Student Name

CS 480 Fall 2024 Programming Assignment #02

Due: **Sunday, October 27, 2024 at 11:59 PM CST**
Points: **20**

Instructions:

1. Place **all your deliverables (as described below) into a single ZIP** file named:

`LastName_FirstName_CS480_Programming02.zip`

2. Submit it to Blackboard Assignments section before the due date. **No late submissions will be accepted**.

Objectives:

1. (20 points) Implement and evaluate a constraint satisfaction problem algorithm.

Input data files:

You are provided two CSV (comma separated values) files (see Programming Assignment #02 folder in Blackboard):

- `driving2.csv` - with **driving distances** between state capitals.
- `parks.csv` - with **numbers of National Parks** per state.
- `zones.csv` - with **zone IDs** assigned to each state.

You **CANNOT** modify nor rename input data files. Rows and columns in those files represent individual state data (state labels/names are in the first row and column). Numerical data in both files is either:

- a non-negative integer corresponding to the distance between two state capitals,
- negative integer -1 indicating that there is no direct "road" (no edge on the graph below) between two state capitals.

Deliverables:

Your submission should include:

- Python code file(s). Your py file should be named:

`cs480_P02_AXXXXXXXX.py`

where `AXXXXXXXX` is your IIT A number (this is REQUIRED!). If your solution uses multiple files, makes sure that the main (the one that will be run to solve the problem) is named that way and others include your IIT A number in their names as well.

Problem description:

Consider the graph presented below (fig. 1). Each node represents a single state (or the District of Columbia (DC)). If two states are neighbors, there is an edge between them. States are grouped into **zones Z1 through Z12** an **you can only travel westward** (this is not meant to be realistic!). Each node is also assigned a number representing the number of national parks in the corresponding state. For example: WA 3 means that there is three (3) national parks within this state.

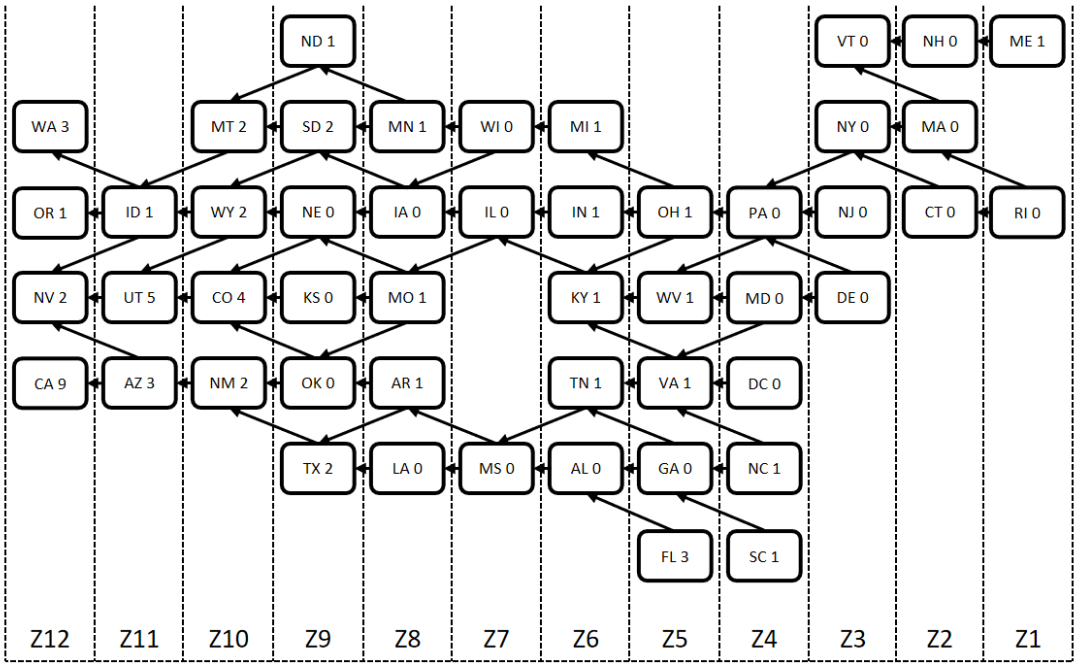

Figure 1: A graph representing all 48 contiguous US states and District of Columbia.

Your task is to use a Constraint Satisfaction Problem solver algorithm (backtracking) to **plan a road trip across the US that starts in <u>ANY state</u> and ends in one of the zone Z12 states (CA, NV, OR, or WA)**. Your road trip path MUST consist of exactly one state per each zone between (and including) your initial zone (where your initial state is) and zone Z12.

