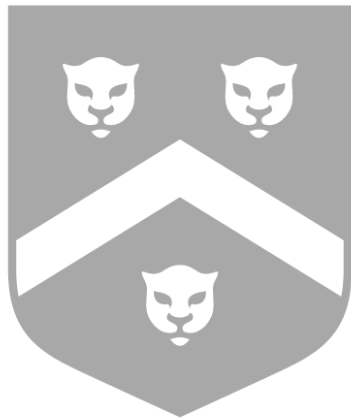


Classical AI



• Professor Frank Kreimendahl

School of Computing and Data Science
Wentworth Institute of Technology

May 15, 2023



Search

Formalization
Representation

Basic
Algorithms

Search

Formalizing Problem Solving

Search

Formalization
Representation

Basic
Algorithms

State: hypothetical world state

Operations: actions that modify world

Goal: desired state or test

The major features of the program that are worthy of discussion are:

1. The recursive nature of its problem-solving activity.
2. The separation of problem content from problem-solving technique as a way of increasing the generality of the program.
3. The two general problem-solving techniques that now constitute its repertoire: means-ends analysis, and planning.

Newell, Shaw, and Simon: “Report On A General Problem-Solving Program”, 1959



Representation

Search

Formalization

Representation

Basic

Algorithms

SampleWorld: first assignment

SW state space

SW search space



Search

Basic Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

Basic Algorithms

-First Search

Search

Basic
Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

-
- 1: $Open \leftarrow$ ordered list containing initial state
 - 2: **while true do**
 - 3: **if** $Open$ is empty **then**
 - 4: **return** failure
 - 5: $Node \leftarrow Open.Pop()$
 - 6: **if** $Node$ is goal **then**
 - 7: **return** $Node$ (or path to $Node$)
 - 8: **else**
 - 9: $Children \leftarrow$ Expand($Node$)
 - 10: Add $Children$ to front of $Open$
-

DFS Evaluation

Search

Basic
Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

Assume branching factor b and solution depth d

Completeness:

Time:

Space:

Admissibility:

Breadth-First Search

Search

Basic
Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

-
- 1: $Open \leftarrow$ ordered list containing initial state
 - 2: **while true do**
 - 3: **if** $Open$ is empty **then**
 - 4: **return** failure
 - 5: $Node \leftarrow Open.Pop()$
 - 6: **if** $Node$ is goal **then**
 - 7: **return** $Node$ (or path to $Node$)
 - 8: **else**
 - 9: $Children \leftarrow Expand(Node)$
 - 10: Add $Children$ to end of $Open$
-

BFS Evaluation

Search

Basic
Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

Assume branching factor b and solution depth d

Completeness:

Time:

Space:

Admissibility:

Uniform-Cost Search

Search

Basic Algorithms

DFS
BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

-
- 1: $Open \leftarrow$ priority queue containing initial state
 - 2: **while** true **do**
 - 3: **if** $Open$ is empty **then**
 - 4: **return** failure
 - 5: $Node \leftarrow Open.Pop()$
 - 6: **if** $Node$ is goal **then**
 - 7: **return** $Node$ (or path to $Node$)
 - 8: **else**
 - 9: $Children \leftarrow \text{Expand}(Node)$
 - 10: Add $Children$ to $Open$, sorted by path cost
-



Search

Basic
Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

- Check for cycles with *Node*'s ancestors
- Maintain **closed** list to detect duplicates

Comparisons

Search

Basic Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

Algorithm	Time	Space	Complete	Admissible
DFS	b^m	bm	if $m \geq d$	no
BFS	b^d	b^d	yes	if ops cost 1
Uniform-cost	b^d	b^d	yes	yes

b : branching factor

d : solution depth

m : maximum explored depth

Resource Requirements for Solutions

Search

Basic Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

Assume $b = 10$, 100k nodes/sec, 100 bytes/node

Sol. depth	Nodes	Time	Space
1	11	.11 ms	1.1Kb
2	111	1.1 ms	11Kb
4	11,111	.11 sec	1Mb
6	10^6	11 sec	111Mb
8	10^8	18 min	11Gb
10	10^{10}	31 hours	1Tb
12	10^{12}	128 days	111Tb
14	10^{14}	35 years	11Pb

Search Tradeoffs

Search

Basic
Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

BFS is complete and admissible, but uses b^d space

DFS uses bd space, but is not admissible and only complete if $m > d$

How can we get the best of both algorithms?

Iterative Deepening Search

Search

Basic
Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

-
-
- 1: **for** $d = 1$ to ∞ **do**
 - 2: DFS to level d
 - 3: **if** DFS succeeds **then**
 - 4: **return** solution
-

IDS Evaluation

Search

Basic
Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

Assume branching factor b and solution depth d

Completeness:

Time:

Space:

Admissibility:

Runtime

Search

Basic Algorithms

DFS

BFS

Uniform-Cost

Comparisons

Requirements

Search

Search Tradeoffs

IDS

Runtime

How can this be efficient? This generates nodes near the start many times.

Intuition: Because of branching, most of the generated nodes are at the same depth as the goal node.

$$\text{generated nodes} = b^d + 2b^{d-1} + \dots + (d-1)b^2 + db$$