Artificial Intelligence
Assignment 1

September 14, 2025

Viet Huy Duong
811433146

---

1. Explain the rationale behind heuristic search being fast yet incomplete. (5 pts)

Ans: The core idea of heuristic search is the guiding principle through the estimated distance from the current position to the goal. It leads to 2 factors. First is speed, depending on the heuristic algorithm, the search process might be faster if the estimated value is good enough, better than DFS and BFS in many cases. Second is incompleteness. As I said before, the algorithm depends on the estimated value, so the heuristic search tends to prune branches that seem less promising, so in many cases it may overlook paths that lead to optimal solutions based on heuristics, making the search incomplete, meaning fail to find all solution.

2. Explain triangular inequality of heuristic functions. (5 pts)

Ans: In heuristic search, inequality means $h(n) \leq c(n, n') + h(n')$ where $h(n)$ is the heuristic cost from node n to the goal, $c(n, n')$ is the actual cost of moving from node n to node n', and $h(n')$ is the heuristic cost from node n' to the goal. That ensures the heuristic does not overestimate the cost from current state to goal. For A* to find the optimal path, the heuristic must satisfy two properties: admissibility and consistency. The triangular inequality is the requirement for consistency. The consequence of consistency is f-value along a path never decreases in A*.

3. Explain the problems of global heuristic searches. (5 pts)

Ans: The problems of global heuristic search include the heuristic function, like the computational cost to explore the large portion of search space to calculate the heuristic value. And then, the memory requirement, for example, A* stores all nodes in memory (both open and closed lists) to track the paths. The third problem is designing the heuristic, the performance of search algorithms extremely depends on the heuristic. Some problems might be very difficult to construct the heuristic function, or the heuristic function too complex for running on a large search space.

4. Explain the problems of local search. (5 pts)

Ans: Explain the problems of local search. Contrasted with the global one, local search is more efficient in computational cost and memory, however, local search is harder to gain the optimal. Firstly, local search usually gets stuck in local optima, that means local search still produces the results that are good but not the best possible. Then, the second challenge is designing effective heuristics. Creating the heuristics to guide local search is more challenging because of the lack of global context.

5. Explain simulated annealing abstractly including the use of probability, temperature, temperature drop rate, and stochastic nature in the algorithm. (5 pts)

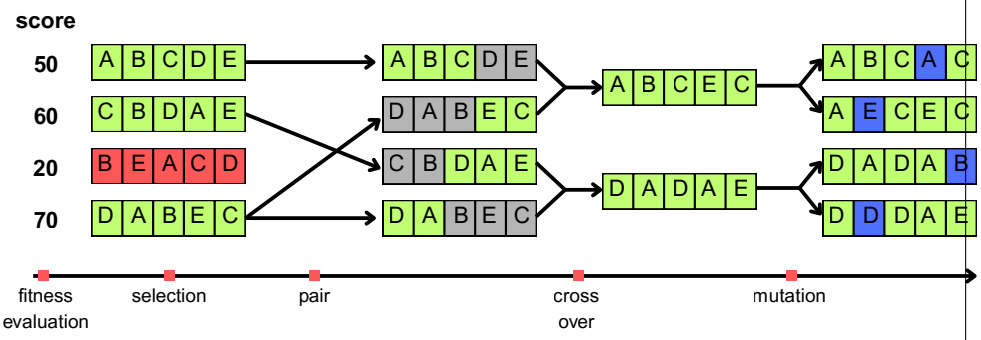
Ans: Simulated annealing is a local heuristic search algorithm, with the stochastic factor. Simulated annealing balances the exploration and exploitation by probability, including temperature and a cooling schedule:

- Probability: the algorithm even accepts the worse state as the next state with probability $e^{(-\delta/T)}$ with delta is the change in objective function ($\delta < 0$ in worse state) T is temperature.
- Temperature: at above, the temperature T controls the likelihood accepting the worse states. High T allows the searching exploration the search space, while low T focuses on exploitation.

