Chapter S:II (continued)

II. Basic Search Algorithms

- ❑ Systematic Search
- ❑ Graph Search Basics
- ❑ Depth-First Search
- ❑ Backtracking
- ❑ Breadth-First Search
- ❑ Uniform-Cost Search

Breadth-First Search (BFS) [UCS]

Breadth-first search is an uninformed, systematic search strategy.

BFS characteristics:

- Nodes at upper levels in G are preferred.
- Node expansion happens in levels of equal depth.
- Terminates (on locally finite graphs) with a solution, if one exists.
- Determines a goal node that is closest to the start node s, measured in the number of edges on a solution path.

Remarks:

- Operationalization of BFS: The OPEN list is organized as a queue, i.e., nodes are explored in a FIFO (first in first out) manner. [OPEN list DFS] [OPEN list UCS]
- By enqueuing newly generated successors in OPEN (insertion at the tail) instead of pushing them (insertion at the head), DFS is turned into BFS.

Breadth-First Search

Algorithm: BFS

Input: s. Start node representing the initial problem.
successors(n). Returns the successors of node n.
$\star(n)$. Predicate that is *True* if n is a goal node.
$\perp(n)$. Predicate that is *True* if n is a dead end.

Output: A goal node or the symbol *Fail*.

Breadth-First Search

```
BFS($s$, $successors$, $\star$, $\bot$)

1.  push($s$, OPEN);
2.  LOOP
3.    IF ((OPEN $= \emptyset$) OR memory_exhausted()) THEN RETURN(Fail);
4.    $n$ = pop(OPEN);
      push($n$, CLOSED);
5.    FOREACH $n'$ IN successors($n$) DO      // Expand $n$.
        enqueue($n'$, OPEN);      // Insert node at the end of OPEN.
        set_backpointer($n'$, $n$);
        IF $\star(n')$ THEN RETURN($n'$);



      ENDDO
6.  ENDLOOP
```

Breadth-First Search

```
BFS($s$, successors, $\star$, $\perp$)

  1.  push($s$, OPEN);
  2.  LOOP
  3.    IF ((OPEN $= \emptyset$) OR memory_exhausted()) THEN RETURN(Fail);
  4.    $n$ = pop(OPEN);
        push($n$, CLOSED);
  5.    FOREACH $n'$ IN successors($n$) DO    // Expand $n$.
          enqueue($n'$, OPEN);    // Insert node at the end of OPEN.
          set_backpointer($n'$, $n$);
          IF $\star(n')$ THEN RETURN($n'$);
          IF $\perp(n')$
          THEN
            remove_last(OPEN);
            cleanup_closed();
          ENDIF
        ENDDO
  6.  ENDLOOP
```

Breadth-First Search

Discussion

BFS issue:

- Unlike depth-first search, BFS has to store the explored part of the graph completely. Q. Why?

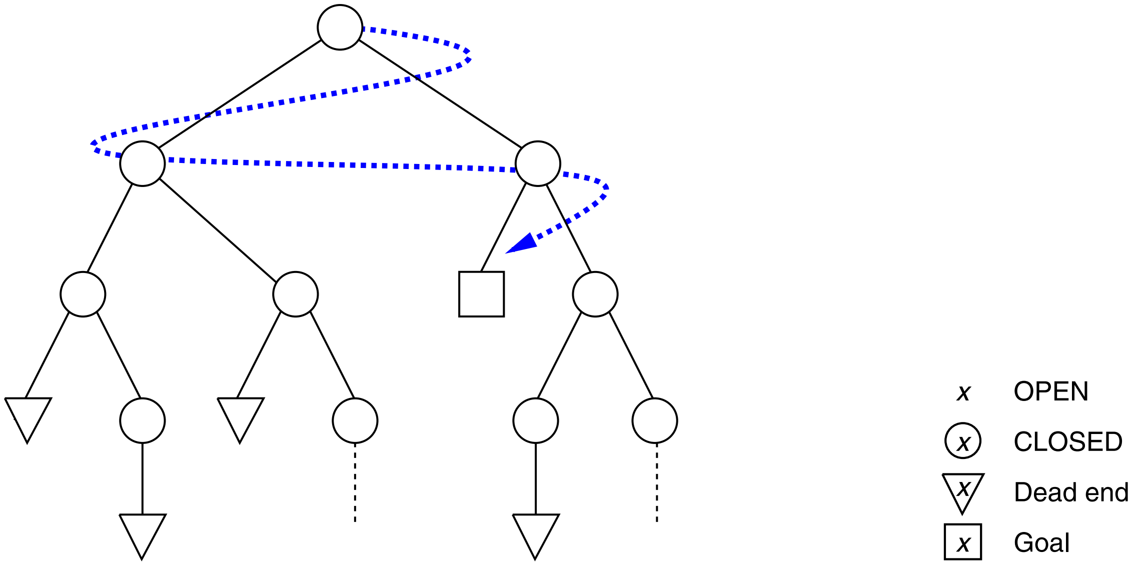
Breadth-First Search

Discussion

BFS issue:

- Unlike depth-first search, BFS has to store the explored part of the graph completely. Q. Why?

