



UNIVERSITY OF
BIRMINGHAM

ARTIFICIAL INTELLIGENCE 1 UNINFORMED SEARCH

DR LEONARDO STELLA

SCHOOL OF COMPUTER SCIENCE

2023/2024 - Week 8

AIMS OF THE SESSION

This session aims to help you:

- Understand the concept of asymptotic analysis
- Formulate a search problem
- Explain the steps involved in Breadth-First Search (BFS), Depth-First Search (DFS) and its variations
- Apply the algorithms to solve search problems

OUTLINE

- 1 Recap: Asymptotic Analysis
- 2 Formulating a Search Problem
- 3 Breadth-First Search
- 4 Depth-First Search

OUTLINE

- 1 Recap: Asymptotic Analysis
- 2 Formulating a Search Problem
- 3 Breadth-First Search
- 4 Depth-First Search

RECAP: QUIZ

Q1. To determine the quality of an algorithm and/or compare the performance of multiple algorithms, a popular approach is **benchmarking**. Which of the following statements are correct about this approach?

- ☐ It is a mathematical abstraction over the exact number of operations and content of the input (size)
- ☐ It allows logic proofs of a given assertion
- ☐ It measures the time it takes to run an algorithm (speed)
- ☐ It measures the memory consumption on a given computer with specific input data
- ☐ It is independent of any particular implementation

RECAP: QUIZ

Q1. To determine the quality of an algorithm and/or compare the performance of multiple algorithms, a popular approach is **benchmarking**. Which of the following statements are correct about this approach?

- ☐ It is a mathematical abstraction over the exact number of operations and content of the input (size)
- ☐ It allows logic proofs of a given assertion
- ☒ It measures the time it takes to run an algorithm (speed)
- ☒ It measures the memory consumption on a given computer with specific input data
- ☐ It is independent of any particular implementation

RECAP: QUIZ

Q2. To determine the quality of an algorithm and/or compare the performance of multiple algorithms, another popular approach is **asymptotic analysis**. Which of the following statements are correct about this approach?

- ☐ It is a mathematical abstraction over the exact number of operations and content of the input (size)
- ☐ It allows logic proofs of a given assertion
- ☐ It measures the time it takes to run an algorithm (speed)
- ☐ It measures the memory consumption on a given computer with specific input data
- ☐ It is independent of any particular implementation

RECAP: QUIZ

Q2. To determine the quality of an algorithm and/or compare the performance of multiple algorithms, another popular approach is **asymptotic analysis**. Which of the following statements are correct about this approach?

- ☒ It is a mathematical abstraction over the exact number of operations and content of the input (size)
- ☐ It allows logic proofs of a given assertion
- ☐ It measures the time it takes to run an algorithm (speed)
- ☐ It measures the memory consumption on a given computer with specific input data
- ☒ It is independent of any particular implementation

RECAP: QUIZ

- Q3. The first step in the analysis is to abstract over the input. In practice, we characterise the size of the input, which we denote by n .
- ☐ True
 - ☐ False

RECAP: QUIZ

- Q3. The first step in the analysis is to abstract over the input. In practice, we characterise the size of the input, which we denote by n .
- ☒ True
 - ☐ False

RECAP: QUIZ

- Q4. The second step is to abstract over the implementation. The idea is to find some measure that reflects the running time of the algorithm.
- ☐ True
 - ☐ False

RECAP: QUIZ

- Q4. The second step is to abstract over the implementation. The idea is to find some measure that reflects the running time of the algorithm.
- ☒ True
 - ☐ False

RECAP: QUIZ

Q5. Which of the following conditions must hold for $f(n) \in \mathcal{O}(g(n))$?

- ☐ $\exists k > 0 \exists n_0 \forall n > n_0 : |f(n)| \leq kg(n)$
 'there exists a positive constant k and a (positive) value n_0 such that, for all $n > n_0$, function f is bounded asymptotically from above by a function g (up to a constant factor)'
- ☐ $\exists k > 0 \exists n_0 \forall n > n_0 : |f(n)| \geq kg(n)$
 'equivalent to saying that function f is bounded asymptotically from below by a function g (up to a constant factor)'
- ☐ $\exists k_1, k_2 > 0 \exists n_0 \forall n > n_0 : k_1g(n) \leq |f(n)| \leq k_2g(n)$
 'equivalent to saying that function f is bounded asymptotically from above and from below by a function g (up to a constant factor)'
- ☐ None of the above

RECAP: QUIZ

Q5. Which of the following conditions must hold for $f(n) \in \mathcal{O}(g(n))$?

