CSCI 360 Project 2: MasterChef Team Formation

Spring 2020

Due: March 25th, 2020

1 Overview

This is a programming assignment. You will be provided sample inputs and outputs (see below). Please understand that the goal of the samples is to check that you can correctly parse the problem definitions, and generate a correctly formatted output. The samples are very simple, and it should not be assumed that if your program works on the samples it will work on all test cases. There will be more complex test cases and it is **your task to make sure that your program will work correctly on any valid input**. You are encouraged to **try your own test cases** to check how your program would behave in other special/complex case that you might think of. This homework will be graded via an automated script, and thus your outputs should match the specified format **exactly**. The output format is a simple `.txt` file, and examples are provided in the `Input/Output format` section. You should upload and test your code on Vocareum, and submit your final scripts there.

2 Project Description

This project will provide you with an opportunity to practice what you have learned about the Game Playing in the class. In a typical zero-sum two-player game, players are generally competing for a certain common resource, and their gain is a function of their share of the resource. Often players have other challenges such as satisfying other constraints on other personal resources such as time, energy or computational power. In this homework, we will solve the MasterChef team forming challenge, a game that captures the nature of a zero-sum two-player game that can be tackled with adversarial search algorithms.

2.1 MasterChef Team Formation

MasterChef is a competitive cooking show where amateur and home chefs compete for the title of MasterChef while going through a series of challenges. The final winner wins $250,000, a MasterChef trophy, and the title of MasterChef. One type of challenges is MasterChef's Team Challenge. During the team challenge, the cooks are split into 2 teams by either team captains or the judges, and they prepare meals as a team. Diners taste both teams' meals and vote for their favorite. The team with more votes wins the team challenge and advances, while the losing team has to participate in the Pressure Test or face elimination based on the teams' performance.

In this project, you are one of the two captains in charge of picking team members to form the "best" team possible. We will refer to the other captain as Bob.

2.2 Rules of the game

You and Bob are chosen to be the captains of the MasterChef Team Challenge of Season 11. Your team is Team A, and Bob's is Team B. Unlike the regular Team Challenge, each team will be composed of exactly *5* contestants. So, the game is for you to form a Dream team of 5 members. Each of you is going to take a turn and pick a contestant from the same pool of contestants. The process ends after your team and Bob's team both have exactly 5 members (i.e., 10 distinct contestants are picked in total). We assume the followings:

- both you and Bob have exactly the same information about the contestants
- both of you want to form the best team possible
- each pick is made in an open setting, so both you know each other's decision as soon as it is made
- there are always an even number of already-picked contestants in the input file. This means, the number of chosen contestants (marked 1 and 2) at the given state of the game is always even.
- you are always the one who makes the pick before Bob at each round. **Your next pick (given the state of the game provided in the input file) is the output of your program**.
- each member must belong to exactly one team
- no new member is added to the pool during the process

2.3 Power of a team

The strength of a team is measured by its "Power". Each selected member adds to the Power of his/her team according to his/her cooking capacity ("Capacity"), and how happy he/she is to work under the particular captainship, ("Happy"). Before the team formation starts, each member has announced their happiness about the captainship of you and Bob as Happy_A and Happy_B, respectively.

Another factor that affects a team's Power is how diverse the members are. This attribute is called "Diversity". In this project, it is modelled as a bonus point of 120 which is given to a team whose contestants' last digit of the IDs are all different. For example, if $Mem(A) = \{12000, 10101, 20398\}$ and $Mem(B) = \{12110, 12392, 29237\}$, then $D_A = 120$ and $D_B = 120$. Note that all of the scenarios where (i) both Team A and B get the Diversity bonus, (ii) either one gets the bonus, or (iii) neither gets the bonus are possible. Thus, $D_t = 120$ if all members of Team t has different last digit of IDs, and $D_t = 0$ otherwise.

The Power of a Team t, $P(t)$, is therefore calculated using the following equation:

$$P(t) = D_t + \sum_{i \in Mem(t)} H_{i,t} * C_i$$

where

- $Mem(t)$ is the set of members in Team t. Remember $\mid Mem(t) \mid = 5, t \in \{A, B\}$
- C_i is the cooking capacity of a member
- $H_{i,t}$ is the happiness of the i-th member about the captainship of Team t
- D_t is the team's diversity point, given its members $Mem(t)$

2.4 Advantage

Your advantage A is calculated as the difference between the Powers of your team and Bob's:

$$\begin{aligned} A &= P_A - P_B \\ &= (D_A + \sum_{i \in Mem(A)} H_{i,A} * C_i) - (D_B + \sum_{j \in Mem(B)} H_{j,B} * C_j) \end{aligned}$$

Note that Team B's advantage against your team is A.

2.5 Goal

Given a pool of contestants and the information about their IDs, cooking capacities, happiness over the captains, you and Bob you want to maximize your advantage A while Bob wants to minimize A. You are going to solve this problem by using an adversarial search algorithm, namely the minimax algorithm with alpha-beta pruning as studied in class.

