Universitas Siber Asia

MODUL PEMBELAJARAN
PROGRAM STUDI PJJ INFORMATIKA
Learning Module (Program: Informatics)

Mata Kuliah *(Lecture Name)*	:	**Kecerdasan Buatan** *(Artificial Intelligence)*
Bobot SKS *(Credits)*	:	**3**
Dosen Pengembang Modul *(Modul Team)*	:	**Cian Ramadhona Hassolthine, S. Kom. , M. Kom.** **Ega Dioni Putri, S. T. , M. M. G.** *(Informed Search + English translation)*
Dosen Pengampu *(Lecturer for Class IF501)*	:	**Ega Dioni Putri, S. T. , M. M. G.**
Capaian Pembelajaran Mata Kuliah *(Expected Learning Outcomes)*	:	1. Mampu menjelaskan konsep dasar Kecerdasan Buatan - *Student can explain the basic concepts of AI* 2. Mampu menjelaskan konsep masalah dan penyelesaian masalah - *Student can explain the concept of problem and problem solving* 3. Mampu menjelaskan teknik pencarian buta dan pencarian terbimbing - *Student can explain the concept of blind search (uninformed search) and heuristic search (informed search)* 4. Mampu memahami teknik-teknik Kecerdasan Buatan dan cara merepresentasikan masalah - *Student is able to understand the technique of AI and problem representation methods* 5. Mampu memahami sistem JST (Jaringan Syaraf Tiruan) - *Student is able to understand artificial neural network* 6. Mampu memahami algoritma genetika - *Student is able to understand genetic algorithm* 7. Mampu memahami metode sistem pakar - *Student is able to understand expert system* 8. Mampu menjelaskan konsep penalaran ketidakpastian - *Student can explain the concept of uncertain reasoning* 9. Mampu menjelaskan konsep pembelajaran *(learning)* - *Student can explain the concept of learning* 10. Mampu memahami sistem fuzzy - *Student is able to understand fuzzy system* 11. Mampu menjelaskan konsep teknik pembangunan *game* AI - *Student can explain the concept of AI-based game development technique* 12. Mampu membuat program sederhana *board game* - *Student can build a simple board game*

		13. Mampu membuat perancangan *game* AI - *Student is able to design AI-based game* 14. Mampu menjelaskan konsep genre game berbasis AI - *Student is able to explain the concept of AI-based game*
Kompetensi Akhir di Tahap Ini (Sub-CPMK) *(Outcome of This Session)*		Mahasiswa mampu memahami konsep teknik pencarian buta *(blind search)* dan *informed search*
Minggu Perkuliahan Online Ke- *(Session)*		**3**

Teknik Pencarian Buta dan Pencarian Terbimbing

(Informed Search and Uninformed Search Technique)

A. Pencarian Buta *(Blind Search / Uninformed Search)*

Ada beberapa algoritma yang dikategorikan ke dalam teknik pencarian buta.

The following algorithms are categorized as blind search or also called "uninformed search".

- *Breadth First Search (BFS)*
- *Uniform Cost Search (UCS)*
- *Depth First Search (DFS)*
- *Depth Limited Search (DLS)*
- *Iterative Deepening Search (IDS)*
- *Bidirectional Search (BS)*

Terdapat empat komponen yang harus didefinisikan ketika mencari solusi dari sebuah permasalahan pencarian.