- $\exists k > 0 \exists n_0 \forall n > n_0 : |f(n)| \leq kg(n)$
 'there exists a positive constant k and a (positive) value n_0 such that, for all $n > n_0$, function f is bounded asymptotically from above by a function g (up to a constant factor)'
- $\exists k > 0 \exists n_0 \forall n > n_0 : |f(n)| \geq kg(n)$
 'equivalent to saying that function f is bounded asymptotically from below by a function g (up to a constant factor)'
- $\exists k_1, k_2 > 0 \exists n_0 \forall n > n_0 : k_1 g(n) \leq |f(n)| \leq k_2 g(n)$
 'equivalent to saying that function f is bounded asymptotically from above and from below by a function g (up to a constant factor)'
- None of the above

RECAP: QUIZ

Q6. Given the following algorithm, determine its complexity.

```
function SUMMATION(sequence) returns a number
sum ← 0
for i = 1 to LENGTH(sequence) do
    sum ← sum + sequence[i]
return sum
```

- ☐ $\mathcal{O}(\log n)$
- ☐ $\mathcal{O}(n \log n)$
- ☐ $\mathcal{O}(n)$
- ☐ $\mathcal{O}(n^2)$

RECAP: QUIZ

Q6. Given the following algorithm, determine its complexity.

```
function SUMMATION(sequence) returns a number
sum ← 0
for i = 1 to LENGTH(sequence) do
    sum ← sum + sequence[i]
return sum
```

- ☐ $\mathcal{O}(\log n)$
- ☐ $\mathcal{O}(n \log n)$
- ☒ $\mathcal{O}(n)$
- ☐ $\mathcal{O}(n^2)$

RECAP: QUIZ

Q6. Given the following algorithm, determine its complexity.

```
function SUMMATION(sequence) returns a number
sum ← 0
for i = 1 to LENGTH(sequence) do
    sum ← sum + sequence[i]
return sum
```

- Step 1: we abstract over the input, i.e., the length of the *sequence*
- Step 2: abstract over the implementation, i.e., total number of steps.
If we call this characterisation $T(n)$ and we count the lines of code, we have: $T(n) = 2n + 2$
- This means that the above runs in $\mathcal{O}(n)$ asymptotically

OUTLINE

- 1 Recap: Asymptotic Analysis
- 2 Formulating a Search Problem**
- 3 Breadth-First Search
- 4 Depth-First Search

PROBLEM-SOLVING AGENTS

- In this lecture, we introduce the concept of a goal-based agent called **problem-solving agent**

Definition: An **agent** is 'something', an 'entity', that perceives the world (or the environment) and acts in this environment.

Definition: A problem-solving agent is an agent that

- uses atomic representations (each state of the world is perceived as indivisible);
- requires a precise definition of the problem and its goal/solution.

FORMULATING A SEARCH PROBLEM

- Many problems in AI, especially in certain contexts, e.g., sequential tasks and games, can be viewed as a search for a solution
- We make the following assumptions about the environment:
 - **Observable**, the agent is able to know the current state
 - **Discrete**, there are only finitely many actions at any state
 - **Known**, possible to determine which states are reached by which action
 - **Deterministic**, each action has exactly one outcome

Activity. A robot needs to navigate from one point to another in a room. In pairs or small groups, consider the following questions:

- How can we make the environment observable? [Localisation, SLAM](#)
- How can we ensure that the set of actions is finite? [Discretise](#)
- How can the agent determine which states are reached by which action? [Mapping states to nodes and actions to edges of a graph](#)
- When does each action produce one outcome? [Do not consider stochastic/probabilistic approaches](#)

FORMULATING A SEARCH PROBLEM

Definition: Formulating a search problem is the process to formally define a search for a solution. A search problem is defined by five components:

- **Initial state**, the state where the agent starts its search;
- **Action set**, the set \mathcal{A} describing the actions that can be executed in any state $s_i \in \mathcal{S}$;
- **Transition model**, a mapping between states and actions, i.e., the states resulting from executing each action $a_i \in \mathcal{A}$ in every state $s_i \in \mathcal{S}$;
- **Goal test**, to determine if a state is a goal state;
- **Path cost function**, which assigns a value (cost) to each path.