- Temperature drop rate: also known as cooling schedule, the reduction of T over time.
- Stochastic Nature: As the name simulated annealing, high temperature means large fluctuations of particles due to kinetic energy, which can be visualized as a larger fluctuation in the search state avoiding being trapped in the local optima. And then the matter cools down, and the chaos gradually disappears, at this point the state becomes more stable as well as the search process ends

6. Explain genetic algorithms using a simple figure. (5 pts)

Ans: Genetic algorithm is a heuristic search algorithm inspired by the natural evolution of genes. Including initialization, fitness evaluation, selection, pairing and crossover, mutation. An iteration of search process described below:

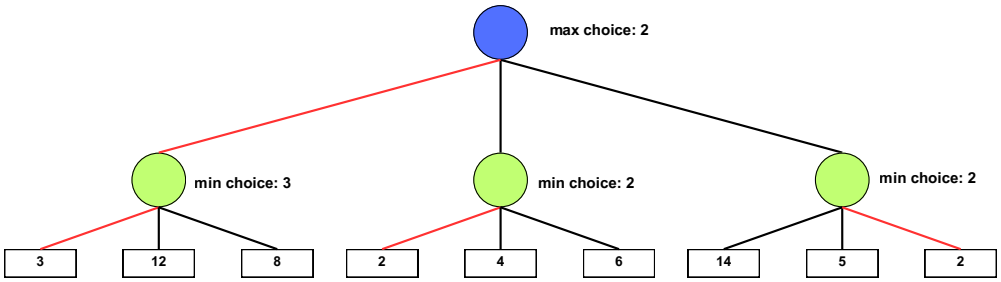


7. Explain min-max algorithm in adversarial game search, its properties and limitations. (5 pts)

Ans: Min-Max algorithm is a heuristic search algorithm used in adversarial gameplay like chess, Caro,... In minmax search, we assume that there are two players, Max player tries to increase the score, while Min tries to decrease it. And the Game assumes each player will choose the optimal step for them at the current stage.

Step by Step:

- Modeling the game to be a tree.
- Start from the leaf, assign values to leaf nodes (e.g. 100 for Max win, -100 for Max lose).
- Backwards to the root, for each stage, Min and Max choose the step which is better for them.
- Choose the best move at the root.



The properties:

- Optimal: Guaranteed best move if fully searched
- Complete: return the solution for finite tree

- Adversarial: approriate for adversarial game

Drawbacks:

- The complexity: Min-Max is efficient for adversarial games but is limited by complexity and depends on the evaluation function.
- Memory requirement: to store many states.
- Depends on the evaluation function.
- The assumption about the enemy: Min-Max assumes that the enemy without any complex strategy, sometimes will not efficient in reality.

8. Explain the ant colony optimization. (5 pts)

Ans: ACO is inspired by the natural behavior of ants, often used in combinatorial optimization problems, such as traveling salesman, scheduling, routing vehicle problems. In nature, ants use a chemical called pheromone to mark their paths. When an ant finds food, other ants tend to follow that path, the more ants have the higher the concentration of pheromone, leading to the optimal path being reinforced by a large amount of pheromone. From there, ACO simulates this phenomenon to find optimal solutions to problems. ACO works in the following steps: Initialization: Create a set of agents, representing the agent searching for the first round. Initialize the initial pheromone level on the paths, usually a small, uniform value to avoid bias Build the solution: An agent will start moving on the graph, choosing a path based on 2 factors pheromone, and heuristic. The path from i to j will be chosen with probability:

$$P_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum(\tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta})}$$

Where:

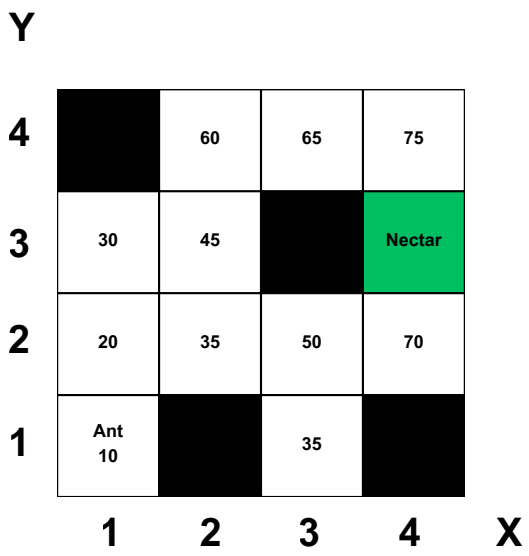
- P_{ij}: Probability of choosing the edge from i to j.
- τ_{ij}: Pheromone level on the edge $i - j$.