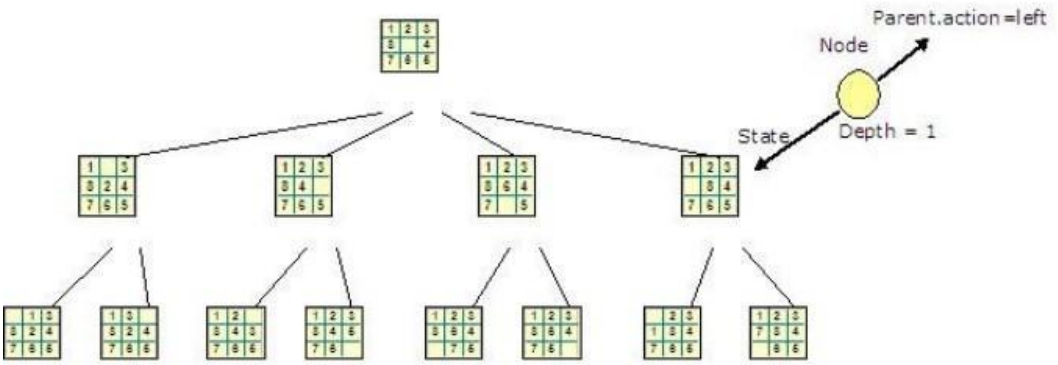
4 - must defined components in a "search" problem solving:

1. *Initial state*
2. *Goal state*
3. *Steps to reach the Goal state*
4. *Cost*

Seluruh jenis algoritma pencarian buta di atas mempunyai strategi masing-masing untuk mencapai goal yang dimulai dari *initial state*. Pohon keputusan *(decision tree)* dipilih sebagai salah satu teknik pencarian untuk mencapai *goal state*.

Gambar di bawah ini memperlihatkan contoh kasus penyelesaian permainan *8-puzzle* dengan mentransformasikan solusi permainan ke dalam topologi pohon.

Each type of blind search algorithm above has its own strategy to reach the goal state starting from the initial state. In this case, the decision tree is selected as one of the searching methods. The tree representation below shows an example of an 8-puzzle game solving.



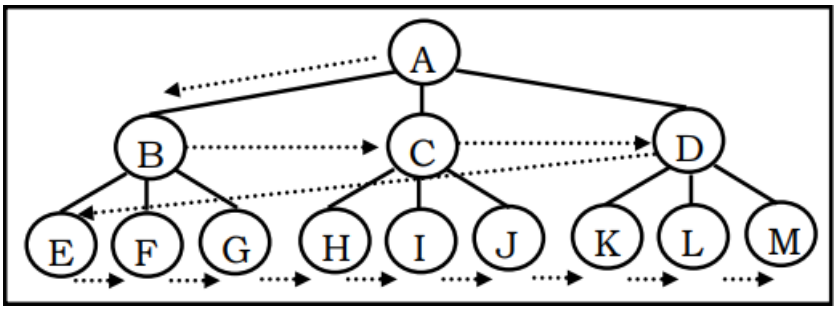
Kumpulan simpul-simpul yang dibentuk tetapi belum disambungkan dengan simpul yang lain dinamakan dengan *fringe*. Setiap elemen dari *fringe* merupakan *left node* (simpul yang ditinggalkan) dari *tree*. Berikut akan dijelaskan algoritma-algoritma yang dikategorikan ke dalam kelas *blind search*.

A collection of formed nodes, but in disconnected condition, is called "fringe". Each element of the fringe is a "left node" from a tree. Some algorithms in blind search are explained in the following.

1) ***Breadth First Search (BFS)***

   Algoritma yang mula-mula menjelajahi simpul akar *(root node)* pada level *n*, kemudian semua simpul anak *(child node / successor)* dari simpul akar di kedalaman berikutnya alias level *n+1*, dilanjutkan dengan mengunjungi semua *successor* dari *successor*. Begitu seterusnya sampai tiba pada *successor* di kedalaman terakhir. *Fringe* dalam BFS merupakan struktur data antrian *First In First Out (FIFO)*.

   Breadth First Search (BFS) is an algorithm that first explores the root node at level n, then all child nodes (successors) of the root node at the next depth (level n+1), followed by visiting all successors of the successors, until arriving at the successor at the last depth. Fringe in BFS is a First In First Out (FIFO) queue data structure.



