_2)/n = w'(P_2)$ . Similarly  $w(P_1) = w(P_2)$  and  $l(P_1) = l(P_2)$ , then  $w'(P_1) = w'(P_2)$ . On the other hand, if  $w(P_1) < w(P_2)$ , then since  $w(u, v)$  is an *integer* for every  $(u, v) \in E$ , we must have  $w(P_1) + 1 \leq w(P_2)$ . Thus  $w'(P_1) < w(P_1) + 1 \leq w(P_2) \leq w'(P_2)$ , regardless of what  $l(P_2)$  is.