Final Exam

Mengxuan Wu

This final exam paper has been reconstructed from memory, and as a result, some details may be missing. I hope it serves as a useful reference and aid for future students.

Problem 1: Greedy

You have n suspicious transactions, each occurring at time t_i with an error tolerance e_i. A new account also has n transactions occurring at times m_i. The i-th transaction from the new account can be linked to the j-th transaction from the old account if the time difference is within the error tolerance, i.e., $|m_i - t_j| \leq e_j$. Each transaction can only be linked once.

Design an algorithm to determine if all n transactions from the old account can be linked to transactions from the new account. An $O(n^2)$ time complexity is acceptable. Explain why your algorithm is correct.

Problem 2: Divide and Conquer

A node in a complete binary tree is considered a local minimum if its value is less than or equal to the values of all its neighbors (not just its children).

Design an algorithm to find a local minimum in a complete binary tree with $O(\log n)$ time complexity, starting from the root node (n is the number of nodes in the tree). You only know the value of a node when you visit it. Explain why your algorithm is correct.

Problem 3: Dynamic Programming

As the manager of a computer shop, you can buy new computers at a fixed price P at the start of each month (no matter how many computers you buy, the price will always be P). Each month i, c_i computers will be sold immediately. Unsold computers are stored in a warehouse with capacity W and a monthly storage fee F per computer. Given the number of computers sold over n months, design an algorithm to minimize the total cost of buying and storing computers. The time complexity of your algorithm should be a polynomial in n and W.

Problem 4: Polynomial Time Reduction

Two algorithms are defined as follows:

- VERTEX-COVER: Determines in polynomial time if a graph G has a vertex cover of size k.
- FIND-VERTEX-COVER: Finds a vertex cover of size k for a graph G in polynomial time.

Prove that VERTEX-COVER and FIND-VERTEX-COVER are polynomial-time equivalent, i.e., VERTEX-COVER $\equiv_p$ FIND-VERTEX-COVER.

Problem 5: Network Flow

You have a computer with operating system A and n software programs. A new operating system B is installed on the same computer. Transplanting software i from A to B gives a performance improvement of $p_i \geq 0$. Some software program pairs i and j work closely together: if only one of them is transplanted, there's a performance degradation of $d_{ij} \geq 0$. Software 1 cannot be transplanted to B.

Design an algorithm to maximize the net performance improvement after transplanting the software programs. Explain the correctness of your algorithm (excluding the network flow algorithm's correctness).

Problem 6: Randomized Algorithm

To find a four-coloring of a graph G, an edge is considered satisfied if its two endpoints have different colors. Design a randomized algorithm to color G such that at least $\frac{3}{4}$ of the edges are satisfied. Explain why your algorithm is correct.

Solutions

The solutions are from myself and may not be the best solutions.

Problem 1: Greedy

Initialize an empty bipartite graph with n vertices on each side. If the i-th transaction from the new account can be linked to the j-th transaction from the old account, add an edge between the i-th on the left side and the j-th on the right side. Run the extended Gale-Shapley algorithm to find a maximum matching. If the matching is perfect, all transactions can be linked.

Problem 2: Divide and Conquer

To achieve $O(\log n)$ time complexity, obviously we go through the tree in a binary search manner.

```
Algorithm 1: Find-Local-Minimum
  Input: Graph $G$, root node $r$
  Output: Local minimum
  Binary-Search($G$, $r$);
```

```
Algorithm 2: Binary-Search
  Input: Graph $G$, current node $v$
  Output: Local minimum
  if $v$ is a leaf node then
    return $v$;
  end
  if $v$ is smaller than both children then
    return $v$;
  end
  if $v$'s left child is smaller than $v$ then
    return Binary-Search($G$, $v$'s left child);
  end
  return Binary-Search($G$, $v$'s right child);
```

Problem 3: Dynamic Programming

Let $dp[i][j]$ be the minimum cost of buying and storing computers for the first i months with j computers in the warehouse. The recurrence relation is as follows:

$$dp[i][j] = \min \begin{cases} dp[i-1][k] + F \cdot k & \text{if } k = j + c_i\text{, i.e., do not buy any new computers} \\ dp[i-1][k] + P + F \cdot k & \text{if } k < j + c_i\text{, i.e., not enough computers in the warehouse} \end{cases}$$

Problem 4: Polynomial Time Reduction

Omitted since it's presented in the lecture slides.

Problem 5: Network Flow

Similar to image segmentation problem, we can construct a graph with n nodes representing the software programs and extra nodes s and t. For each software program i, add an edge from s to i with capacity p_i and an edge from i to t with capacity ∞. For each software program pair i and j, add edges from i to j and from j to i with capacity d_{ij}. Run a maximum flow algorithm to find the maximum net performance improvement.

Problem 6: Randomized Algorithm

Same as the $\frac{7}{8}$-approximation algorithm for SAT, which is presented in the lecture slides.