- The first three components define the **state space** of the problem
- The state space can take the form of a directed graph or network

SOLUTION, COST AND PATH

- The agent's task is to find out how to act, now and in the future
- In other words, the task consists in finding the sequence of actions from the initial state to a goal state

Definition: The **solution** of a search problem is the sequence of actions from the initial state to a goal state. The **cost** of a solution is the sum of the cost of the actions from the initial state to the goal state.

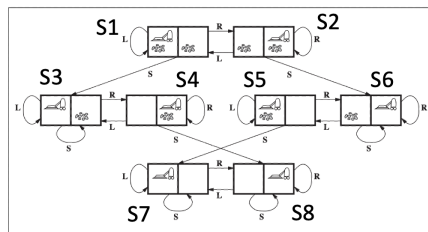
Definition: A **path** in the state space is a sequence of states connected by a sequence of actions.

EXAMPLE: VACUUM WORLD

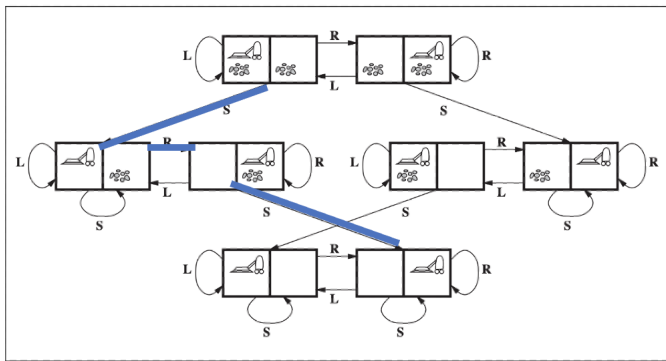
Activity. Consider the vacuum world example from the book. In pairs or small groups, formulate the corresponding search problem. Hint: the state is determined by the dirt location and agent location.

- Initial state: any of the states
- Actions: L (left), R (right), S (suction)
- Transition model: see image
- Goal test: checks if both squares are clean
- Path cost: each action has unitary cost

- Solution: S, R, S
- Cost: $3 = 1 + 1 + 1$



EXAMPLE: VACUUM WORLD

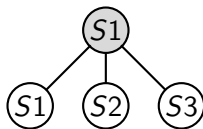
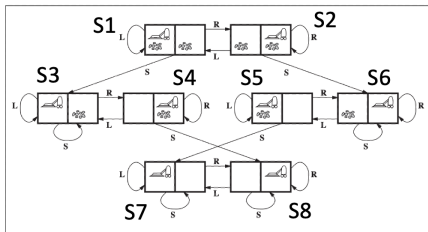


SEARCHING FOR SOLUTIONS

- The solution to a search problem is a sequence of actions from initial state to a goal state
- Possible solutions, or sequences of actions, form a **search tree**:
 - The initial state is placed at the root
 - Actions correspond to the branches
 - Nodes correspond to the state space
- The process consists of expanding the current node (state) by applying each possible actions; it generates a new set of states

SEARCHING FOR SOLUTIONS

- Let us consider the vacuum world example from before
- If $S1$ is the initial state and $\{S7, S8\}$ is the set of goal states, the search tree after expanding the root node is:

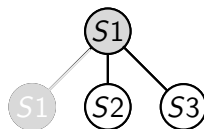
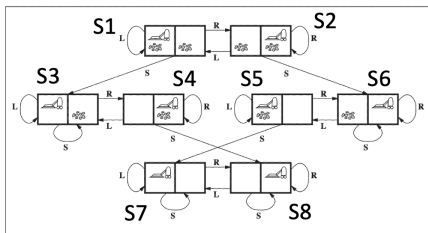


SEARCHING FOR SOLUTIONS

- Each node resulting from the first expansion is a **leaf node**

Definition: The set of all leaf nodes available for expansion at any given time is called the **frontier** (another common term is the **open list**).