Keuntungan BFS:

a. Tidak akan menemui jalan buntu
b. Jika ada satu solusi, maka BFS akan menemukannya. Jika ada lebih dari satu solusi, maka solusi minimum akan ditemukan.

Kelemahan BFS:

a. Membutuhkan memori yang cukup banyak karena menyimpan semua simpul dalam satu pohon
b. Membutuhkan waktu yang cukup lama karena akan menguji *n level* untuk mendapatkan solusi pada level yang ke-*(n+1)*

Advantages of BFS:

- *No dead ends*
- *If there is one solution, then BFS will find it. If there is more than one solution, then the minimum solution will be found.*

Disadvantages of BFS:

- *Requires quite a lot of memory because it stores all the nodes in one tree*
- *Takes quite a long time because it will test n levels to get a solution at the (n+1)th level*

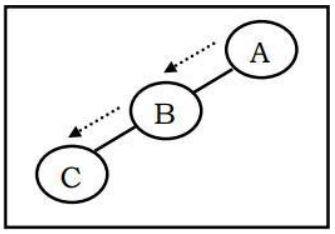
2) ***Uniform Cost Search (UCS)***

Modifikasi dari BFS dengan selalu menjelajahi simpul yang paling sedikit *cost*-nya pada *fringe* menggunakan perhitungan *path cost function g(n)*. Misalnya, biaya (banyaknya langkah) dari initial state ke node n). Node disusun dengan algoritma *queue* untuk menentukan jumlah (biaya) dalam mencapai node n.

Modification of BFS by always exploring the least cost node on the fringe using the path cost function g(n) calculation.

3) ***Depth First Search (DFS)***

Penjelajahan simpul dimulai dari kedalaman yang paling dalam pada *tree*. *Fringe* dalam DFS termasuk struktur data *queue (stack) Last In First Out (LIFO).*



Keuntungan:

a. Membutuhkan memori yang relatif kecil karena hanya simpul-simpul pada lintasan aktif saja yang disimpan
b. Secara kebetulan, metode ini akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan.

Kelemahan:

a. Memungkinkan tidak ditemukannya tujuan yang diharapkan
b. Hanya akan mendapatkan satu solusi pada setiap pencarian

Node exploration starts from the deepest part of the tree. Fringe in DFS includes Last In First Out (LIFO) queue (stack) data structure.

Advantages DFS:

- *Requires relatively small memory, only the nodes on the active path are stored*
- *This method will find a solution without having to test more in the state space.*

Disadvantages DFS:

- *Possibly not finding the expected goal*
- *Will only get one solution in each search*

4) ***Depth Limited Search (DLS)***

Kekurangan algoritma DFS dalam menyediakan *space (memory)* dapat diatasi melalui algoritma ini dengan memberi batasan kedalaman ruang pencarian, misalnya *l,* maka *l* diperlakukan seolah-olah tidak punya *successors*.

The lack of the DFS algorithm in providing space (memory) can be overcome by first determining the depth limit, for example l, that l is treated as if they have no successors.

5) ***Iterative Deepening Depth First Search (IDS)***

Secara umum, strategi algoritma ini digunakan dengan mengkombinasikan algoritma DFS yang mencari *the best depth limit*. Ini dilakukan dengan menambahkan limit dari 0, kemudian 1, 2, dan seterusnya sampai *tujuan* ditemukan.

This algorithm strategy is used by combining the DFS algorithm that searches for the best depth limit. This is done by adding a limit of 0, then 1, 2, and so on until the goal is found.

6) ***Bidirectional Search (BS)***

Ide dari algoritma ini adalah untuk mencari secara bersamaan baik dari *goal* ke *initial state* maupun dari *initial state* ke *goal*, dan berhenti ketika kedua langkah pencarian bertemu di pertengahan pencarian. Dalam algoritma ini, terdapat dua *fringe*, *fringe* pertama digunakan untuk langkah dari *initial state ke goal (forward)* dan *fringe* satu lagi untuk langkah dari *goal ke initial state (backward)*. Setiap *fringe* diimplementasikan dengan algoritma LIFO atau FIFO, tergantung dari strategi pencarian yang digunakan (misalnya Forward=BFS, Backward=DFS).

