CSCI 561 – Foundations of Artificial Intelligence

Instructor: Dr. K. Narayanaswamy

Assignment 2 – Greedy and A* Search Algorithms

Due: 03/01/2013 11:59:59pm

Netflix Prize is a competition for the best algorithm to predict users' rating for their unseen movies. This contest is held by Netflix, which is an American company providing internet streaming media to public. Figure 1 shows the contest: Netflix has 17,700 movies in its database, and 480,000 user members (data from 2009). Each user only watched several movies from the huge database, and for these movies, he/she would give a rating ranging from 1 to 5. Given these ratings, participated teams are asked to predict ratings for those unseen movies of each user, i.e. filling the blanks with potential rating values.

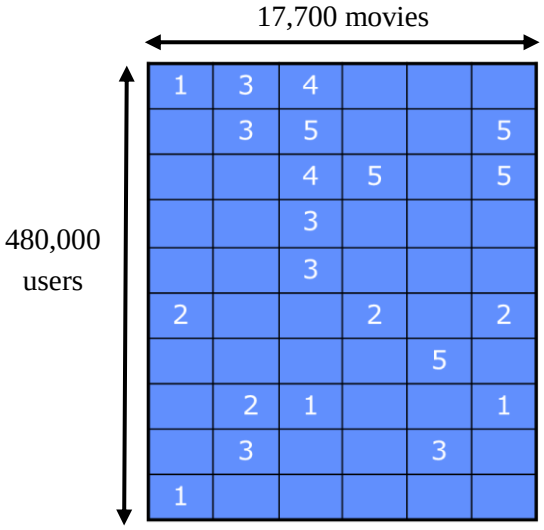

Figure 1: Movie rating data

One key problem behind this contest is how to measure the **similarity** between two movies as objective/realistic as possible. For instance, in Figure 1, user1 only watched movies 1, 2 and 3. Suppose we developed a similarity measurement algorithm and found that movie 4 has a high similarity with movie 3, then probably user1 will rate movie 4 with a value similar to the rating he/she gave to movie 3. We can fill all the missing rating blanks in this manner.

In this assignment, we will provide a simplified version of this problem and you are asked to use **greedy and A* algorithms** to find the **dissimilarity** of two movies. The smaller the dissimilarity value is, the more similar those two movies are.

In Figure 2, an **indicator matrix** is shown, with **30 users** as rows and **12 movies** as columns. If user A watched movie B, then the corresponding entry (A, B) in the indicator matrix is 1; otherwise

it is 0 (In the figure, we left 0 as blank).

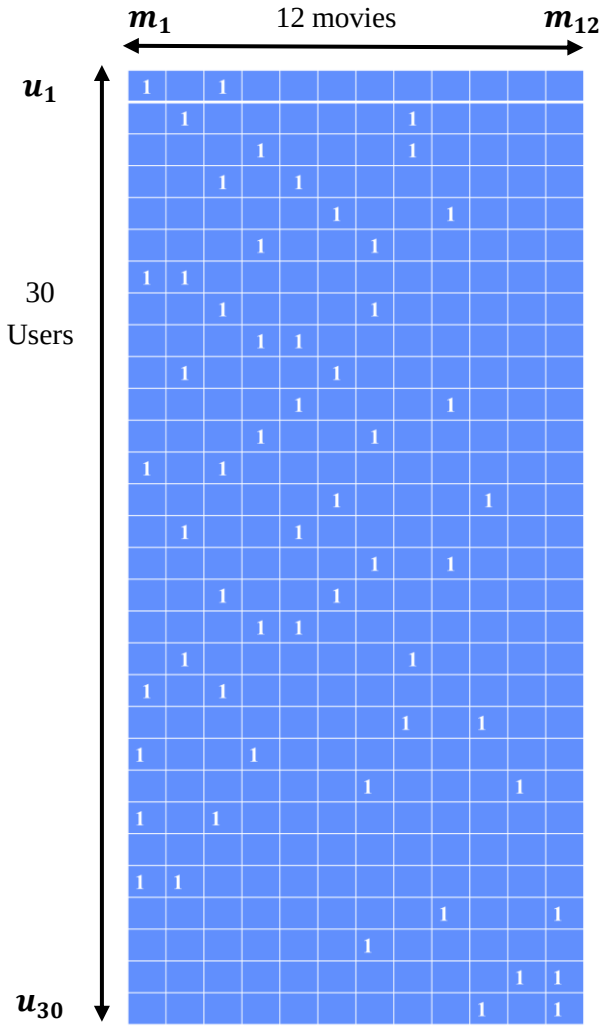

Figure 2: Indicator matrix between users and movies

For two movies m_i & m_j, **if there is at least one user who watches both of them**, then they are defined to be **neighbors**. For two neighboring movies m_i & m_j, the dissimilarity $d_{m_i m_j}$ between them is defined as: $\{(\boldsymbol{i}+\boldsymbol{j})\%\boldsymbol{T_0}+\boldsymbol{1}\}\cdot\{\boldsymbol{T_0}-\boldsymbol{V_{m_i}}\cdot\boldsymbol{V_{m_j}}\}$, where % is the modulo operator, i, j are movie indexes, V_{m_i}& V_{m_j} are corresponding column vectors of m_i & m_j in the indicator matrix, $\boldsymbol{V_{m_i}}\cdot\boldsymbol{V_{m_j}}$ is dot product of two vectors V_{m_i} and V_{m_j}, and T_0 is a constant. In our case, $T_0 = 5$. This definition doesn't apply to movies without a common user, while you are asked to compute dissimilarity between them.