- η_{ij}: Heuristic value (typically the inverse of the distance, $1/d_{ij}$).
- α, β: Parameters controlling the influence of pheromone and heuristic.

Then comes the pheromone update step: After all agents complete their movement, the pheromone will be updated: - Evaporation: reduce the pheromones on ALL edges to avoid premature convergence without reaching the optimal (according to the formula $\tau_{ij} \leftarrow (1-\rho)\tau_{ij}$, where ρ is the evaporation rate). - And then increase the pheromones on the edges of the optimal path. based on the assessment of the movement of the agents. - The process repeats until the desired convergence is achieved or until a set number of iterations is reached

9. Local Search: An ant is searching for food. The intensity of the nectar's scent is given in the matrix cells. Ant is at location (1, 1) and the nectar is at location (4, 3), where 4 is the row index and 3 is the column index as shown in the figure. The ant can move horizontally, vertically and diagonally. There are obstructions in three cells where ant cannot go. The cells are (2, 1), (3, 3) and (4, 1). Draw the tree for possible movements, and the best path taken by ant assuming that ant can sense the smell up to only two cells beyond horizontally and vertically and one cell diagonally. For example, at (1, 1), she can sense the smell in (1, 2), (2, 1), (1, 3), (3, 1) and (2, 2). (10 pts)

Ans: Because there are some problem with the original table in the question, i will redefine the map here:



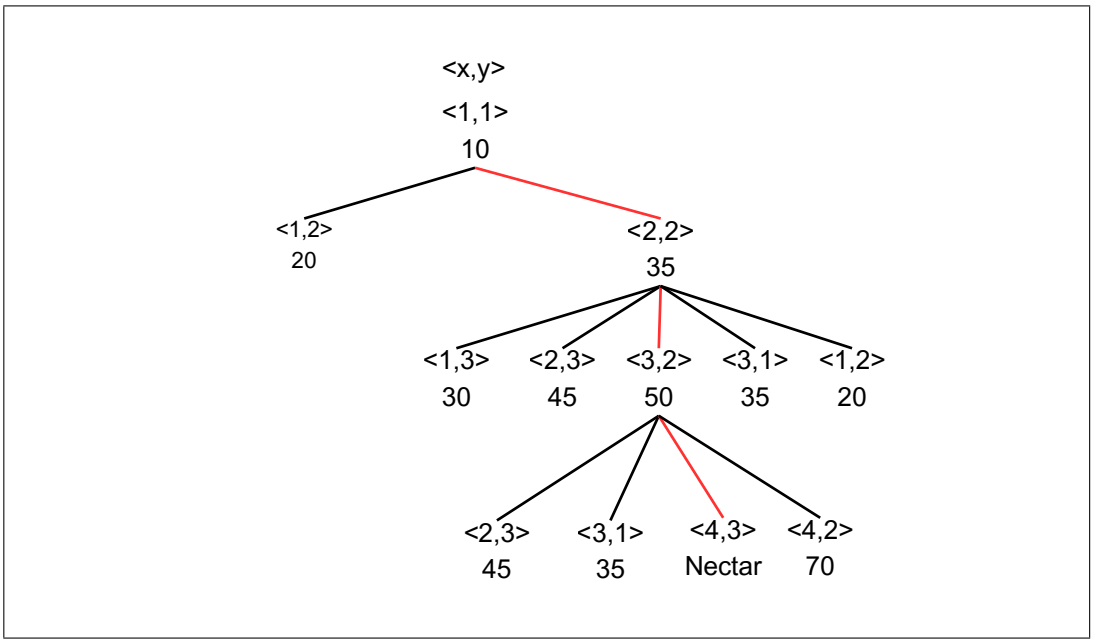
Local search is performed step by step, the location of each cell will be presented by $< x, y >$ for:

- (1,1) this is the start node, expand it to (1,2) and (2,2), (2,2) is the node with higher estimated reward (35) , let's expand (2,2).

- (2,2) have these child nodes: (1,3), (2,3), (3,2), (3,1), (1,2), (3,2) is the node with highest reward. Go to (3,2).

- The (3,2)'s child node included the goal, so go to nectar and complete the searching.

The entire search progress is visualized bellow:



10. Q3? (Type anything here)

Ans: A3.