The idea of this algorithm is to search simultaneously from the goal to the initial state and from the initial state to the goal, and stop when both search steps meet in the middle of the search. In this algorithm, there are two fringes: the first one is used for the step from the initial state to the goal (forward) and the other fringe is used for the step from the goal to the initial state (backward). Each fringe is implemented with the LIFO or FIFO algorithm, depending on the search strategy used (e.g. Forward=BFS, Backward=DFS).

B. Pencarian Terbimbing *(Heuristic Search/Informed Search)*

Tidak seperti teknik pencarian buta, yang hanya murni mengandalkan struktur pada ruang pencarian, teknik pencarian terbimbing atau yang juga disebut pencarian heuristik menggunakan pengetahuan spesifik domain untuk membuat keputusan yang lebih pintar tentang jalur mana yang harus dijelajahi. Hal ini bisa mengurangi jumlah langkah yang dibutuhkan untuk menemukan sebuah solusi.

Penerapan dari teknik pencarian ini cukup banyak kita temui dalam kehidupan sehari-hari, misalnya *search engine* seperti Google, pemecahan permainan *puzzle* atau *sudoku,* pencarian rute terbaik oleh robot navigator, mesin penerjemah, dan

sistem-sistem perencanaan otomatis yang membantu berbagai sektor seperti penjadwalan otomatis, pengontrolan logistik otomatis, dsb. yang seluruhnya ditujukan untuk mengoptimalkan sumber daya dan menekan biaya operasional.

Konsep utama dari teknik pencarian ini meliputi:

- **Heuristik**

  Sebuah penerapan aturan praktis atau estimasi cerdas yang memandu mengarahkan pencarian ke solusi dengan cara yang lebih efisien daripada metode pencarian menyeluruh. Heuristik memperkirakan "biaya" untuk mencapai sasaran solusi dari kondisi tertentu. Makin mendekati biaya yang sebenarnya, makin baik kualitasnya.

- **Fungsi Heuristik *(h(n))***

  Fungsi yang menghitung biaya perkiraan (estimasi) dari suatu simpul tertentu menuju ke simpul tujuan. Misalnya, dalam masalah pencarian jalur pada *grid*, jarak garis lurus (jarak Euclidean) ke tujuan dapat berfungsi sebagai heuristik.

- **Fungsi Evaluasi *(f(n))***

  Algoritma pencarian yang terbimbing / terinformasi seperti A* menggunakan fungsi evaluasi yang menggabungkan heuristik dengan biaya aktual yang dikeluarkan untuk mencapai simpul pada saat tertentu. Hal ini membantu menyeimbangkan eksplorasi jalur yang menjanjikan untuk mencapai tujuan secara efisien.

Beberapa algoritma pencarian terbimbing yang populer antara lain:

1. ***Greedy Best-First Search***

   Algoritma ini hanya menggunakan fungsi heuristik h(n) dan selalu memperluas simpul yang tampak paling dekat dengan tujuan, menurut heuristik. Pencarian Greedy bisa cepat, tetapi tidak selalu optimal dan bisa terjebak dalam “minima lokal”, artinya jalur yang dipilih mungkin bukan jalur terbaik yang mungkin.

2. ***A* (A star) Search***

   Pencarian A* menggunakan fungsi evaluasi $f(n) = g(n) + h(n)$ dengan $g(n)$

adalah biaya untuk mencapai simpul saat ini dan *h(n)* adalah perkiraan biaya untuk mencapai tujuan. Dengan menggabungkan biaya aktual dan perkiraan, A* bertujuan untuk menemukan solusi optimal secara efisien. A* bersifat lengkap (akan menemukan solusi jika ada) dan optimal (menemukan jalur berbiaya terendah), asalkan heuristik *h(n)* dapat diterima (tidak melebih-lebihkan biaya sebenarnya).

