**NYC Itinerary Planner**

**Part I. Problem and plan**

1. **Problem Description**

The first destination which out of state students and international students visit is New York City. But as students, we face a lot of **constraints.** A few of which are time, money and places to stay in the city. Along with these strict constraints, students also have other conditions. For example, visiting Statue of Liberty, eating at Joe’s Pizzeria are few of the other conditions.

To solve this problem, we take the help of IDP, where we can develop a knowledge base system and add our constraints to it. For, the scope of this project, we are building a system ~~specifically for Stony Brook students~~ with the target being **creating an itinerary for an NYC trip.**

**INPUTS**

* **User Input:**
  + Total Number of Hours
  + Start Point and End Point
  + Total Money the user is willing to spend
  + Mandatory places to visit
* **Inputs to the build the Knowledge Base System**
  + ~~Google Places API/TripAdvisor API to get destinations in New York City along with the entry fee.~~
  + ~~Google Route API or travel time approximation based on distances to get distances along with cost.~~
  + TripAdvisor Top Destinations to visit in NYC list.
  + TripAdvisor Location Search API to get Location Id of a place.
  + TripAdvisor Location Details API to get Actual Location based information.

**OUTPUTS**

* An ordered itinerary based on constraints provided ~~along with the ratings and reviews for these places.~~

Ideally, I would want to create an interface like what was shown in the IDP demo for course selection where I can just select or unselect places of interest and/or run the solver again, trying for a different solution/itinerary.

1. **State of the Art Systems**

Most work done in this area uses Machine Learning and solves different subset of the problems. There are ML based systems build which focus on accounting for situational information such as temperature and precipitation (*Real-Time Context-Aware Recommendation System for Tourism*, 2023) and these kind of systems can be attached to vehicles to make real time change in plans.

There is also generic work done using ML like Tour Spot Recommendation System via Content-Based Filtering (*Tour Spot Recommendation System via Content-Based Filtering*, 2023)which is not really our are of interest. We want to model these problems and solve it using an SMT solver rather than ML.

There is a language defined specifically to handle such planning problems called PDDL+ . PDDL+ (*PDDL+ : Modelling Continuous Time-Dependent Effects Maria Fox and Derek Long*, n.d.) is an extension of the Planning Domain Definition Language (PDDL), which is a standard language used to specify problems and domains for automated planning systems.

Itinerary planning is a complex task that involves both discrete decisions (like choosing which places to visit) and continuous variables (like the time spent at each place or the distance between places). PDDL+ can help model these complex problems very well. PDDL 3 (*PDDL3*, n.d.) also helps model soft constraints very well and these could be used as places that a user would really want to visit but it is not mandatory in our use case.

In the paper (*A Temporal Constraint Based Planning Approach for City Tour and Travel Plan Generation*, n.d.) the authors have proposed schemes for specification and processing of temporal constraints in a Hierarchical Task Network (HTN) planner. They have redefined the basic PDDL3.0 temporal operators as a restricted subset of Metric Interval Temporal Logic (MITL) family. The planner has been used to devise city tour and travel plans under temporal constraints and the generated plans have been rendered through Google-map based navigation interface.

Using PDDL+ or a flavor of it like in HTN seems to be the optimal way to go about this problem as PDDL is dedicated to modeling planning based problems. They also have a very detailed guide on how to use it with VSCode (*PDDL*, 2017) . ~~We will however be using the IDP instead as IDP also gives us the opportunity to model such problems even though it is not specifically built for our system unlike PDDL.~~

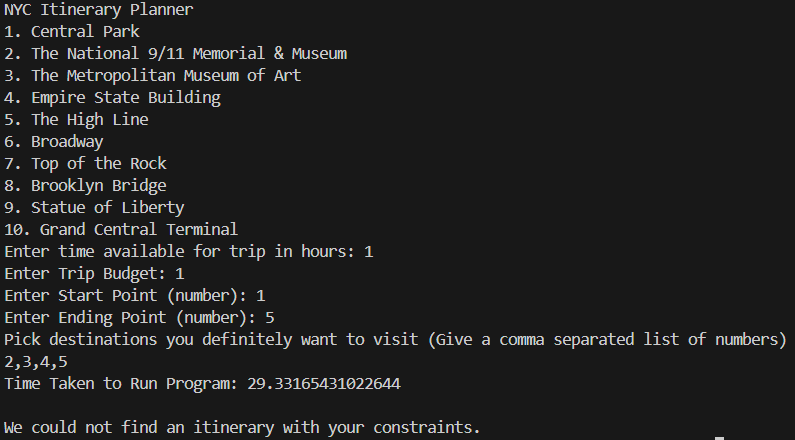
We will be using Clingo instead of IDP Z3 as Clingo seems much easier to both install and use, and it has much better support online as IDP Z3 is relatively new.