11. Write a memory bound A* algorithm that retains M% (use ceiling function to avoid fractions) of the total fringe at any time where M is a parameter using universal and existential quantifiers and set-based abstractions. Discard the p% worst cost fringe candidates after every move. The value of p improves linearly as the search progresses at the rate of r%. (10 pts)

Ans: The algorithm for memory-bounded A* bellow. There is a small update from the original A* in the fringe pruning after each iteration.

Algorithm 1 Memory-Bounded A* Algorithm

Require: Start node s, goal predicate *Goal*, successor function *Neighbor*, cost function c, heuristic h, parameters M, p, r

$F \leftarrow \{s\}$
$S \leftarrow \emptyset$
$g(s) \leftarrow 0$
$back(s) \leftarrow \perp$
$f(n) \leftarrow g(n) + h(n) \quad \forall n$
$k \leftarrow 0$
while $F \neq \emptyset$ **do**
    if $p \geq 100$ or $M == 0$ **then**
        return failure
    end if
    if $\exists n \in F : Goal(n)$ **then** // return path if the goal is visited
        Reconstruct and return path from s to n using *back*
    end if
    $n \leftarrow \arg\min_{m \in F} f(m)$ // find the node with min value in fringe
    $S \leftarrow S \cup \{n\}$ // add n to the closed list
    $F \leftarrow F \setminus \{n\}$ // delete n in fringe
    $F \leftarrow F \cup Neighbor(n)$
    $f(m) \leftarrow c(n, m) + h(m), \quad \forall m \in Neighbor(n)$
    Sort F in ascending order of $f(n)$
    $k_M \leftarrow \lceil (M/100) \cdot |F| \rceil$
    $F \leftarrow \{F[i] \mid 1 \leq i \leq k_M\}$
    $N' \leftarrow |F|$
    $k_p \leftarrow \lceil (p/100) \cdot N' \rceil$
    $F \leftarrow \{F[i] \mid 1 \leq i \leq N' - k_p\}$
    $p \leftarrow p + r$
end while
return failure

12. Given a population of 4 initial positions in a 4 X 4 board of N-queens given as the column vector $< 1, 3, 2, 4 >$, $< 1, 1, 1, 1 >$, $< 1, 2, 3, 4 >$, $< 1, 1, 1, 2 >$, Use local search and movement of one queen picked randomly to place the queen such that the number of attacking queen reduces for each of initial positions separately. Show at least four moves. Use random number generator from a computer to

pick up the column number of the queen to be moved. Show the process and the table of random number generated. (10 pts)

Ans: I implemented the program follow this Algorithm (CountConflicts is the function return the number of queen in danger).

Algorithm 2 Find Best Move

```
1:  procedure BestMove(board)
2:      row ← random[1 → n]
3:      col ← board[row]
4:      min ← CountConflicts(board)
5:      best ← [col]
6:      for c = 1 to n do
7:          if c ≠ col then
8:              temp ← board
9:              temp[row] ← c
10:             conf ← CountConflicts(temp)
11:             if conf < min then
12:                 min ← conf
13:                 best ← [c]
14:             end if
15:         end if
16:     end for
17:     return best
18: end procedure
```

Run BestMove 4 times for each initial board. we have the table for each board bellow: **Configuration: [1, 3, 2, 4], Initial Conflicts: 2**

Move	Row Selected	Old Position	New Position	New Board	Conflicts
1	1	(1, 1)	(1, 1)	[1, 3, 2, 4]	2
2	3	(3, 2)	(3, 2)	[1, 3, 2, 4]	2
3	2	(2, 3)	(2, 3)	[1, 3, 2, 4]	2
4	1	(1, 1)	(1, 1)	[1, 3, 2, 4]	2

Random Numbers Generated (Row Selected): [1, 3, 2, 1]

Configuration: [1, 1, 1, 1], Initial Conflicts: 6 (conflict is 6 > 4 is completely normal, my conflict function determines which pairs of queens can attack each other)

Move	Row Selected	Old Position	New Position	New Board	Conflicts
1	4	(4, 1)	(4, 2)	[1, 1, 1, 2]	4
2	1	(1, 1)	(1, 4)	[4, 1, 1, 2]	2
3	2	(2, 1)	(2, 1)	[4, 1, 1, 2]	2
4	1	(1, 4)	(1, 4)	[4, 1, 1, 2]	2