2.6 Data

Each contestant's data contains the following attributes:

- ID: 5 digits integer indicating the contestant's unique ID. 32-bit integer. No two contestants have the same ID in a test case.
- Capacity: a value in range [0.0, 200.0] indicating the cooking capacity of the contestant. 64-bit Double
- Happy_A, Happy_B: the contestant's happiness about the captainships of Team A and Team B, respectively. 64-bit double in [0.0, 1.0]
- Current pick state as exactly one of the following integers:
  - 0: has not been picked yet (i.e. available),
  - 1: has already picked by you, or
  - 2: has already picked by Bob

These attributes will appear in the input file in the order as listed above, separated by commas. Note that the end of a line is always a line feed (LF) character (ASCII `\x0a`, a.k.a. `\n` in many programming languages).

An example of a contestant's data is:

ID	Capacity	Happy_A	Happy_B	Pick State
64504	193.537866	0.239586	0.572906	0

2.7 Input Format

The inputs will always be valid `.txt` file of the following format.

- Line 1: Number of contestants in the pool
- Line 2: Algorithm to use. Either
  - `minimax` for minimax algorithm, or
  - `ab` for alpha-beta pruning
- Each of the rest of the lines contains a contestant's attribute as described in `Data` section.

Here is an example input file, `input.txt`:

```
14
minimax
75201,92.192554,0.822285,0.134675,0
64504,193.537866,0.239586,0.572906,0
83601,10.631835,0.547191,0.251238,1
87705,111.105311,0.931969,0.653667,1
25202,23.053795,0.075295,0.993499,0
39202,87.595928,0.208003,0.255219,0
42904,168.518783,0.426926,0.432817,0
12703,75.011463,0.456201,0.037517,2
55502,168.823333,0.127049,0.159396,0
82301,82.524738,0.755311,0.406141,0
46902,142.316246,0.511181,0.217463,1
98403,83.132871,0.236225,0.612434,2
79802,118.762381,0.880846,0.579115,2
88903,187.685198,0.277158,0.679969,0
```

Listing 1: Example input file.

In this example input file, there are `14` (Line 1) contestants, and you are asked to use `minimax`(Line 2) to form the best team possible. The first contestant (ID: `75201`) has the cooking capacity of `92.192554`, is happy about the captainships of you and Bob's by `0.822285` vs. `0.134675`, respectively, and has not been picked yet (`0`).

2.8 Output Format

Your program should generate an output file called `output.txt` that has a single line indicating the ID of the contestant you are going to pick. The end of line is again an LF (ASCII \x0a). An example is as follows:

```
88903
```

We provide 10 pairs of input and output files as sample test cases. Refer to the "resource/asnlib/publicdata" directory in Vocareum.

2.9 Tie Breaking

If there is a tie (multiple contestants are equally good to choose), please always choose the one with the *smallest* ID.

2.10 Branching

Please iterate through nodes in *ascending* order of IDs.

3 Project Rules

1. The main file of your solution should be named `project2cs360s2020.*`, where * is `.cpp`, `.java`, or `.py`
2. Your program must finish each test case within 3 minutes on Vocareum. Although upper bounds have been provided, test cases are created with this time limit in mind, and you don't need to expect the maximum size of all parameters in one test case
3. You may use **ONLY C++, Java, Python 2.7, 3.x,** and **any libraries that are included in Vocareum**
4. C++ and Java files will be compiled using the following commands (you may confirm this work by submitting your code for testing on the small submission test cases):
   - C++:
     - `g++ -o project2cs360s2020 project2cs360s2020.cpp`
     - `timeout 180s ./project2cs360s2020`
   - Java:
     - `javac project2cs360s2020.java`
     - `timeout 180s java project2cs360s2020`
   - Python:
     - python `project2cs360s2020.py`
5. Projects must be submitted through Vocareum. Please only upload your program code. Don't create any subfolder or upload any other files. Please refer to this[1] to get started
6. You are strongly encouraged to submit your program 24 hours ahead of the deadline to avoid any last-minute submission issues on Vocareum
7. There is a grace period of 24 hours during which time your submission will be accepted, but with a 20% penalty. After 24 hours, your submission will not be accepted

4 Grading

The grading script will

- create an "input.txt" file, and delete any old "output.txt" file
- run your program, which should be named "project2cs360s2020.*", where * is ".cpp", ".java", or ".py"
- check your "output.txt" file

Note that if your code does not compile, or somehow fails to load and parse input.txt, or writes an incorrectly formatted output.txt, or no output.txt at all, or OuTpUt.TxT, *you will get zero points.* Anything you write to stdout or stderr will be ignored and is ok to leave in the code you submit (but it will likely slow you down). Please test your program with the provided sample files to avoid this.

5 Academic Honesty and Integrity

All project material is checked vigorously for dishonesty. Violations of academic honesty are forwarded to the Office of Student Judicial Affairs. To be safe, you are urged to err on the side of caution. Do not copy work from another source. Sanctions for dishonesty will be reflected in your permanent record and will negatively impact your future success. As a general guide:

- Do not copy code or written material from another student. Do not collaborate on this project. The project is to be solved individually. Do not copy code off the web. This is easier to detect than you may think. - Do not share any test cases that you may create to check your program's behavior in more complex scenarios with other students. Do not copy code from past students. - Do ask the CPs/TAs/Instructor in case you are unsure about if certain actions constitute dishonesty. It is better to be safe than sorry.

[1]`<http://help.vocareum.com/article/30-getting-started-students>`