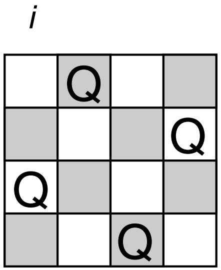
Breadth-first search can be the favorite strategy in certain situations:

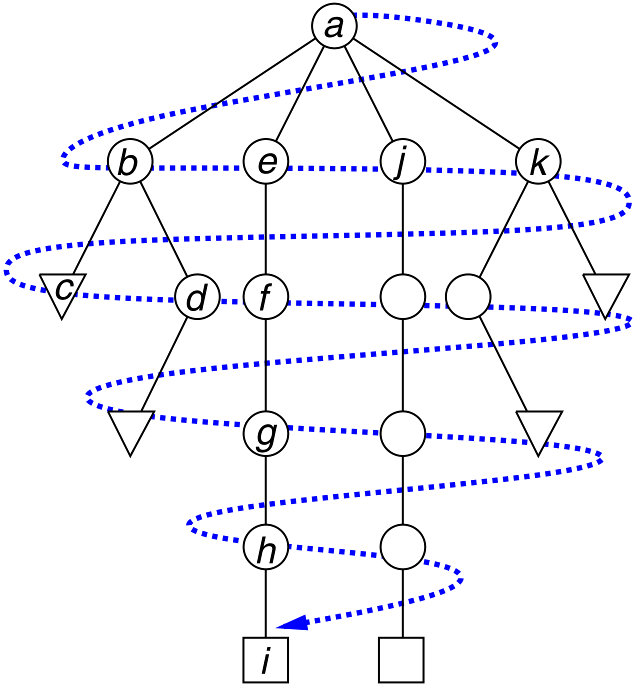


Breadth-First Search

Example: 4-Queens Problem

BFS node processing sequence:



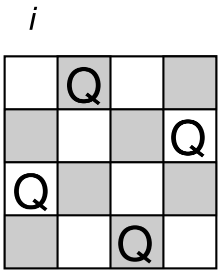


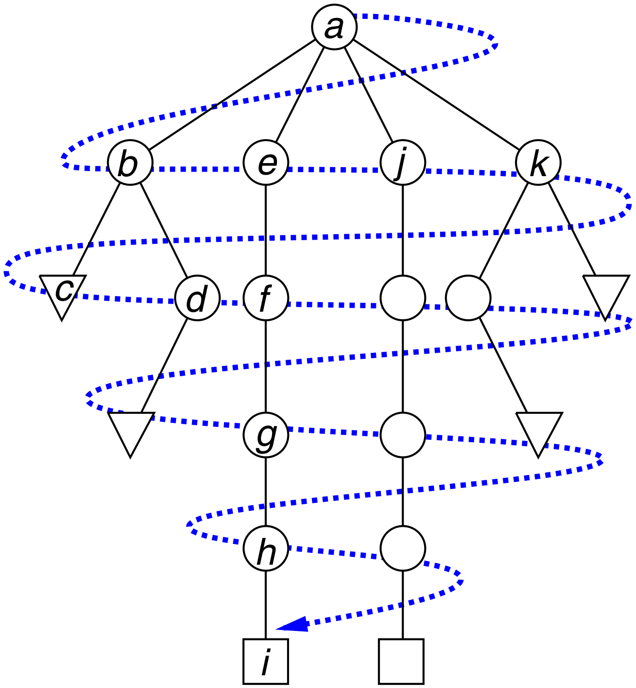
x	OPEN
○ *x*	CLOSED
▽ *x*	Dead end
□ *x*	Goal

Breadth-First Search

Example: 4-Queens Problem

BFS node processing sequence:





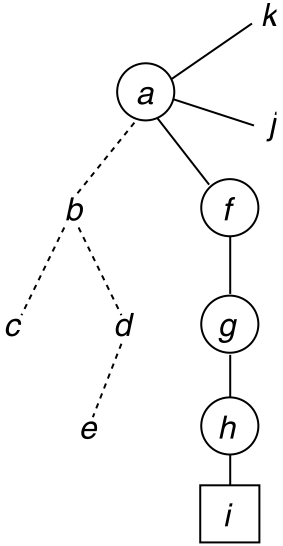
x OPEN

ⓧ CLOSED

▽ Dead end

□ Goal

Compare to the DFS strategy:



Uniform-Cost Search [BFS]

Uniform-cost search is an uninformed (systematic) search strategy.

