1 Exercises Advanced Algorithms (Part II)

1.1 Sorting Networks, Counting Networks

Question 1 (CLRS, Question 27.2-2). Prove that a comparison network with \( n \) inputs correctly sorts the input sequence \( \langle n, n-1, \ldots, 1 \rangle \) if and only if it correctly sorts the \( n-1 \) zero-one sequences \( \langle 1, 0, 0, \ldots, 0, 0 \rangle, \langle 1, 1, 0, \ldots, 0, 0 \rangle, \ldots, \langle 1, 1, 1, \ldots, 1, 0 \rangle \).

Question 2. [CLRS, Question 27.2-5] Prove that an \( n \)-input sorting network must contain at least one comparator between the \( i \)-th and \( (i+1) \)-st lines for all \( i = 1, 2, \ldots, n-1 \).

Question 3. [CLRS, Question 27.4-3] Show that any network that can merge 1 item with \( n-1 \) sorted items to produce a sorted sequence of length \( n \) must have depth at least \( \log n \).

Question 4. How many binary bitonic sequences of length \( n \) are there?

Question 5. [CLRS, Problem 27-1] An odd-even-sorting network on \( n \) inputs \( \langle a_1, a_2, \ldots, a_n \rangle \) is a transposition sorting network with \( n \) levels of comparators connected in the “brick-like” pattern illustrated below.

```
   a1  b1
   a2  b2
   a3  b3
   a4  b4
   a5  b5
   a6  b6
   a7  b7
   a8  b8
```

As can be seen in the figure, for \( i = 1, 2, \ldots, n \) and \( d = 1, 2, \ldots, n \), line \( i \) is connected by a depth-\( d \) comparator to line \( j = i + (-1)^{i+d} \) if \( 1 \leq j \leq n \).

Prove that odd-even sorting networks actually sort.

Question 6. Prove that any sorting network must have depth \( \Omega(\log n) \).

Question 7. Give a construction of a sorting network of depth \( O(\log^2 n) \) that works even if \( n \) may not be a power of 2.

Question 8. Construct a network that is a sorting network but not a counting network.

Question 9. Prove that a perfect halver for \( n \) inputs must have depth \( \Omega(\log n) \).
1.2 Matrix Multiplication and Multithreading

**Question 10.** Exam Question 2014 Paper 3 Question 1 (Algorithms II). **Warning:** Parts of the solution may require Theorem 27.1 (Greedy-Scheduler-Theorem) in CLRS3, which is not covered in the lectures.

**Question 11.** [CLRS, Question 4.2-7] Show how to multiply the complex numbers $a + bi$ and $c + di$ using only three multiplications of real numbers. The algorithm should take $a, b, c,$ and $d$ as input and produce the real component $ac−bd$ and the imaginary component $ad+bc$ separately.

**Question 12.** What can be said about the relation between the time complexity for multiplying two arbitrary square matrices $A$ and $B$ and the time complexity for multiplying a matrix $C$ with itself?

**Question 13.** [CLRS, Question 4.2-2] Write pseudocode for Strassen’s algorithm.

1.3 Linear Programming

**Question 14.** [CLRS: 29.1-5] Convert the following linear program into slack form:

\[
\begin{aligned}
\text{maximize} & \quad 2x_1 - 6x_3 \\
\text{subject to} & \quad x_1 + x_2 - x_3 \leq 7 \\
& \quad 3x_1 - x_2 \geq 8 \\
& \quad -x_1 + 2x_2 + 2x_3 \geq 0 \\
& \quad x_1, x_2, x_3 \geq 0
\end{aligned}
\]

What are the basic and non-basic variables?

**Question 15.** [CLRS: 29.1-6] Show that the following linear program is infeasible:

\[
\begin{aligned}
\text{maximize} & \quad 3x_1 - 2x_2 \\
\text{subject to} & \quad x_1 + x_2 \leq 2 \\
& \quad -2x_1 - 2x_2 \leq -10 \\
& \quad x_1, x_2 \geq 0
\end{aligned}
\]

**Question 16.** [CLRS: 29.1-7] Show that the following linear program is unbounded:

\[
\begin{aligned}
\text{maximize} & \quad x_1 - x_2 \\
\text{subject to} & \quad -2x_1 + x_2 \leq -1 \\
& \quad -x_1 - 2x_2 \leq -2 \\
& \quad x_1, x_2 \geq 0
\end{aligned}
\]

**Question 17.** [Thanks to the student for mentioning this question.] Consider the linear program for the minimum-weight shortest-path from $s$ to $t$ from the lecture notes (Slide 23 from III Linear Programming).
1. What happens if there exists a negative-weight cycle?