Random Numbers Generated (Row Selected): [4, 1, 2, 1]

Configuration: [1, 2, 3, 4], Initial Conflicts: 6

Move	Row Selected	Old Position	New Position	New Board	Conflicts
1	4	(4, 4)	(4, 3)	[1, 2, 3, 3]	4
2	3	(3, 3)	(3, 4)	[1, 2, 4, 3]	2
3	3	(3, 4)	(3, 4)	[1, 2, 4, 3]	2
4	2	(2, 2)	(2, 2)	[1, 2, 4, 3]	2

Random Numbers Generated (Row Selected): [4, 3, 3, 2]

Configuration: [1, 1, 1, 2] (the original question is $< 1, 1, 1, 2, 4 >$ so i assume the initial board is $< 1, 1, 1, 2 >$) , Initial Conflicts: 4

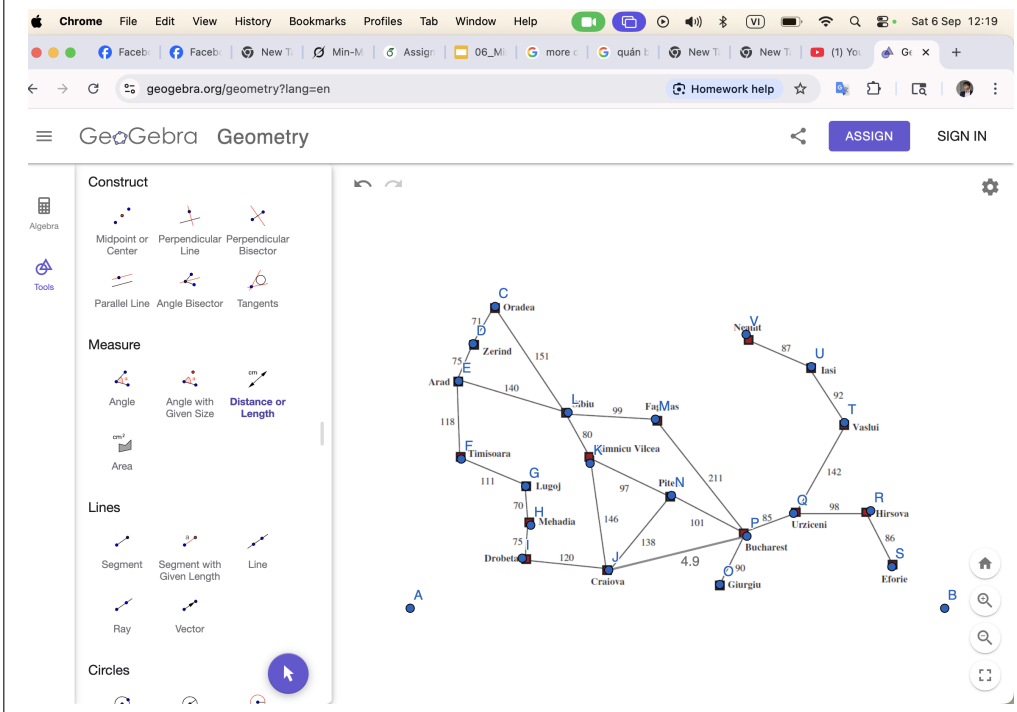
Move	Row Selected	Old Position	New Position	New Board	Conflicts
1	3	(3, 1)	(3, 4)	[1, 1, 4, 2]	1
2	3	(3, 4)	(3, 4)	[1, 1, 4, 2]	1
3	2	(2, 1)	(2, 3)	[1, 3, 4, 2]	1
4	2	(2, 3)	(2, 1)	[1, 1, 4, 2]	1

Random Numbers Generated (Row Selected): [3, 3, 2, 2]

13. For the example in the book (see Figure on the next page), work out if a person has to travel from Neamt to Craiova. Show the steps for greedy best first search and A* search. Measure the distance from Neamt using a ruler, and convert to mile knowing that one centimeter is approximately equivalent to 40 miles in the map for heuristic estimation. (20 pts)

Ans: Because the problem was presented on computer, so i use the geometry website to measure the heuristic distance.

for example:

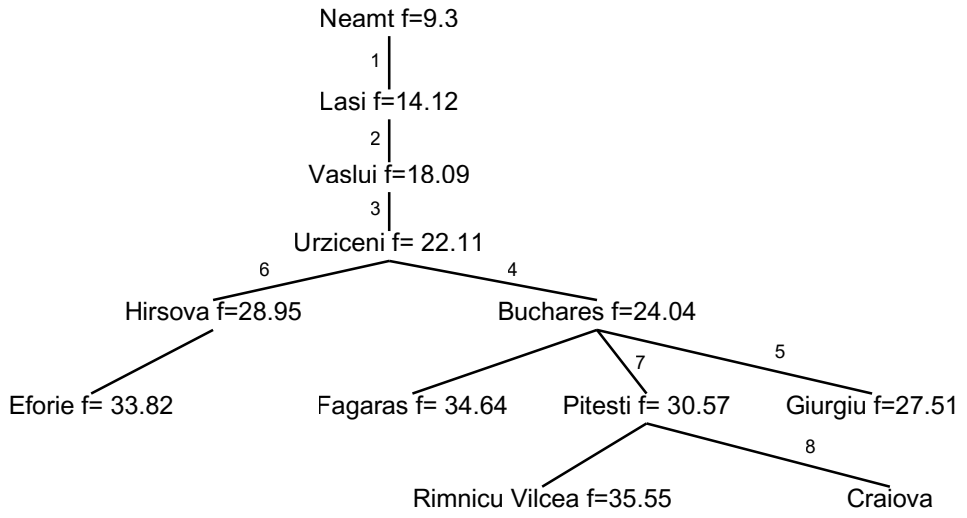


And then,The two tables below summarize the information needed for A*. Because the longest heuristic is 10 so i decide to normalize the real cost between each city to (1-10) while still remain the regression relation. The heuristic and normalized are also checked to be admission and consistant.

City	heuristic
Arad	8.3
Bucharest	4.8
Craiova	0
Drobeta	real
Eforie	9.9
Fagaras	5.4
Giurgiu	4
Hirsova	9.1
Iasi	10
Lugoj	4.1
Mehadia	3.2
Neamt	9.3
Oradea	9.8
Pitesti	real
Rimnicu Vilcea	real
Sibiu	5.7
Timisoara	6.4
Urziceni	6.9
Vaslui	9.6
Zerind	9.1

Edge	Original Weight	Normalized Weight
Oradea - Zerind	71	3.36
Zerind - Arad	75	3.55
Oradea - Sibiu	151	7.16
Arad - Sibiu	140	6.64
Arad - Timisoara	118	5.59
Timisoara - Lugoj	111	5.26
Lugoj - Mehadia	70	3.32
Mehadia - Drobeta	75	3.55
Drobeta - Craiova	120	5.69
Sibiu - Fagaras	99	4.69
Sibiu - Rimnicu Vilcea	80	3.79
Rimnicu Vilcea - Pitesti	97	4.60
Rimnicu Vilcea - Craiova	146	6.92
Fagaras - Bucharest	211	10.00
Pitesti - Bucharest	101	4.79
Pitesti - Craiova	138	6.54
Bucharest - Giurgiu	90	4.27
Bucharest - Urziceni	85	4.03
Urziceni - Hirsova	98	4.64
Urziceni - Vaslui	142	6.73
Vaslui - Iasi	92	4.36
Iasi - Neamt	87	4.12

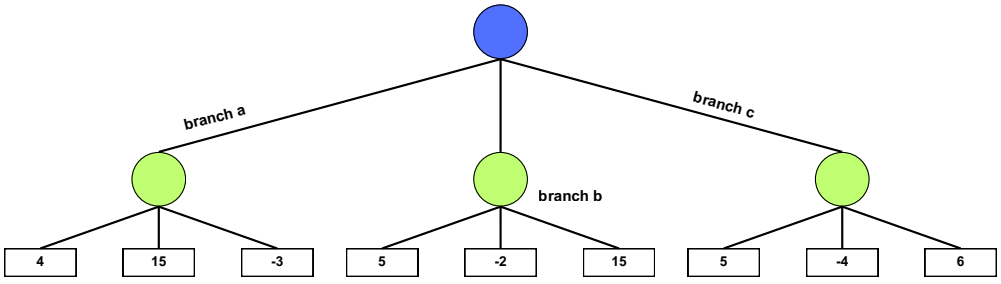
The search process will be presented step by step as shown below:



And the real total cost from Neamt to Craiova along the shortest path is **645**.