- The path from $S1$ to $S1$ is a **loopy path** (or **repeated state**) and in general is not considered unless specified:



DISCUSSION

- It is important to note that typical AI problems have a large number of states and it is virtually impossible to draw the state space graph
- On the contrary, the state space graph for the vacuum world example has a small number of states
- For example, the state space graph for the game of chess is very large
- The process of formulating a search problem is a creative process. However, one formulation can be better than another

OUTLINE

- 1 Recap: Asymptotic Analysis
- 2 Formulating a Search Problem
- 3 Breadth-First Search**
- 4 Depth-First Search

UNINFORMED SEARCH STRATEGIES

Definition: **Uninformed Search** (also called **blind search**) is a term used to define the set of strategies having no additional information about the state space beyond that provided in the problem formulation.

- Uninformed search strategies can only generate successors and distinguish a goal state from a non-goal state
- The key difference between two uninformed search strategies is the **order** in which nodes are expanded

BREADTH-FIRST SEARCH

- Breadth-First search is one of the most common search strategies:
 - The root node is expanded first
 - Then, all the successors of the root node are expanded
 - Then, the successors of each of these nodes
 - Uninformed search strategies can only generate successors and distinguish a goal state from a non-goal state
- In different words, the order in which frontier nodes are expanded is level by level (a given depth of the tree)
- This is equivalent to expanding the **shallowest** unexpanded node in the frontier; equivalent to a **queue (FIFO)**

BREADTH-FIRST SEARCH

```

function BREADTH-FIRST-SEARCH(problem) returns a solution node or failure
  node  $\leftarrow$  NODE(problem.INITIAL)
  if problem.IS-GOAL(node.STATE) then return node
  frontier  $\leftarrow$  a FIFO queue, with node as an element
  reached  $\leftarrow$  {problem.INITIAL}
  while not IS-EMPTY(frontier) do
    node  $\leftarrow$  POP(frontier)
    for each child in EXPAND(problem, node) do
      s  $\leftarrow$  child.STATE
      if problem.IS-GOAL(s) then return child
      if s is not in reached then
        add s to reached
        add child to frontier
  return failure

```


BREADTH-FIRST SEARCH

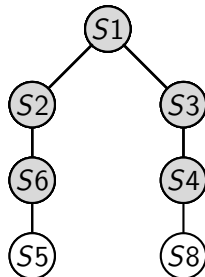
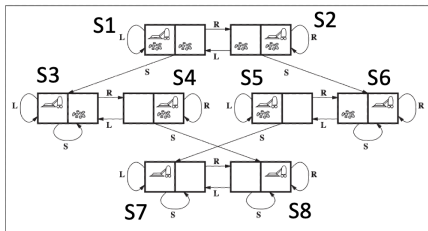
- Steps of the BFS algorithm:
 - Expand** the shallowest node in the frontier
 - Do not add** children in the frontier if the node is already in the frontier or in the list of visited nodes (to avoid loopy paths)
 - Stop** when a goal node is added to the frontier



BREADTH-FIRST SEARCH

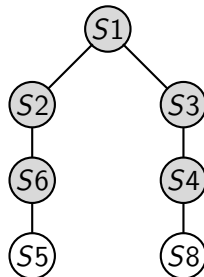
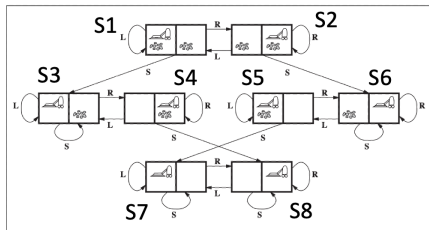
Activity. Consider the vacuum world example from the book. In pairs or small groups, solve the problem using BFS.

- **Expand** the shallowest node in the frontier
- **Do not add** children in the frontier if the node is already in the frontier or in the list of visited nodes (to avoid loopy paths)
- **Stop** when a goal node is added to the frontier



BREADTH-FIRST SEARCH

- **Solution:** S, R, S
- **Cost of the solution:** 3
- **Order of nodes visited:** S1, S2, S3, S6, S4



MEASURING PERFORMANCE

- We can evaluate the performance of an algorithm based on the following criteria:
 - **Completeness**, is the algorithm guaranteed to find a solution provided that one exists?
 - **Optimality**, is the strategy capable of finding the optimal solution?
 - **Time complexity**, how long does the algorithm take to find a solution?
 - **Space complexity**, how much memory is needed to perform the search?