2. Prove that, if there are no negative-weight cycles, the optimal solution $\tilde{d}_t$ of the linear program equals the correct distance $d_t$.

3. Find a counter-example in which the linear program does not compute all values $d_v$ correctly. How would you formulate the single-source-shortest path problem as a linear program?

**Question 18.** Prove that the set of feasible solutions of a linear program in standard form forms a convex set.

**Question 19.** [Thanks to the student for mentioning this question (and answer).] Find a linear program which has at least one optimal solution that is not a vertex.

**Question 20.** [CLRS: 29.1-8] Suppose that we have a general linear program with $n$ variables and $m$ constraints, and suppose we convert it into standard form. Give an upper bound on the number of variables and constraints in the resulting linear program.

**Question 21.** [CLRS: 29.1-9] Give an example of a linear program for which the feasible region is not bounded, but the optimal objective value is finite.

**Question 22.** [CLRS: 29.2-5] Rewrite the linear program for maximum flow so that it uses only $O(V + E)$ constraints.

**Question 23.** [CLRS: 29.3-6] Solve the following linear program using SIMPLEX:

$$\begin{align*}
\text{maximize} & \quad 5x_1 - 3x_2 \\
\text{subject to} & \quad x_1 - x_2 \leq 1 \\
 & \quad 2x_1 + x_2 \leq 2 \\
 & \quad x_1, x_2 \geq 0
\end{align*}$$

**Question 24.** [CLRS: 29.5-5] Solve the following linear program using SIMPLEX:

$$\begin{align*}
\text{maximize} & \quad x_1 + 3x_2 \\
\text{subject to} & \quad x_1 - x_2 \leq 8 \\
 & \quad -x_1 - x_2 \leq -3 \\
 & \quad -x_1 + 4x_2 \leq 2 \\
 & \quad x_1, x_2 \geq 0
\end{align*}$$
1.4 Approximation Algorithms

Question 25. Let $G = (V, E)$ be an undirected graph with maximum degree $\Delta$. A dominating set is a subset of vertices $S \subseteq V$ so that for every vertex $u \in V$ there exists a vertex $v \in S$ with $\{u, v\} \in E(G)$. The goal is to find a dominating set as small as possible. Design an approximation algorithm based on greedy for the problem and analyse the quality of its solution.

Question 26. Given an undirected graph $G = (V, E)$, a vertex cover of $G$ is a set of vertices $C \subseteq V$ so that each edge in $G$ is incident to at least one vertex in $C$. A minimum vertex cover is a vertex cover with smallest possible size $|C|$. Consider a greedy approach which iteratively adds the vertex with the highest degree to $C$ and then removes all covered edges from $E$. Find an example that shows that this greedy algorithm does not always find the optimum solution.

Question 27. [CLRS: 35.1-3, this one improves on the previous question and is marked with a \"⋆\" in CLRS] Professor Bündchen proposes the following heuristic to solve the vertex-cover problem. Repeatedly select a vertex of highest degree, and remove all of its incident edges. Give an example to show that the professor’s heuristic does not have an approximation ratio of 2. (*Hint: Try a bipartite graph with vertices of uniform degree on the left and vertices of varying degree on the right.*)

Question 28. How can you implement APPROX-VERTEX-COVER in time $O(V + E)$?

Question 29. [CLRS: Problem 35.3-3] Consider the analysis of GREEDY-SET-COVER (Theorem 35.4). Show that the following weaker form of Theorem 35.4 is trivially true:

$$|C| \leq |C^*| \cdot \max\{|S| : S \in \mathcal{F}\}$$

Question 30. How would you solve an instance of the Vertex-Cover problem using the Greedy Algorithm for the Set-Cover?

Question 31. Consider the problem SUBSET-SUM. Design a simple Greedy algorithm which runs in polynomial-time and achieves an approximation ratio of 2.