Take three movies A, B and C as an example, suppose A and B are neighbors, B and C are neighbors, but A and C are **NOT** neighbors. Although we can't compute dissimilarity between A and C using the above defined similarity formula, dissimilarity can be propagated from A to C through B. The **cumulative dissimilarity** $d_{AB}+d_{BC}$ along the dissimilarity propagating path $\boldsymbol{A}-B-\boldsymbol{C}$ is defined as one potential dissimilarity between A and C. There may be other dissimilarity propagating paths from A to C; i.g. $\boldsymbol{A}-D-E-\boldsymbol{C}$, $\boldsymbol{A}-F-H-I-\boldsymbol{C}$, and each one has an associated dissimilarity value. The smallest cumulative dissimilarity among all propagating paths is said to be the true dissimilarity between A and C. This principle for

dissimilarity measurement applies to any two non-neighboring movies.

Your task is to implement the following algorithms to search the dissimilarity between $\boldsymbol{m_1}$ and $\boldsymbol{m_{12}}$:

Heuristic dissimilarity values between movie $m_i(i = 1,2,\dots,11)$ and movie m_{12} are provided in Table 1.

Table 1: Heuristic Dissimilarity

Movie	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}
Estimated Dissimilarity	12	10	9	10	7	6	7	6	4	4	3	0

1. Greedy search

Using **heuristic dissimilarity** for the heuristic

2. A* search

Using **cumulative dissimilarity** as cost and **heuristic dissimilarity** for the heuristic

Input: There is no input for this program. Your program should build a tree/graph using the information above, which will serve as the search space for the Greedy and A* searches. You should not expect any command line argument(s). When expanding a node and there are multiple neighbors of equal cost, you should break tie by **giving higher priority to movies with smaller indexing numbers**.

Output: For each algorithm, you program will print the following information.

- 1st Line: List the names of the movies (m_i, separated by commas) in the order traversed by the search. Each movie is printed only once on the movie's first visit (precisely when the corresponding node is **expanded** and not when it is revisited due to backtracking). The search should stop at m_{12}, which is the last node to be printed.
- 2rd Line: Output the dissimilarity between m_1 and m_{12}
- 3th Line: Output the dissimilarity propagating path from m_1 to m_{12}, including m_1 and m_{12}

Your program will print the output for both greedy search and A* search at the same time, one after the other. There is no algorithm selection interface in this assignment.

Sample Output: Note, these are not correct outputs. They are only shown to give you a sense of what the output format should look like.

Greedy Search (A* Search):

Traversal sequence:

$m_1,\ m_5,\ m_3,\ m_7,\ m_9,\ m_{12}$

Dissimilarity: 20

Propagating path:

$m_1,\ m_5,\ m_3,\ m_{12}$

Program Specifications:

1) Your program must be written in either Java or C++.

2) Your program MUST compile and run on aludra.usc.edu
3) Your program might be deemed incorrect if its output does not follow the format described in the sample output.
4) The answers output by your program should be calculated from your implementation of the search algorithms. Outputting hard-coded answers is considered cheating and your grade on this assignment will be 0.

Question: In addition to the program and the file Readme.txt, you should also submit another PDF file named **Answer.pdf**, which contains the answer to the following question.

1) What are the advantages and disadvantages of each search algorithm (Greedy and A*)? Draw examples from the output of your program.

Grading Policy: (Total Points: 100)

Programming (80 pts):

Your program must implement the two search algorithms.

1) Correct traversal for Greedy search: 20 points
2) Correct dissimilarity value retrieved from Greedy search: 5 points
3) Correct propagating path for Greedy search: 15 points
4) Correct traversal for A* search: 20 points
5) Correct dissimilarity value retrieved from A* search: 5 points
6) Correct propagating path for A* search: 15 points

Answer.pdf (10 pts):

1) Answer to Question#1 comparing Greedy and A* searches (10 pts)

Readme.txt (10 pts):

1) A brief description of the program structure and any other issues you think will be helpful for the grader to understand your program. (5 pts)
2) Instructions on how to compile and execute your code. (5 pts)
3) **Please include your name, student ID and email address on the top.**
4) You must submit a program in order to get any credit for the Readme.txt. In short, if you submit ONLY a Readme.txt file you will get 0.

Submission Guidelines: Your program files will all be submitted via blackboard. You MUST follow the guidelines below. Failure to do so will incur a -25 point penalty.

1) Compress and zip ALL your homework files (this includes the Readme.txt and Answers.pdf and all source files) into **one** .zip file. Note that only .zip file extensions are allowed. Other compression extensions such as .tar, rar, or 7z will **NOT** be accepted.
2) Name your zip file as follows: firstname_lastname.zip. For example, John_Smith.zip would be a correct file name.
3) To submit your assignment, simply select the appropriate assignment link from the Assignments subsection of the course blackboard website. Upload your zip file and click **submit** (clicking send is not enough).

Please make sure ALL source files are included in your zip file when submitted. Errors in submission will be assessed –25 points. A program that does not compile as submitted will be given 0 points.

Only your FINAL submission will be graded.

For policies on late submissions, please see the Syllabus from the course home page. These policies will be enforced with **no exceptions**.

Plagiarism on Programming Assignments: Discussion of concepts, design issues, and programming ideas with colleagues and fellow students is proper and an integral part of learning. However, everyone should do their own coding. Code will be automatically compared and any suspected cases of plagiarism will be investigated and pursued thoroughly. If you happen to reuse snippets of code from the Internet or other sources, please document the sources and nature of what was used in detail to avoid potential problems. If you have any questions or doubts regarding the proper modes of collaboration or using external reference sources, please check with the Professor or TA before you submit your assignment.