MEASURING PERFORMANCE

- Generally, the size of the space graph is typically used, i.e., $|\mathcal{V}| + |\mathcal{E}|$, the size of the set of vertices set and edges, respectively
- In AI, we use an implicit representation of the graph via the initial state, actions and transition model (as the graph could be infinite)
- Therefore, the following three quantities are used:
 - **Branching factor**, the maximum number of successors of any node: b
 - **Depth** of the shallowest goal node (number of steps from the root): d
 - The **maximum length** of any path in the state space: m

PERFORMANCE OF BFS

- Let us evaluate the performance of BFS.
 - **Completeness:** if the goal node is at some finite depth d , then BFS is **complete** and will find the goal (given that b is finite)
 - **Optimality:** BFS is **optimal** if the path cost is a nondecreasing function of the depth of the node (e.g., all actions have the same cost)
 - **Time complexity:** $\mathcal{O}(b^d)$, assuming a uniform tree where each node has b successors, we generate $b + b^2 + \dots + b^d = \mathcal{O}(b^d)$
 - **Space complexity:** $\mathcal{O}(b^d)$, if we store all expanded nodes, we have $\mathcal{O}(b^{d-1})$ explored nodes in memory and $\mathcal{O}(b^d)$ in the frontier

OUTLINE

- 1 Recap: Asymptotic Analysis
- 2 Formulating a Search Problem
- 3 Breadth-First Search
- 4 Depth-First Search**

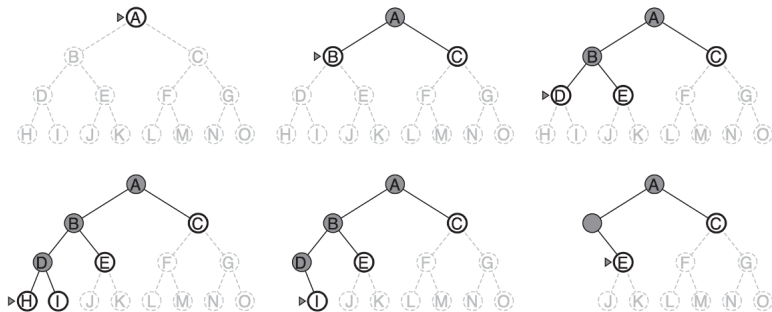
DEPTH-FIRST SEARCH

- Depth-First search is another common search strategy:
 - The root node is expanded first
 - Then, the first (or one at random) successor of the root node is expanded
 - Then, the deepest node in the current frontier is expanded
- This is equivalent to expanding the deepest unexpanded node in the frontier; a **stack (LIFO)** is used for expansion
- Basically, the most recently generated node is chosen for expansion

DEPTH-FIRST SEARCH

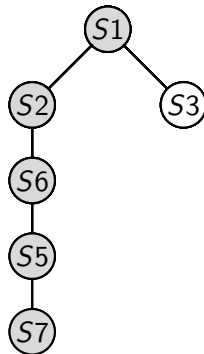
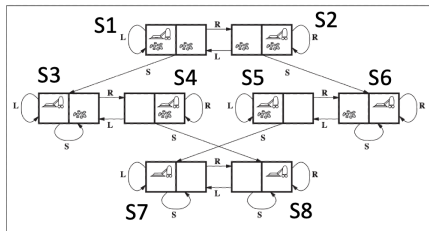
- Steps of the DFS algorithm:
 - **Expand** the deepest node in the frontier
 - **Do not add** children in the frontier if the node is already in the frontier or in the list of visited nodes (to avoid loopy paths)
 - **Stop** when a goal node is visited

DEPTH-FIRST SEARCH



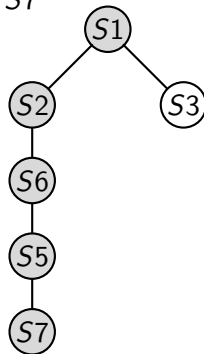
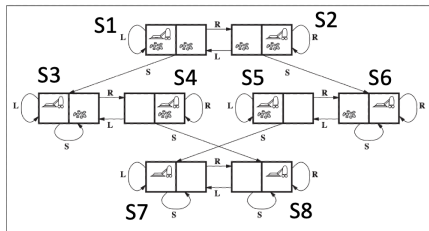
DEPTH-FIRST SEARCH

Activity. Consider the vacuum world example from the book. In pairs or small groups, solve the problem using DFS.