1. **Tasks and Sub-Tasks**

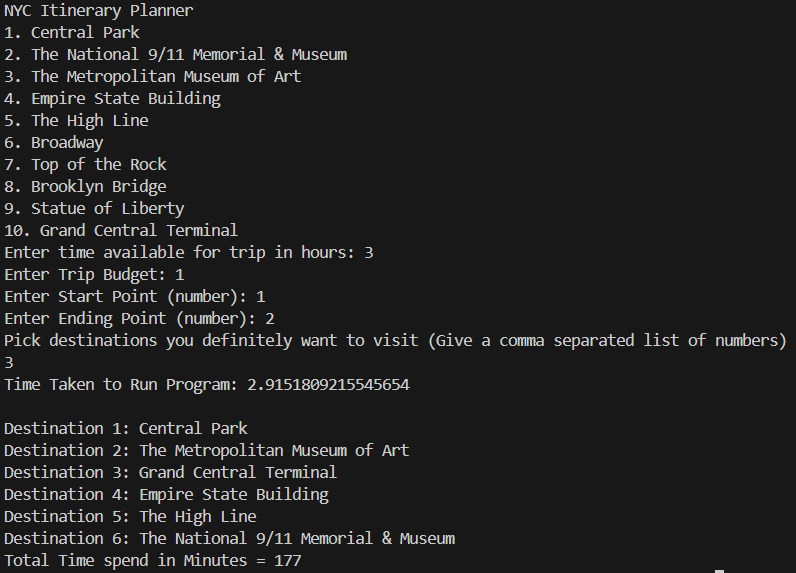
I will be building the system from scratch will take inspiration mainly from the IDP demos. I plan on using either IDP Z3 or IDP and will come to a conclusion while doing assignment 2.

1. **Code the MVP specifications**
   1. Start off by providing the simpler constraints like money and time and associate each event in the itinerary with a cost in terms of money as well as time.
   2. The start and end destination can be generic, but we add a fact initially with Stony Brook being the start and end point.
   3. The MVP can have a fixed total time and money and travel costs can be ignored or kept constant.
2. **Expand the MVP**
   1. Check for viability of using Google APIs like Place and Route or try tripadvisor APIs to build the knowledge base.
   2. Build the one time knowledge base for places of interest in the city with their ratings by either using APIs or manually pulling in data to expand the MVP to an alpha version.
   3. Build a simple GUI as seen in the IDP demo for course selection.
3. **Extension of Project based on Pace**
   1. If Route API can be used (we have cost restrictions, as well as implementation times), we can use those calls to get travel related costs or we come up with custom heuristics based on latitude, longitude to come up with travel time and cost.
   2. Split destinations based on type, such as restaurant, monument, museum, art performances, etc, to add more specific constraints.
   3. Work on the visualization aspects further.
4. **Project Plan**
5. **Feb 26th – March 10th** 
   1. Take a look at IDP3 and IDP Z3, check which would be more ideal for my case (ease of implementing as well as feature availability).
   2. Setup IDP3 / IDP Z3 on my local machine and start playing around.
   3. Implement the various constraints and get the MVP ready.
6. **March 11th to April 19th** 
   1. Expand the MVP and look at the viability of APIs.
   2. Come up with either the heuristics or use the APIs based on time remaining.
   3. Get a basic GUI ready along with a knowledge base with a good amount of data.
   4. Test the correctness of the constraints with the knowledge base and ask for further feedback.
7. **April 20th to May 3rd**
   1. Improve the GUI further.
   2. Add more features like categorization and custom start and end points.
8. **Implementation, Testing and Evaluation**

* A brief guide on the code implementation has been added to the readme.docx file.
* Parsed and obtained 60 destinations to build a complete graph by specifying all the edges in the clingo file. This did not work out as the program was simply not efficient enough and took too long to even read the file. Hence, we used the top 10 destinations to check if the code works as expected.
* Following images show correctness. **We verify by checking if total time is less than the time user has inputted** (Note: User Input is in hours and Output is in minutes), if start and end points are the same as well as if all mandatory destinations are covered.