14. Min-max tree: For the following Min-max tree identify the min-max value at each node, and using alpha-beta pruning identify which branches need not be explored. Explain clearly the rationale, specify the alpha-beta condition which applies for each node not to be expanded. (10 pts)

Ans: In this problem, i will explain step by step how to apply alpha beta pruning with this tree: firstly, i will call the branch with number like bellow:



- Step 1: fully visits branch a, return the value for branch a is -3 (min value)
- Step 2: Marks that the root (max node) will only choose the node with value $max \geq -3$.
- Step 3: visits each node in branch b, if total node < -3 stop visiting and pruns all other node in this branch. the minimum value of branch b is -2. Return -2 and update the value at max node ($max \geq -2$).
- Step 4: similarity, visited each node in branch c, we see that $-4 < -2$ marks branch c value ≤ -4, so we prune the last node 6.

So we done the minimax search with alpha beta pruning and the **last result is -2**

15. For a 6 X 6 board, design a good connect four evaluation-function for in-max trees assuming that a players (including computer) can see only upto two moves ahead and show the Min-Max tree for the next two moves by human and two moves by the computer for the following board position. Assume that red belongs to human; black belongs to computer. The next move is for the human.

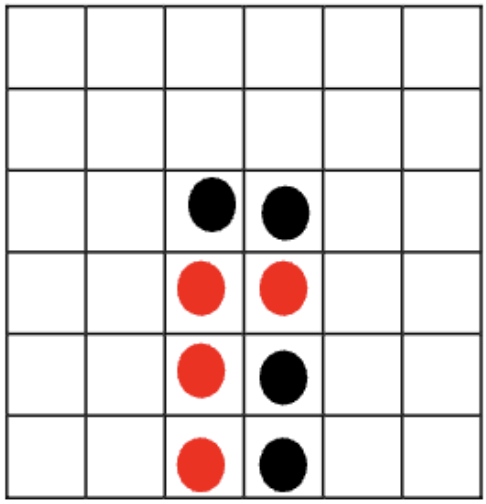


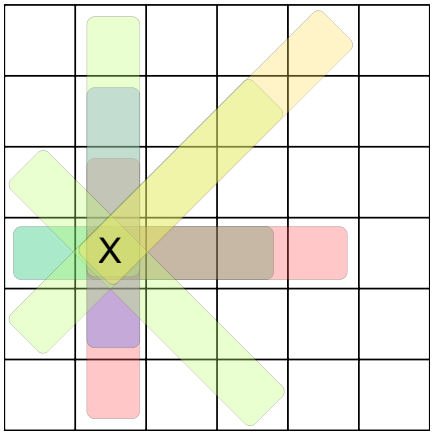
Figure 3 for Bonus Problem

(10 pts)

Ans: I will describe how I modeled the problem and how I built the heuristic for the problem.

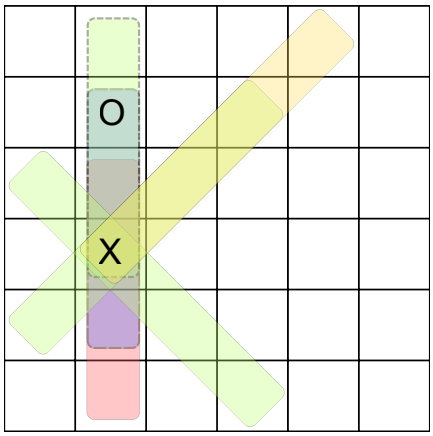
A chessboard will be represented by a 6x6 tensor, with the value 1 representing the player's piece, -1 representing the machine's piece, and 0 representing an empty square.

To evaluate the heuristic of a next move choice, I will rely on the following factors. at that chessboard position, how many consecutive lines of 4 cross it, for example:



like the illustration above, at square X, we have 8 possible winning moves.

Next, we will consider the current state of the chessboard. for each winning move considered above, if there is an enemy piece on that path, that means that path is blocked, we will remove it from the set of possible paths.

