You have a car.

A car is made of an engine, a transmission, a tire, and a roof.

Engine choice: *engines.txt* file

Tire choices: *tires.txt* file

Transmission choices: *transmissions.txt* file

Roof choices: Sunroof, Moonroof, No roof

Engine Tire Transmission Roof

You have a car like **START STATE** (EFI,Danlop,AT,Noroof)

You want to have a car like **GOAL STATE** (V12,Pirelli,CVT,Sunroof)

Each year you can change only 1 component of the car which results in a way that makes a valid car model i.e **You can’t change your component in a way which makes your car invalid at any point of time**. The valid car models are given in the valid\_book.csv file.

You want to have your dream car as soon as possible i.e Minimize the years after which you get your desired car..

How to calculate the cost between two states: Number of mismatched components.

Example: **Δ**E = 3 for (EFI,Danlop,AT,Noroof) and (EFI,Danlop,AT,Sunroof) as only the roof state mismatches.

If at any point, you arrive at an invalid state, you can’t go any further from that one, just discard that state.

Probability function for simulated annealing: e^(ΔE/t) **[Note: this function is required only when ΔE is negative]**

Here, ΔE = next\_node.val - current\_node.val

Y = year passed aka BFS level

t = 1 / Y

Override \_\_hash\_\_ method of Car class [Ref: <https://stackoverflow.com/questions/2909106/whats-a-correct-and-good-way-to-implement-hash>

]

Probability modeling:

random.uniform(0, 1) <= e