For example, if you start in TN (Zone Z6), you will need to visit (assign a value to each zone variable along the way) EXACTLY one state in zones Z6, Z7, Z8, Z9, Z10, Z11, and, Z12. Visited states must constitute a valid path (in other words: there need to be roads between each state on the path).

There is an additional constraint your road trip: the number of national parks you want to visit (`NO_OF_PARKS`).

Assumptions:

- Each zone Z1,…, Z12 is a variable with corresponding state labels as possible values.
  - For example, possible values for zone Z2 are CT, MA, and NH.

- PARKS_VISITED will be a separate variable. You will **use it to keep track of the number of visited national parks along your assignment path**:
  - you can treat is as the **last** variable to be assigned a value (a total number of parks visited along the path) tp after all the zone variables, or
  - you can implement it as a part of IF statement that decides if the path / assignment is complete or not.
- **Visiting a state** is equivalent to **visiting ALL national parks within that state**,
- You can start at any state / in any zone. Your initial state/zone choice will dictate how many zone variables you will need to involve.
  - For example: if your initial state is TN (Zone Z6), your zone variable list is going to be Z6, Z7, Z8, Z9, Z10, Z11, Z12.
- An assignment is **complete** if:
  - all zone variables between (and including) the initial zone and zone Z12 have a valid value assigned to it,
  - PARKS_VISITED >= `NO_OF_PARKS`
- The range / domain of possible values for a given zone variable ZK is always going to be constrained by the value assigned to a previous zone.
  - For example: if Z4 = MD, possible values for Z5 become: VA, WV,
- Use the Backtracking algorithm from your textbook (see pseudocode below).
- Assume that edge weights represent **driving distances between state capitals**.

Your program should:

- Accept two (2) command line arguments so your code could be executed with

```
python cs480_P02_AXXXXXXXX.py INITIAL NO_OF_PARKS
```

where:

- `cs480_P02_AXXXXXXXX.py` is your python code file name,
- `INITIAL` is the label/name of the initial state,
- `NO_OF_PARKS` minimum (it can be more!) number of national parks you intend to visit.

Example:

```
python cs480_P02_A11111111.py MD 5
```

If the number of arguments provided is NOT two (none, one, or more than two), your program should display the following error message:

```
ERROR: Not enough or too many input arguments.
```

and exit.

- Load and process both input data files provided (assume that input data files are ALWAYS in the same folder as your code - this is REQUIRED!).
- Run Backtracking algorithm to find a path / assignment between `INITIAL` and zone Z12 states.

- Report results on screen in the following format:

```
Last Name, First Name, AXXXXXXXX solution:
Initial state: INITIAL
Minimum number of parks: NO_OF_PARKS

Solution path: STATE1, STATE2, STATE3, …, STATEN-1, STATEN
Number of states on a path: X1
Path cost: Y1
Number of national parks visited: N1
```

where:

- `AXXXXXXXX` is your IIT A number,
- `INITIAL` is the label/name of the initial state,
- `NO_OF_PARKS` is the **minimum** number of national parks to be visited,
- `STATE1, STATE2, STATE3, …, STATEN-1, STATEN` is a solution represented as a list of visited states
- `Y1` total driving distance (assume that visiting a state capital counts for visiting all national parks within a given state, if any),
- `N1` is the **final** number of national parks visited (`N1 >= NO_OF_PARKS`)

If no path is found replace appropriate information with:

```
Solution path: FAILURE: NO PATH FOUND
Number of states on a path: 0
Path cost: 0
Number of national parks visited: 0
```

Backtracking algorithm pseudocode:

```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
  return BACKTRACK(csp, { })

function BACKTRACK(csp, assignment) returns a solution or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp, assignment)
  for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
      if value is consistent with assignment then
        add {var = value} to assignment
        inferences ← INFERENCE(csp, var, assignment)
        if inferences ≠ failure then
          add inferences to csp
          result ← BACKTRACK(csp, assignment)
          if result ≠ failure then return result
          remove inferences from csp
        remove {var = value} from assignment
  return failure
```

Notes:

- SELECT-UNASSIGNED-VARIABLE should always pick next zone variable. If the last assigned variable was Z4, next one should be Z5,
- ORDER-DOMAIN-VALUES should order all POSSIBLE domain values (next states) **alphabetically**,
- INFERENCE outcome is going to be based on previous zone variable assignment.