Due dates D1: Sunday Feb 9 + D2: Sunday March 1, 2020
Late Submissions 20% per day per late deliverable
Teams You can do the project individually or in teams of at most 3.
Teams must submit only 1 copy of the project via the team leader's account.
Purpose In this project, you will implement and analyse a heuristic search.

Heuristic Search

1 Indonesian Dot Puzzle

The Indonesian Dot Puzzle is played on a $n \times n$ board. On each position of the board, a wooden token is placed. Each token has 2 sides: one with a white dot (○), and the other with a black dot (●). Initially, the tokens are placed on the board on a random side, and the goal is to find the smallest number of moves to bring the board to its goal configuration, where all tokens have their white side up (○).

For example, given the initial 4×4 board illustrated in Figure 1, the final goal is shown in Figure 2.

	1	2	3	4
A	●	●	●	○
B	●	○	○	●
C	●	●	○	○
D	○	●	●	●

Figure 1: Example of an initial board with $n = 4$.

	1	2	3	4
A	○	○	○	○
B	○	○	○	○
C	○	○	○	○
D	○	○	○	○

Figure 2: Goal state if $n = 4$.

The only legal move is to *touch* a token, which has the effect of flipping it and hence changing its color from ○ to ●, or vice versa, but also flips its (at most) 4 adjacent tokens (those immediately up, down, left and right) provided that they are within the bounds of the board.

For example, given the initial board shown in Figure 1, if token `C2` is touched, then $\widehat{\text{C2}}$ will become ○, but its 4 adjacent tokens `B2`, `D2`, `C1` and `C3` will also be flipped to ●, ○, ○ and ● respectively, leading to the board shown in Figure 3. If, after `C2`, the next token is chosen to be `A1`, then the resulting board will be as shown in Figure 4.

	1	2	3	4
A	●	●	●	○
B	●	●	○	●
C	○	○	●	○
D	○	○	●	●

Figure 3: Board after `C2` from Figure 1 is touched.

	1	2	3	4
A	○	○	●	○
B	○	●	○	●
C	○	○	●	○
D	○	○	●	●

Figure 4: Board after `A1` from Figure 3 is touched.

2 Your Task

Implement a solution for the Indonesian Dot for any size of boards (n) from [3 to 10].

2.1 Search Algorithms

Your project must implement the following 3 search algorithms:

1. Depth-first search (DFS)
2. Best-first search (BFS) we have to come up with the heuristics for #2 and #3
3. Algorithm $A^\star$ Just implement A* and set g(n) to 0 for BFS

Notes:

1. In order to ensure termination:
   (a) DFS will be performed up to a maximum depth of max_d in the state space (root = 1, and max_d is included). If no solution is found after reaching level max_d in the state space, the algorithm should backtrack and try another node.
   (b) BFS and $A^\star$ will be performed up to a maximum search path length of max_l. If no solution is found after searching max_l nodes, then you should consider that there is no solution.
2. For BFS and $A^\star$, you must develop your own heuristic(s) h that should be used for both BFS and $A^\star$ so you can compare the behavior of the search algorithms.
3. Ties between equivalent boards should be broken by preferring the board that has its first white token(s) at an earlier position based on a left-to-right, then top-down order. For example, assuming that the boards $b1$ and $b2$, shown in Figures 5 and 6, are equivalent for your search algorithm, then priority should be given to $b2$ because its first ∘ is at position `A1`, whereas the first ∘ in $b2$ is at `A4`, and `A1` is located before `A4`.

	1	2	3	4
A	●	●	●	○
B	●	○	○	●
C	●	●	○	○
D	○	●	●	●

Figure 5: Example of a board $b1$ - the first ∘ is at position `A4`.

	1	2	3	4
A	○	●	●	○
B	●	○	○	●
C	●	●	○	○
D	○	●	●	●

Figure 6: Example of a board $b2$ - the first ∘ is at position `A1`, so $b2$ has priority over $b1$ in Figure 5.

Similarly, assuming that the board $b3$ and $b4$, in Figures 7 and 8, are equivalent, then priority should be given to board $b4$ because although both boards have their first ∘ at the same position, the 2^{nd} ∘ of $b4$ is located before $b3$'s 2^{nd} ∘.

	1	2	3	4
A	●	●	○	●
B	●	●	●	●
C	●	●	○	○
D	○	●	●	●

Figure 7: Example of a board $b3$ - 1^{st} ∘ at `A3` and 2^{nd} ∘ at `C3`.

	1	2	3	4
A	●	●	○	●
B	●	○	○	●
C	●	●	○	○
D	○	●	●	●

Figure 8: Example of a board $b4$ - 1^{st} ∘ also at `A3` but 2^{nd} ∘ at `B2` - this board has priority over $b3$.

2.2 Programming Environment

To program the project, you must use Python 3.7 and run on the lab computers. In addition, you must use GitHub (make sure your project is private while developing).