* The above constraints can’t be satisfied as the time taken to visit all the mandatory destinations is more than the time provided by the user.



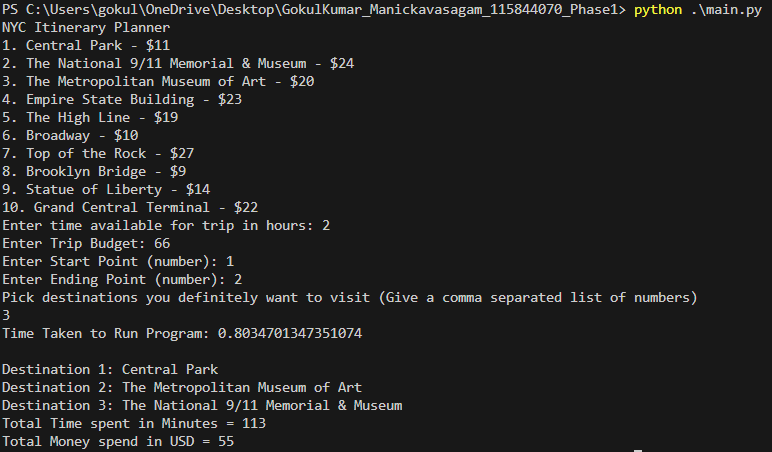
* The above image shows a successful case and it satisfies all the constraints.
* We do have code for measuring time as visible from the above images, these will be added for the final submission/ presentation.

**Observations:**

* I initially tried to run the code with a huge knowledge base of 3600+ lines and the code took too long to be even read. So, we had to cut down on the number of destinations.
* **#sum** behaved exactly as discussed in class and would only sum the unique numbers, I had to add an extra variable to make each component of the sum distinct.
* Parsing the webpages, parsing the inputs and outputs took considerably longer than I expected, especially to make the final output neat and clean.
* We can use strings directly instead of variables in clingo and this makes the clingo knowledge base much more readable than using A, B, C, etc.

**Features Tests and Observations added Post MVP**

* Added trip budget (money) as a constraint.
  + Used a randomize function to generate random entry fee for each destination.

****

* + Predicate node/1 changed to node/2 as it now includes name of destination and entry fee.
  + Made changes across driver code to change how it generates the new clingo file and for output parsing.
* Compared execution time with changing input (Changed time available)
  + All other inputs were kept constant.
  + The testcase with 2 hours as the input time took the longest as it was the toughest constraint to solve. There is only one solution and hence it took ~13 seconds even on multiple retries.
  + The time recorded includes **only time to run the already generated clingo file**. It does not include the time for parsing the output.
* One more general observation was that, tougher the problem constraints, longer the exection time. Satisfiable problems takes lesser time than non-satisfiable ones for similar inputs.

1. **References**
2. PDDL. (2017, April 15). Visual Studio Marketplace. Retrieved March 15, 2024, from https://marketplace.visualstudio.com/items?itemName=jan-dolejsi.pddl
3. PDDL3. (n.d.). Plan Constraints and Preferences in PDDL3. Retrieved March 15, 2024, from https://planning.wiki/\_citedpapers/pddl32005.pdf
4. PDDL+ : Modelling Continuous Time-dependent Effects Maria Fox and Derek Long. (n.d.). Planning.wiki - The AI Planning & PDDL Wiki. Retrieved March 15, 2024, from https://planning.wiki/\_citedpapers/pddlplus2003.pdf
5. Real-Time Context-Aware Recommendation System for Tourism. (2023, April 2). NCBI. Retrieved March 15, 2024, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10098936/
6. A temporal constraint based planning approach for city tour and travel plan generation. (n.d.). IEEE. Retrieved March 15, 2024, from https://ieeexplore.ieee.org/document/6481849/
7. Tour Spot Recommendation System via Content-Based Filtering. (2023, January 6). IEEE. Retrieved March 15, 2024, from https://ieeexplore.ieee.org/document/10016820/