3. ***Beam Search***

   Beam search membatasi jumlah simpul yang dijelajahi di setiap tingkat pohon pencarian (tidak semua dicek), menggunakan heuristik untuk mengevaluasi dan menyimpan hanya simpul yang paling menjanjikan (jumlah simpul-simpul disimpan sebagai bilangan yang *fixed* dan disebut *"beam width"*) di setiap level. Meskipun mengurangi penggunaan memori dan waktu pencarian, algoritma ini mengorbankan kelengkapan dan keoptimalan demi efisiensi.

4. ***Iterative Deepening A* Search***

   Algoritma ini menggabungkan *A* search* dan *iterative deepening depth-first search* (IDS), serta lebih efisien dari A* dari segi memori. Sesuai namanya, algoritma ini melakukan serangkaian pencarian dengan teknik depth-first depth-first dengan ambang batas biaya yang terus meningkat, tujuannya untuk menyempurnakan solusi dengan mempertimbangkan lebih banyak jalur secara progresif. Dalam proses mencari solusi, *IDA** menggunakan fungsi heuristik untuk menghitung sisa biaya hingga sampai ke tujuan. cocok untuk ruang pencarian besar.

Keuntungan penggunaan teknik *informed search* meliputi aspek-aspek berikut:

1. **Efisiensi**: dengan informasi heuristik, teknik pencarian terbimbing bisa menghindari eksplorasi jalur yang tidak menjanjikan, membuat penemuan solusi jauh lebih cepat daripada pencarian buta
2. **Optimasi**: algoritma seperti A* dirancang untuk menemukan solusi optimal jika heuristik dapat diterima, memastikan ada tidaknya jalur terbaik menuju *goal state* sejak awal eksekusi.

3. **Skalabilitas**: dengan heuristik yang sesuai, pencarian terbimbing bisa mengakomodir permasalahan yang lebih besar dan kompleks daripada pencarian buta karena lebih terarah tanpa menjalankan komputasi yang tidak perlu.
4. **Adaptabilitas**: heuristik sering disesuaikan untuk domain tertentu yang spesifik sehingga implementasinya dalam pemecahan masalah lebih beragam, mulai dari pencarian rute, permainan puzzle, sampai *games* berbasis AI.

Namun demikian, tentu tetap ada keterbatasan dari teknik pencarian terbimbing. Berikut beberapa di antaranya:

1. **Kualitas heuristik**: efektivitas pencarian sangat tergantung pada kualitas heuristik, bahkan bisa jadi solusi tidak ditemukan karena faktor ini
2. **Kebutuhan memori**: untuk algoritma tertentu seperti A* membutuhkan memori yang signifikan karena menyimpan semua simpul untuk menyimpan informasi terkait riwayat jalur yang paling murah
3. **Beban komputasi berlebih**: dalam sebagian kasus, proses komputasi heuristik dapat menambah faktor kompleksitas, terutama jika heuristik tersebut kompleks dan mahal untuk dikalkulasi.

Informed search, also known as heuristic search, is a strategy in artificial intelligence (AI) that leverages additional information (heuristics) to guide the search process more efficiently toward a goal. Unlike uninformed (or blind) search methods, which rely solely on the structure of the search space, informed search methods use domain-specific knowledge to make smarter decisions about which paths to explore, reducing the number of steps needed to find a solution.