2.3 The Input

Your code should be able to receive the path of an input file that contains test cases. The input file will contain a sequence of lines, containing:

1. the size of the puzzle (n)
2. the maximum depth search for DFS (max_d)
3. the maximum search path length (max_l) for BFS and $A^\star$
4. the values of the initial puzzle's $n \times n$ tokens, where each token will be represented by a `1` (•) or a `0` (○). The order of the tokens will be left-to-right, top-down.

When using DFS, you can ignore max_l and when using BFS or $A^\star$, you can ignore max_d.

For example, the input file could contain:

```
4 9 50 1110100111000111
3 100 2000 011001100
```

puzzle `#0`: puzzle of Figure 1 with $max_d = 9$ and $max_l = 50$
puzzle `#1`: a 3×3 puzzle with $max_d = 100$ and $max_l = 2000$

Note: There is no need to check the validity of the input file; you can assume that it will be correct.

2.4 The Output

For each input puzzle and each algorithm, your program should generate 2 output files: one for the solution path, and one for the search path. Since we have 3 algorithms, this means that for each line of the input file (see Section 2.3), your program must output 6 text files:

```
#_dfs_solution.txt   #_bfs_solution.txt   #_astar_solution.txt
#_dfs_search.txt     #_bfs_search.txt     #_astar_search.txt
```

where `#` is the puzzle number starting at zero.

For example, for the 1^{st} line of the input file, you should generate: `0_dfs_solution.txt` `0_dfs_search.txt`, `0_bfs_solution.txt`, `0_bfs_search.txt`, `0_astar_solution.txt`, and `0_astar_search.txt`.

2.4.1 Solution Files

Each solution file (`#_dfs_solution.txt`, `#_bfs_solution.txt` and `#_astar_solution.txt`) will contain the final solution found by the algorithm, or the string `no solution`.

For DFS, if a solution is found within max_d levels, your program will first display the initial configuration preceded by a 0 (zero), then for each move in the solution path, you should display the token to touch followed by the new configuration of the board, one puzzle per line. For example, if your final solution path for puzzle #1 above is: `A1 B2 D2 E5 C2` then `0_dfs_solution.txt` should contain[1]:

0	0 1 1 0 0 1 1 0 0 1 0 1 1 1 0	*0, then the initial board*
A1	1 1 1 0 0 1 1 0 0 1 0 1 1 1 0	*touch A1, followed by resulting board*
B2	1 1 1 0 0 1 1 0 0 1 0 1 1 1 0	*touch B2, followed by resulting board*
D2	1 1 1 0 0 1 1 0 0 1 0 1 1 1 0	*touch D2, followed by resulting board*
...		
E5	1 1 1 0 0 1 1 0 0 1 0 1 1 1 0	*touch E5, followed by resulting board*
C2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	*touch C2, followed by resulting board (the goal state)*

If your program cannot find a solution after searching max_d levels, then it should output:

```
no solution
```

For BFS and $A^\star$, if a solution is found after searching $\leq max\_l$ nodes, your program will display the solution using the same format as indicated above. Otherwise, it should display: `no solution`.

2.4.2 Search Files

Each search file (`#_dfs_search.txt`, `#_bfs_search.txt` and `#_astar_search.txt`) will contain the search path (the list of nodes that have been searched) by the algorithm. The search files should contain 1 line for each node searched. Each line should contain the $f(n)$, $g(n)$ and $h(n)$ values of the node, followed by the configuration of the node. For example:

15	10	5	111101100
10	5	5	100011000
8	8	0	000000000

Note: If the value of $f(n)$, $g(n)$ or $h(n)$ is irrelevant to a specific search algorithm, just display its value as zero (0).

A sample input file and sample output files are provided on Moodle.

[1] This is a random output.

3 Experiments

Note that this is a project, not an assignment. The difference is that it is open-ended and in addition to the required work stated above, you are expected to be creative, perform additional experimentations and analyse the results of your experimentations. Examples of experimentations include trying different heuristics and search algorithms, comparing the behavior of the different algorithms and with different puzzle sizes, ...

4 Deliverables

The submission of the project will consist of 2 deliverables:

1. Deliverable 1: code + demo for the max_d-limited depth-first search only
2. Deliverable 2: code + report + demo for all functionalities

4.1 Report

The report will be used to describe your experiments (see Section 3) motivate why you did what you did, state your results, and analyse them to draw insight. The intended audience of your report is me (your prof) and your TAs. Hence there is no need to explain the rules of the game again or to explain the theory behind the search algorithms. Your report should focus on **your** work.