DEPTH-FIRST SEARCH

- **Solution:** R, S, L, S
- **Cost of the solution:** 4
- **Order of nodes visited:** S1, S2, S6, S5, S7



PERFORMANCE OF DFS

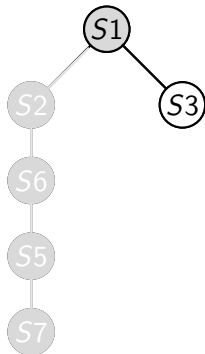
- Let us evaluate the performance of DFS.
 - **Completeness:** DFS **is not complete** if the search space is infinite or if we do not avoid infinite loops; it **is complete** if the search space is finite
 - **Optimality:** DFS **is not optimal** as, e.g., it can expand the left subtree when the goal node is in the first level of the right subtree
 - **Time complexity:** $\mathcal{O}(b^m)$, as it depends on the maximum length of the path in the search space (in general m can be much larger than d)
 - **Space complexity:** $\mathcal{O}(b^m)$, as we store all the nodes from each path from the root node to the leaf node

DEPTH-FIRST SEARCH - VARIATIONS

- Depth-First Search comes with several issues:
 - Not optimal
 - High time complexity
 - High space complexity
- DFS with less memory usage (saving space, reducing complexity)
- Depth-Limited Search, for completeness

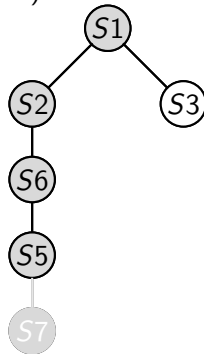
DEPTH-FIRST SEARCH WITH LESS MEMORY USAGE

- Let us consider the example in the image
- Node $S7$ is not a goal node and does not have children
- The next step of DFS would be to expand $S3$
- We can remove the left subtree from memory, since it has been fully explored
- This reduces the space complexity to $O(bm)$
- We need to store a single path along with the siblings for each node on the path



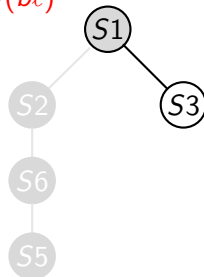
DEPTH-LIMITED SEARCH

- In problems with infinite state spaces, DFS is not complete
- This issue can be mitigated by setting a depth limit ℓ
- This approach is called **depth-limited search** (DLS)
- For $\ell = 3$, we have
- If we choose $\ell < d$,
we add another source of incompleteness
- DLS is not optimal even when $\ell > d$
- However, the time complexity reduces to $\mathcal{O}(b^\ell)$



DLS WITH LESS MEMORY USAGE

- Like before, we can remove the explored paths from memory after we have reached the depth limit ℓ
- By doing that, the space complexity reduces to $O(b\ell)$



COMPARING UNINFORMED SEARCH STRATEGIES

Criterion/Strategy	BFS	DFS	DFS (LMU)	DLS (LMU)
Completeness	Yes*	Yes***	Yes***	Yes, if $\ell \geq d$
Optimality	Yes**	No	No	No
Time	$\mathcal{O}(b^d)$	$\mathcal{O}(b^m)$	$\mathcal{O}(b^m)$	$\mathcal{O}(b^\ell)$
Space	$\mathcal{O}(b^d)$	$\mathcal{O}(b^m)$	$\mathcal{O}(bm)$	$\mathcal{O}(b\ell)$

* If b is finite

** If the path cost is a nondecreasing function of the depth of the node (e.g., all actions have the same cost)

*** If the search space is finite (also, loopy paths are removed)

References



Russell, A. S., and Norvig, P., *Artificial Intelligence A Modern Approach*, 4th Edition. Prentice Hall.

- ➡ Chapter 3 – “Solving Problems by Searching”, Section 3.1 “Problem-Solving Agents” up to and including Section 3.4.6 “Comparing Uninformed Search Algorithms”
- ➡ Appendix A – “Mathematical Background”, Section A.1 “Complexity Analysis and $\mathcal{O}()$ Notation

AIMS OF THE SESSION

You should now be able to:

- Understand the concept of asymptotic analysis
- Formulate a search problem
- Explain the steps involved in Breadth-First Search (BFS), Depth-First Search (DFS) and its variations
- Apply the algorithms to solve search problems



Thank you!