Question 32. Consider the algorithm APPROX-SUBSET-SUM from the lecture. Prove formally that for every element $y$, at most $t$, which can be written as a sum of a subset of $\{x_1, x_2, \ldots, x_n\}$, there exists an element $z \in L_n$ (the list in iteration $n$ after the trimming operation), such that

$$\frac{y}{(1 + \delta)^n} \leq z,$$

where $0 < \delta < 1$ is the trimming parameter.

Question 33. [CLRS: 35.3-3] Show how to implement GREEDY-SET-COVER in such a way that it runs in time $O(\sum_{S \in \mathcal{F}} |S|)$. 


Question 34. [CLRS: 35.2-1] Suppose that a complete undirected graph $G = (V,E)$ with at least 3 vertices has a cost function that satisfies the triangle inequality. Prove that $c(u,v) \geq 0$ for all $u,v \in V$.

Question 35. [CLRS: 35.2-5] Suppose that the vertices for an instance of the travelling-salesman problem are points in the plane and that the cost $c(u,v)$ is the euclidean distance between points $u$ and $v$. Show that an optimal tour never crosses itself.

Question 36. Recall the subtour elimination procedure from Lecture 10: In order to eliminate a subtour going through cities in $S$ only, we add the following constraint:

$$\sum_{i \in S, j \notin S} x(\max(i, j), \min(i, j)) \geq 2.$$ 

Prove that adding this constraint to the linear program is equivalent to adding the constraint

$$\sum_{i \in S, j \in S, i < j} x(i, j) \leq |S| - 1.$$ 

Question 37. [CLRS: 35.2-3] Show how in polynomial time we can transform one instance of the travelling-salesman problem into another instance whose cost function satisfies the triangle inequality. The two instances must have the same set of optimal tours. Explain why such a polynomial-time transformation does not contradict the inapproximability result (Theorem 35.3), assuming that $P \neq NP$.

Question 38. Consider the following problem. Given an undirected, connected graph $G = (V,E)$ with non-negative, integral edge capacities $c(u,v)$ for each edge $(u,v) \in E(G)$ and $|E| \geq |V| = n$, the goal is to find a subset $E' \subseteq E$ with $|E'| = n$ so that (i) $E'$ connects all vertices and (ii) $\sum_{e \in E'} c(e)$ is minimized. Either prove that this problem is NP-hard or design a polynomial-time algorithm.

Question 39. Find an example of a graph in the Euclidean space, with as few vertices as possible, so that the optimal TSP tour does not include a minimum spanning tree.

Question 40. [CLRS: 35.4-2] The MAX-CNF satisfiability problem is like the MAX-3-CNF satisfiability problem, except that it does not restrict each clause to have exactly 3 literals. Give a randomized 2-approximation algorithm for the MAX-CNF satisfiability problem.

Question 41. [CLRS: Problem 35-1] Suppose that we are given a set of $n$ objects, where the size $s_i$ of the $i$th object satisfies $0 < s_i < 1$. We wish to pack all the objects into the minimum number of unit-size bins. Each bin can hold any subset of the objects whose total size does not exceed 1.

The first-fit heuristic takes each object in turn and places it into the first bin that can accommodate it. Let $S := \sum_{i=1}^{n} s_i$.

1. Argue that the optimal number of bins required is at least $\lceil S \rceil$. 

5
2. Argue that the first-fit heuristic leaves at most one bin less than half full.

3. Prove that the number of bins used by the first-fit heuristic is never more than \( \lceil 2S \rceil \).

4. Prove an approximation ratio of 2 for the first-fit heuristic.

5. Give an efficient implementation of the first-fit heuristic, and analyse its running time.

**Question 42.** Consider the following algorithm for MAX-CUT on an unweighted, undirected graph \( G = (V, E) \), which can be regarded as an iterative colouring procedure with three colours possible, grey (=unassigned), red (assigned to \( S \)) and blue (assigned to \( V \setminus S \)). Initially, all vertices are grey. Then the algorithm does the following in each step: If there is a grey vertex \( u \) which has more blue than red neighbours colour it **red**, if there is a grey vertex \( u \) which has more red than blue neighbours colour it **blue**. Otherwise, take a grey vertex and colour it arbitrarily. Prove that this algorithm returns a 2-approximation.