Final Exam of Algorithms (Spring 2021)

June 26, 2021

Problem 1 Let S be a finite set. A binary *relation* on S is simply a collection R of ordered pairs $(x, y) \in S \times S$. An *equivalence relation* is a binary relation which satisfies three properties

- Reflexivity: $(x, x) \in R$ for all $x \in S$
- Symmetry: if $(x, y) \in R$ then $(y, x) \in R$
- Transitivity: if $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$

For instance, the binary relation "has the same birthday as" is an equivalence relation, whereas "is the father of" is not, since it violates all three properties.

1. Show that an equivalence relation partitions set S into disjoint groups $S_1, \ldots, S_k$ (in other words, $S = S_1 \cup S_2 \cup \cdots \cup S_k$) and $S_i \cap S_j = \varnothing$ for all $i \neq j$) such that
   - Any two members of a group are related, that is, $(x, y) \in R$ for any $x, y \in S_i$, for any i.
   - Members of different groups are not related, that is, for all $i \neq j$, for all $x \in S_i$ and $y \in S_j$, we have $(x, y) \notin R$.

   (10 marks)
2. Design an algorithm to determine k (the number of disjoint groups) in $O(|R|)$ time. (10 marks)

Problem 2

1. You are given a sorted array of n distinct elements and would like to determine whether the array contains a target value x. The binary search algorithm compares x to the middle element of the array. If they are not equal, the half in which the target cannot lie is eliminated and the search continues on the remaining half, again taking the middle element to compare to x. Repeating this until the target value is found or the array is empty. Write down the pseudocode of this algorithm and prove that $O(\log n)$ comparisons are required in the worst case. (10 marks)
2. You are given a *unimodal* array of n distinct elements, meaning that its entries are in increasing order up until its maximum element, after which its elements are in decreasing order. Give an algorithm to compute the maximum element of a unimodal array with $O(\log n)$ comparisons. Prove the correctness of your algorithm and analyze the running time rigorously. (10 marks)

Problem 3 Let $G = (V, E)$ be an undirected graph and $q \geqslant 1$ be an integer. A proper q-coloring of G is an assignment $c : V \to \{1, 2, \ldots, q\}$ such that $c(u) \neq c(v)$ for any $\{u, v\} \in E$. The problem q-Coloring is to determine whether an input graph G has a proper q-coloring.

1. Prove that for any $q \geqslant 1$, q-Coloring is in **NP**. (5 marks)

2. Prove that there exists a Karp reduction from 3-Coloring to 4-Coloring, namely 3-Coloring $\leqslant_K$ 4-Coloring. (15 marks)

Problem 4 Let $G = (V_1 \cup V_2, E)$ be a bipartite graph where $E \subseteq V_1 \times V_2$. A *perfect matching* of G is a set of edges $M \subseteq E$ touching each vertex in $V_1 \cup V_2$ *exactly once*. That is, for every $v \in V_1 \cup V_2$, there exists exactly one edge $e \in M$ such that $v \in e$. Consider the problem of determining whether a given graph contains a perfect matching.

1. Show how to find the perfect matching (if any) by reducing to the problem of maxflow. (15 marks)

2. *Hall's condition* is a combinatorial characterization of the existence of perfect matching:

   Proposition 1. *The graph* G *contains a perfect matching* if and only if *for any* $S \subset V_1$, $|N(S)| \geqslant |S|$, *where* $N(S) = \{u \in V_2 \mid \{u, v\} \in E$ *for some* $v \in S\}$ *is the set of neighbours of vertices in* S.

   Prove Hall's condition. (15 marks)

Problem 5

1. Given an unlimited supply of coins of denominations $x_1, \ldots, x_n$, we wish to make change for a value v; that is, we wish to find a set of coins whose total value is v. This might not be possible: for instance, if the denominations are 5 and 10 then we can make change for 15 but not for 12. Give an $O(nv)$ dynamic-programming algorithm for the following problem

Input:	Positive integers $x_1, \ldots, x_n$; v
Problem:	Is it possible to make change for v using coins of denominations $x_1, \ldots, x_n$?

   (10 marks)

2. Suppose now we only allow to use each denomination *at most once*. For instance, if the denominations are $1, 5, 10, 20$, then you can make change for $16 = 1 + 15$ and for $31 = 1 + 10 + 20$ but not for 40 (because you can't use 20 twice). Show how to solve the following problem in $O(nv)$ time.

Input:	Positive integers $x_1, \ldots, x_n$; v
Problem:	Is it possible to make change for v using each denominations x_i at most once?

   (10 marks)

3. Suppose now you are allowed to make change for a value v using at most k coins (but each denominations can be used unlimited times). Give an efficient algorithm for the problem. What is the complexity of your algorithm?

Input:	Positive integers $x_1, \ldots, x_n$; k; v
Problem:	Is it possible to make change for v using at most k coins, of denominations $x_1, \ldots, x_n$?

(10 marks)