CS515/Assignment 08

TOPIC: *Linear Programming*

[Q1] Binary Knapsack

Write a function `binary_knapsack` to solve a given binary knapsack problem using Integer Linear Programming. The function should take three inputs:

- capacity : total (carry) capacity of the knapsack
- weights : weight of each item
- costs : cost of each item

The function signature should be as follows:

```
from typing import List

def binary_knapsack(
    capacity:float,
    weights:List[float],
    costs:List[float]
) -> List: ...
```

The function should return a list of binary decisions indicating if a particular item was added to the knapsack or not (0 = not added; 1 = added). Assume that items are numbered starting at zero.

In [ ]:
```
from typing import List

def binary_knapsack(capacity:float, weights:List[float], costs:List[float]):
    ...
```

[Q2] Max Flow with Linear Programming

Write a function `max_flow` that takes a **flow network** with a given *source* and *sink* as input and returns a formulated LP with appropiate coefficients for objective function and constrains (as accepted by the `scipy.optimize.linprog` function). The function signature should be as follows:

```
from typing import Hashable as Node, SupportsFloat as Capacity, List

# output type hint
c, b_ub, b_eq = List, List, List
A_ub, A_eq, bounds = List[List], List[List], List[List]

def max_flow(
    nodes:  set[Node],
    edges:  dict[tuple[Node],  Capacity],
    source: Node,
    sink:   Node,
)-> c, A_ub, b_ub, A_eq, b_eq, bounds: ...
```

The function should return coefficients for objective functions (`c`), inequality constrains (`A_ub` , `b_ub`), equality constrains (`A_eq` , `b_eq`) and bounds (`bounds`). The returned variables should be of `list` type.

In [1]:
```
from typing import Hashable as Node, SupportsFloat as Capacity

def max_flow(
    nodes:  set[Node],
    edges:  dict[tuple[Node],  Capacity],
    source: Node,
    sink:   Node,
): ...
```

Submission

`A08.py` containing `binary_knapsack` and `max_flow`