Uniform-cost search characteristics:

- ❑ Node expansion happens in levels of equal costs:

  A node n with cost $C_{P_{s-n}}$ will not be expanded as long as a non-expanded node n' with $C_{P_{s-n'}} < C_{P_{s-n}}$ resides on the OPEN list.

≈ Application of the BFS strategy to solve optimization problems.

Remarks:

- Operationalization of uniform-cost search: The OPEN list is organized as a heap, and nodes are explored wrt. the cheapest cost. [OPEN list DFS] [OPEN list BFS]
- The notation $C_{P_{s-n}}$ is an abbreviation for $C_{P_{s-n}}(s)$ and (in this uniform-cost search) defines the sum cost of a path from s to n. [S:III Graph Search Basics]
- Uniform-cost search is also called cheapest-first search.

Uniform-Cost Search

Uniform-Cost Search for Optimization: Generic

Setting:

- The search space graph contains several solution paths.

Task:

- Determine the cheapest path from s to some goal $\gamma \in \Gamma$.

Approach:

- Continue search with the cheapest partial solution obtained so far.
- Continue search only until the costs of the partial solutions exceed the currently optimum cost. Keyword: *Early Pruning*

Prerequisite:

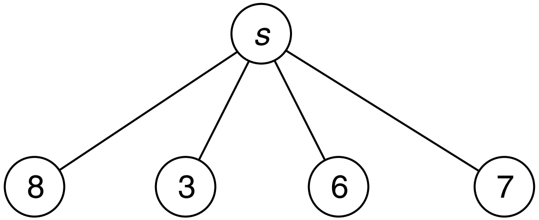
- The accumulated cost on each path increase monotonically.

Uniform-Cost Search

Uniform-Cost Search for Optimization: Example

Determine the minimum column sum of a matrix:

8	3	6	7
6	5	9	8
5	3	7	8
1	2	4	6

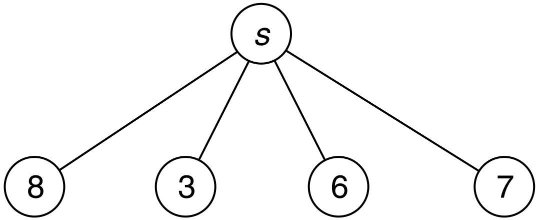


Uniform-Cost Search

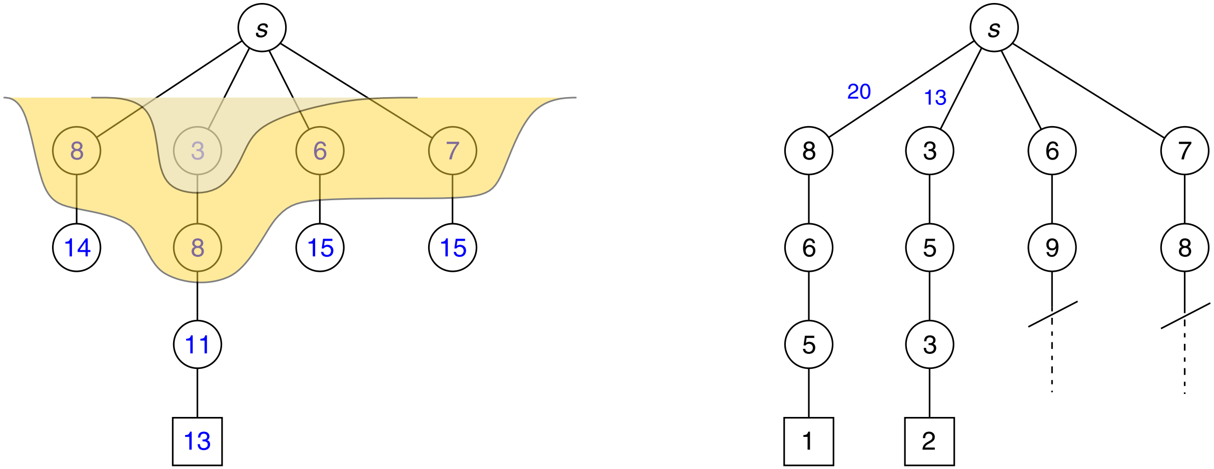
Uniform-Cost Search for Optimization: Example

Determine the minimum column sum of a matrix:

8	3	6	7
6	5	9	8
5	3	7	8
1	2	4	6



Comparison of uniform-cost search (left) and DFS (right):


s
8
3
6
7
14
8
15
15
11
13
s
20
13
8
3
6
7
6
5
9
8
5
3
1
2