Beihang University
School of General Engineering

Beijing, March 9th, 2021
Lin Wei and Yida Ding

Data Management and Artificial Intelligence
Lab Class 2

Task 1 ***Graph – Kruskal's Algorithm*** *(20 minutes)*

Implement the Kruskal's algorithm in Python with the graph visualized in Figure 2. For convenience, you can start with the code of graph construction below. The desired output should be: $\{(F, G), (A, B), (C, F), (A, D), (E, F), (A, E)\}$

```
G=nx.Graph()
G.add_edge('A','B',weight=2)
G.add_edge('A','D',weight=8)
G.add_edge('A','E',weight=14)
G.add_edge('D','E',weight=21)
G.add_edge('B','E',weight=25)
G.add_edge('B','C',weight=19)
G.add_edge('E','C',weight=17)
G.add_edge('E','F',weight=13)
G.add_edge('C','F',weight=5)
G.add_edge('C','G',weight=9)
G.add_edge('F','G',weight=1)
```

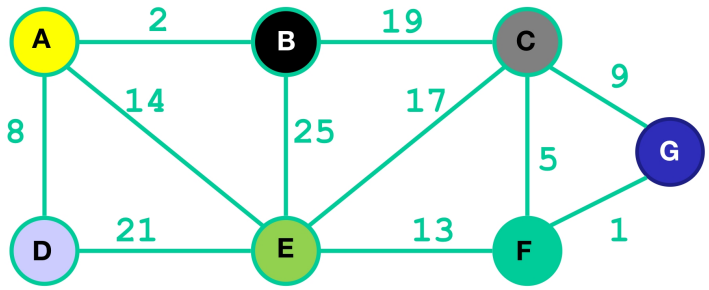

Figure 2: Graph of Task 1

Task 2 ***01-Knapsack Problem*** *(20 minutes)*

In this task you will write a python program to solve the infamous Knapsack Problem. You are provided with a knapsack with limited space and a collection of items with different values and weights. You can only choose to take or not to take an item, but you can't just take part of it. Your task is to maximize the value of items packed into your knapsack without exceeding its total capacity. In DMAI_Lab2_data, we provide three datasets of different sizes. You can start by implementing your algorithm on the smallest data set, then run it on medium and large data sets, and compare their runtime.

A knapsack input contains n + 1 lines. The first line contains two integers, the first is the number of items in the problem, n. The second number is the capacity of the knapsack, K. The remaining lines present the data for each of the items. Each line, from line 2 to line n+1, contains two integers, the item's value v_i followed by its weight w_i.

Task 3 ***Dealing with the toy csv file****(10 minutes)*

In this task, you need to open and deal with a toy csv file "toy.csv". There is some information (names, ages and scores) of four students in the file.

1. Please implement a function Selection(n) which loads the csv file and prints the list of students whose scores are larger than n. For instance, if n = 60, the function will print ["John", "Amy", "Rose"].

2. The GPA of each student depends on the score. Assume that GPA is calculated with the following equation. Please implement a function GPA() which outputs a dictionary with the names of students as the keys and the GPAs as the values, e.g., {Name: GPA}.

$$\text{GPA(n)}=\begin{cases}4, \ if \ n >= 90\\ 3, \ else \ if \ n >= 80\\ 1, \ else \ if \ n >= 60\\ 0, \ else\end{cases}$$

Task 4 ***Working with CSV files****(10 minutes)*

1. Please implement a function **printSelectedColumns** which opens the CSV file airports.csv and prints for each row the values of the columns *name*, *iso_country* and *municipality* on the screen.

2. Please implement a function selectChineseAirports which opens the CSV file airports.csv and prints all columns for rows with Chinese airports (iso_country="CN") on the screen.

Task 5 ***Statistics of CSV files****(10 minutes)*

Please implement a function statistic which opens the CSV file airports.csv and prints the number of unique entries for each column in the file. For instance, column *scheduled service* has two unique entries only: *yes* and *no*.

Task 6 ***Splitting of CSV files****(10 minutes)*

Please implement a function split wich splits the CSV file airports.csv into several csv files, one for each continent, such that each file records the information of airports in that continent, like 'EU.csv', 'AS.csv', and so on.

Task 7 ***Bonus task (once you are finished)***

Implement a function which sorts the entries in *airports.csv* with increasing elevation and write the results to file *airports-sorted.csv*