上 海 交 通 大 学 试 卷（ A 卷）

（ 2021 至 2022 学年 第 1 学期 ）

班级号______________________ 学号_____________ 姓名 ________

课程名称 ____AI2615 算法设计与分析______________________ 成绩 ________

Question 1 (25 Points). Consider the following variant of Kruskal Algorithm. Does it correctly find a maximum spanning tree on an undirected weighted graph? If so, prove its correctness. If not, give a counterexample $(G=(V,E), w: E \to \mathbb{R}^+)$ showing that it fails to find a maximum spanning tree.

Kruskal_variant$(G=(V,E), w: E \to \mathbb{R}^+)$

- Sort the edges by the weight *descending* order, and initialize $S \leftarrow \emptyset$.
- For each edge e in the order, if adding e to S does not create a cycle, then add it to S.
- Terminate when S forms a spanning tree.

我承诺，我将严格遵守考试纪律。

承诺人：________

题号										
得分										
批阅人(流水阅卷教师签名处)										

Question 2 (25 Points). Suppose we want to allocate m different items to n agents. Let M be the set of items and $N=\{1,\ldots,n\}$ be the set of agents. Each agent i has a *demand set* $N(i) \subseteq M$ which describes the subset of items wanted by agent i. In addition, each agent i can be allocated at most $c(i) \in \mathbb{Z}^+$ items. Design a polynomial time algorithm that allocates the maximum number of demanded items. That is, your algorithm takes $\left(\{N(i), c(i)\}_{i=1,\ldots,n}\right)$ as the input and outputs an allocation $(A_1,\ldots,A_n)$ maximizing $\sum_{i=1}^{n}|A_i|$ subject to that

- $A_i \subseteq N(i) \subseteq M$ and $A_i \cap A_j = \emptyset$ for any $i \neq j$, and
- $|A_i| \leq c(i)$.

Prove the correctness of your algorithm and analyze its running time.

A 卷 总 1 页 第 1 页

Question 3 (25 Points). Given a $m \times n$ non-negative integer matrix $A \in \mathbb{Z}_{\geq 0}^{m \times n}$, suppose we want to walk from the entry $(1,1)$ to the entry (m,n). In each step, you can walk from (i,j) to either $(i+1,j)$ or $(i,j+1)$. Design a polynomial time algorithm that finds a valid walk which maximizes the sum of the visited entries. Prove the correctness of your algorithm and analyze its time complexity.

Question 4 (25 Points). Consider the *Knapsack* problem. You have a set of items $N = \{1, \ldots, n\}$ and a capacity constraint $K \in \mathbb{Z}^+$. Each item i has a *weight* $w_i \in \mathbb{Z}^+$ and a *value* $v_i \in \mathbb{Z}^+$. The objective is to find a subset of items $S \subseteq N$ with maximum total value $\sum_{i \in S} v_i$ subject to the capacity constraint $\sum_{i \in S} w_i \leq K$. You can assume $w_i \leq K$ for each item i.

(a) (10 Points) Prove that Knapsack is NP-hard.

(b) (10 Points) Consider a special case of this problem where we have $w_i = v_i$ for each item i. Prove that the following greedy algorithm gives an 0.5-approximation: iteratively pick an item with maximum $w_i = v_i$ until no more item can be added to S.

(c) (5 Points) Provide a tight example showing that the algorithm in (b) cannot do better than 0.5-approximation.