Key Concepts in Informed Search

- ***Heuristic**: A heuristic is a rule of thumb or an estimate that helps guide the search towards the goal. It estimates the "cost" of reaching the goal from a given state, often with no guarantee of being correct. Good heuristics are ones that closely approximate the true cost but are quick to compute.*
- ***Heuristic Function (h(n))**: This is a function that takes a node as input and returns an estimated cost (or distance) to the goal. For example, in a pathfinding problem on a grid, the straight-line distance (Euclidean distance) to the goal might serve as a heuristic.*
- ***Evaluation Function (f(n))**: Some informed search algorithms, like A*, use an evaluation function that combines the heuristic with the actual cost incurred to reach the current node. This helps balance exploration of promising paths with the need to reach the goal efficiently.*

Popular Informed Search Algorithms

1. ***Greedy Best-First Search****: This algorithm uses only the heuristic function h(n) and always expands the node that appears closest to the goal, according to the heuristic. Greedy search can be fast, but it is not always optimal and can get stuck in local minima.*
2. ***A (A-Star) Search****: A* search uses an evaluation function f(n) = g(n) + h(n) , where g(n) is the cost to reach the current node, and h(n) is the estimated cost to the goal. By combining both actual and estimated costs, A* aims to find the optimal solution efficiently. A* is both complete (it will find a solution if one exists) and optimal (it finds the lowest-cost path), provided the heuristic h(n) is admissible (never overestimates the true cost).*
3. ***Beam Search****: Beam search limits the number of nodes it explores at each level of the search tree. It uses a heuristic to evaluate and keep only the most promising nodes (a fixed number called the "beam width") at each level. While it reduces memory usage and search time, it sacrifices completeness and optimality for efficiency.*
4. ***Iterative Deepening A****: This combines the ideas of A* and iterative deepening depth-first search (IDS). It performs a series of depth-first searches with increasing cost thresholds, refining the solution by considering progressively more paths. It's useful for large search spaces where memory constraints are a concern.*

Advantages of Informed Search

- **Efficiency***: By using heuristic information, informed search algorithms can avoid exploring unpromising paths, often making the search much faster than uninformed approaches like breadth-first or depth-first search.*
- **Optimality***: Algorithms like A* are designed to find the optimal solution if the heuristic is admissible, ensuring the best path to the goal is found when feasible.*
- **Scalability***: With appropriate heuristics, informed search algorithms can handle larger and more complex problems than uninformed searches, as they are more directed and can bypass unnecessary computations.*
- **Adaptability***: Heuristics can often be customized for specific domains, which makes informed search applicable to a wide range of problems, including route finding, puzzle-solving, and game AI.*

Limitations of Informed Search

- **Heuristic Quality***: The effectiveness of informed search depends on the quality of the heuristic. Poor heuristics can lead to inefficient search or incorrect solutions (in cases where the heuristic is not admissible).*
- **Memory Requirements***: Algorithms like A* can consume significant memory, as they store nodes to keep track of the least-cost paths, which can be challenging for large-scale problems.*
- **Computational Overhead***: In some cases, computing the heuristic may add complexity, especially if the heuristic itself is complex or expensive to calculate.*

Applications of Informed Search

- ***Pathfinding and Navigation***
  In robotics and game AI, informed search is used for real-time pathfinding and obstacle avoidance, helping characters or robots find the most efficient route to a target.
- ***Puzzle Solving***
  Informed search is applied to puzzles like the 8-puzzle, 15-puzzle, and Sudoku, where heuristics guide the search to achieve optimal solutions.
- ***Speech and Language Processing***
  Beam search and other informed algorithms are used in tasks like speech recognition and machine translation, where paths are evaluated based on linguistic heuristics to improve accuracy.
- ***Automated Planning***
  In planning problems, such as in logistics or scheduling, informed search helps to optimize resources and minimize costs.

Informed search is a powerful approach in AI, enabling intelligent decision-making that leads to faster, more efficient, and often optimal solutions to complex problems. The key lies in the careful selection and design of heuristics that reflect the problem's unique structure and requirements.

References:

https://lmsspada.kemdikbud.go.id/

https://www.geeksforgeeks.org/

Russell, Stuart J. and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 4th edition.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3th edition.

Compilation of online sources, such as teaching materials from domestic and foreign campuses, domestic and foreign learning institutes, and learning modules that are open to the public.