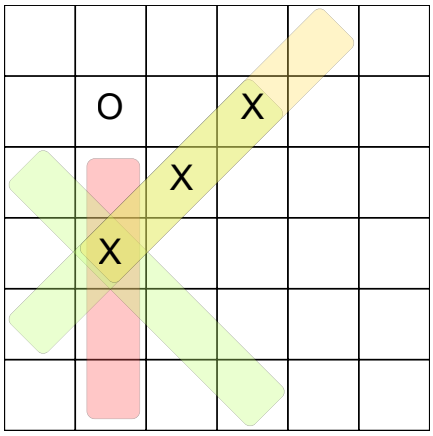


As above, the areas with dashed line border are removed because it were blocked by enemy.

However, if the heuristic is that simple, the algorithm will tend to look for moves in open positions, far away from other pieces. So we need to consider one more factor, which is our pieces, so for possible moves that have not been blocked, if there are allies on that move, we will add points to that move, the

coefficient will decrease based on the distance of the allies, for example if the ally is 3 squares away +1, 2 squares away +2, 1 square away +3 points.

For example:



Which the board like the example, after ignore the line which be blocked, i will weighted the remain lines. The weight function will be:

$$f(\text{line}) = (1 + \sum_{k \neq p}(4 - |p - k|))^{num_{Allies}}$$

So the orange line will have score:

$score = (1 + (4 - 1) + (4 - 2))^2 = 36$

Up to now, my heuristic function has been gradually completed, however, the factors so far have only considered the attack aspect, what about defense?

I decided to use the opponent's attack points as my defense points, this is also reasonable for the logic of the game, because the opponent's potential attack moves should be the moves that need to be blocked first. Then I will do, add up all the values of the score matrix, I will have 1 numerical value, which is the point representing our attack advantage, minus the opponent's attack advantage (or can also be considered our defense point), we have the value and ready for mini-max.

However, we didn't care about the 4-connections game, i will add the win checker condition, that mean if any choice lead to a side win, the value of that cell will become inf or -inf with enemy.

And this is the output of above algorithm with depth of min-max tree is 4 (indicate for 2 next step observation)

```
Starting simulation with depth 4. Max steps: 10
Current Board:
Row 0: . . . . . .
Row 1: . . . . . .
Row 2: . . O O . .
Row 3: . . X X . .
Row 4: . . X O . .
Row 5: . . X O . .
----------------
--- Step 1: Player 1 (X)'s turn ---
Top 5 moves ranked by score:
1. Move (2, 4): Score = 1000000
2. Move (3, 1): Score = 1000000
3. Move (3, 4): Score = 1000000
4. Move (3, 5): Score = -85
5. Move (3, 0): Score = -117
Best move: (2, 4) with score 1000000
Player 1 (X) plays at (2, 4) (row 2, col 4) with score 1000000
Current Board:
Row 0: . . . . . .
Row 1: . . . . . .
Row 2: . . O O X .
Row 3: . . X X . .
Row 4: . . X O . .
Row 5: . . X O . .
----------------
Top 5 moves ranked by score:
1. Move (0, 0): Score = inf
2. Move (0, 1): Score = inf
3. Move (0, 2): Score = inf
4. Move (0, 3): Score = inf
5. Move (0, 4): Score = inf
Best move: (0, 0) with score inf
Player -1 (O) plays at (0, 0) (row 0, col 0) with score inf
```

```
Current Board:
Row 0: O . . . . .
Row 1: . . . . . .
Row 2: . . O O X .
Row 3: . . X X . .
Row 4: . . X O . .
Row 5: . . X O . .
----------------
--- Step 3: Player 1 (X)'s turn ---
Top 5 moves ranked by score:
1. Move (1, 5): Score = inf
2. Move (5, 1): Score = inf
3. Move (0, 1): Score = 1000000
4. Move (0, 2): Score = 1000000
5. Move (0, 3): Score = 1000000
Best move: (1, 5) with score inf
Player 1 (X) plays at (1, 5) (row 1, col 5) with score inf
Current Board:
Row 0: O . . . . .
Row 1: . . . . . X
Row 2: . . O O X .
Row 3: . . X X . .
Row 4: . . X O . .
Row 5: . . X O . .
----------------
--- Step 4: Player -1 (O)'s turn ---
Player 1 wins!
Simulation ended after 4 steps.
```

At the first step of O (enemy), the choice return $< 0, 0 >$ because all step lead to X win.