Your report should be 4-5 pages (without references and appendices) and use the template provided on Moodle. The report should contain at least the following:

- ☐ $\frac{1}{2}$ *to 1 page:* Introduction and technical details.
- ☐ *1 to* $1\frac{1}{2}$ *page:* A description and justification of your heuristic(s). In particular, describe how it works, its strengths and its weaknesses (yes, indicating its weaknesses will give you more points as it shows that you are aware of them!).
- ☐ $\frac{1}{2}$ *to 1 pages:* A description of any difficulties that you have encountered and how you addressed them.
- ☐ *2 pages:* An analysis of the results of your experiments. In particular, you must compare and contrast the results of the search algorithms and the heuristic(s). Please note that your report must be analytical. This means that in addition to stating the facts (e.g. how many moves does a search algorithm yield), you should also analyse them (i.e. explain why something behaves this or that way, in which case something would be better than another ...).
- ☐ $\frac{1}{2}$ *page:* In the case of team work, a description of the responsibilities and contributions of each team member.
- ☐ Your report should have a reference section (not included in the page count) that properly cites all relevant resources that you have consulted (books, Web sites ...), even if it was just to inspire you. Failure to properly cite your references constitutes plagiarism and will be reported.
- ☐ Use appendices (not included in the page count), if you wish to show parts of the output or a code listing.

The report must:

- ☐ follow the Word or LaTeX template provided on Moodle.
- ☐ be submitted in PDF format
- ☐ be called `472_Project1_Report_ID1_ID2_ID3.pdf` where `ID1` is the ID of the team leader.

4.2 The Demos

You will have to demo your code for both deliverable 1 (demo 1) and deliverable 2 (demo 2). Demos will be done during the lab time on the lab machines. Regardless of the demo time, you will demo the program that was uploaded as the official submission on or before the due date. The schedule of the demos will be posted on Moodle. No special preparation is necessary for the demo (no slides or prepared speech). Your program will be expected to read an input file with a sequence of puzzles (see Section 2.3), and output its result for each puzzle (see Section 2.4). The results of your program will need to be uploaded on EAS during your demo. In addition, your TA will ask you questions on your code, and you will need to answer him/her satisfactorily.

5 Evaluation Scheme

Students in teams can be assigned different grades based on their individual contribution to project. Individual grades will be based on a peer-evaluation done after the deadline.

The team grade will be based on:

Deliverable 1 - code	functionality	7.5%
Deliverable 1 - demo	clear answers to questions, knowledge of the program, ...	2.5%
Deliverable 2 - code	functionality	20%
Deliverable 2 - demo	clear answers to questions, knowledge of the program, ...	5%
Deliverable 2 - code	design, programming style, ...	15%
Report & Experimentations	originality, motivation, clarity and conciseness, depth of the analysis, presentation, grammar, ...	50%
Total		100%

6 Submission

6.1 Submission Schedule

Each deliverable is due on the date indicated below.

Deliverable	Due Date	Upload as
Deliverable 1 - code	Feb 9, 2020, midnight	Project 1
Deliverable 1 - demo	Feb 10, 13 during lab time	Project 2
Deliverable 2 - final code + report	March 1, 2020, midnight	Project 3
Deliverable 2 - final demo	March 2, 5 during lab time	Project 4

If you work in a team, identify one member as the team leader. The only additional responsibility of the team leader is to upload all required files (including the files at the demo) from her/his account and book the demo on the Moodle scheduler. If you work individually, by definition, you are the team leader of your one-person team.

6.2 Submission Checklist

In your GitHub project, include a `README.md` file that contains:

1. on its first line: the URL of your GitHub repository,
2. specific and complete instructions on how to run your program on the desktops in the computer labs.

Deliverable 1 - Code Submission

- ☐ Create one zip file containing all your code and the `README.me` file.
- ☐ Name your zip file: `472_Project1_ID1_ID2_ID3.zip` where `ID1` is the ID of the team leader.
- ☐ Have the team leader upload the zip file at: https://fis.encs.concordia.ca/eas/ as `Project1`.

Deliverable 1 - Demo During your actual demo with the TA:

- ☐ Generate the output files for the input that the TA will give you.
- ☐ Create a zip file called: `472_Project2_ID1_ID2_ID3.txt` where `ID1` is the ID of the team leader.
- ☐ Have the team leader upload the zip file at: https://fis.encs.concordia.ca/eas/ as `Project2`.

Deliverable 2 - Code

- ☐ Read and sign the expectation of originality form (available on Moodle; or at: https://www.concordia.ca/content/dam/ginacody/docs/Expectations-of-Originality-Feb14-2012.pdf) and submit it in your package.
- ☐ Create one zip file, containing all your code, the `README.me` file, the report and the expectation of originality.
- ☐ Name your zip file: `472_Project3_ID1_ID2_ID3.zip` where `ID1` is the ID of the team leader.
- ☐ Have the team leader upload the zip file at: https://fis.encs.concordia.ca/eas/ as `Project3`.
- ☐ Print your report and submit a paper copy in the appropriate assignment box in EV 3.177.
- ☐ In addition, a few days after the deadline, make your GitHub repository public.

Deliverable 2 - Demo During your actual demo with the TA:

- ☐ Generate the output files for the input that the TA will give you.
- ☐ Create a zip file called: `472_Project4_ID1_ID2_ID3.txt` where `ID1` is the ID of the team leader.
- ☐ Have the team leader upload the zip file at: https://fis.encs.concordia.ca/eas/ as `Project4`.

Have fun!