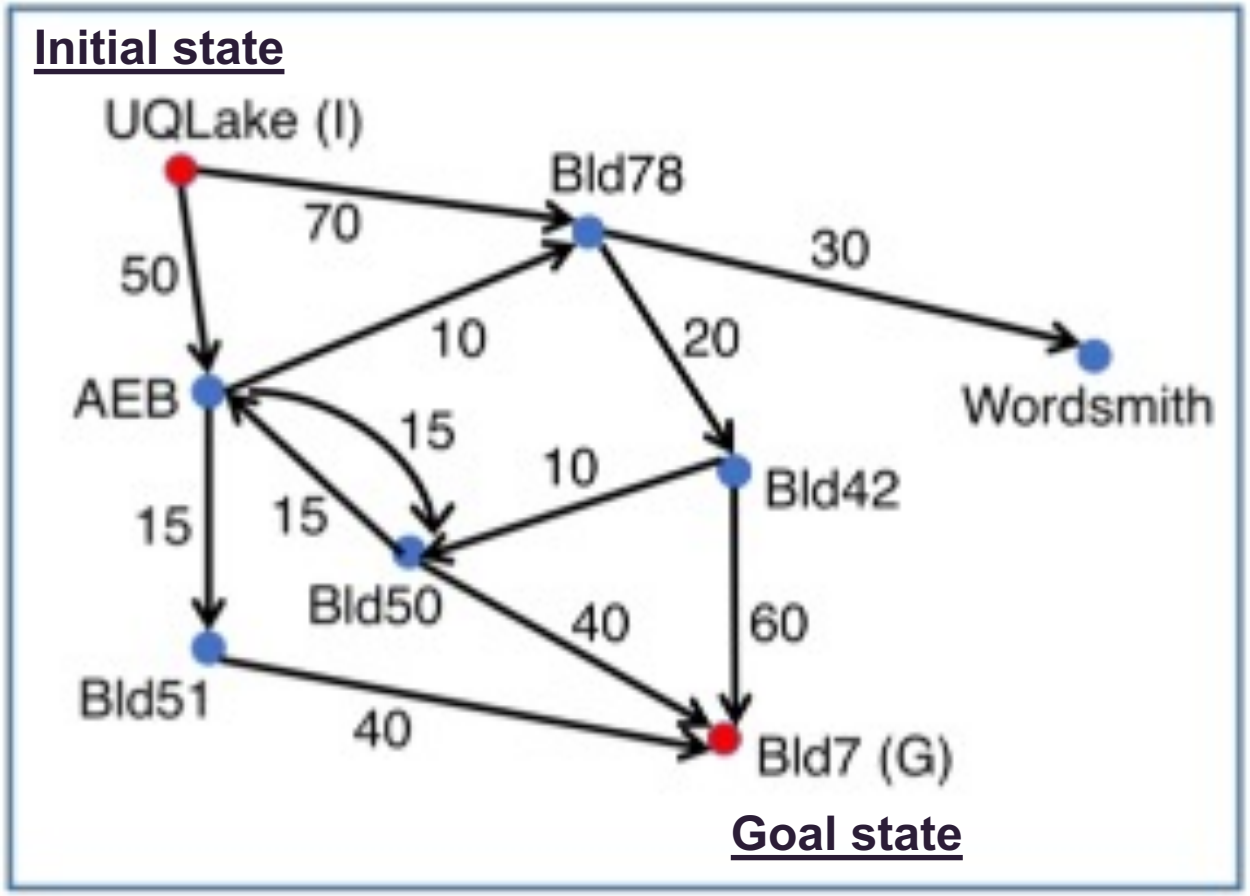
UQ-Map Search Example Problem



Heuristic values (to Bld7)
h(UQLake) = 100
h(Bld78) = 50
h(AEB) = 53
h(Wordsmith) = 1000
h(Bld42) = 50
h(Bld50) = 38
h(Bld51) = 30
h(Bld7) = 0

Searching with Priority Queues, ordered by f(n)
("best-first-search" in R&N)

Pseudocode

```
function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node ← NODE(STATE=problem.INITIAL)
  frontier ← a priority queue ordered by f, with node as an element
  reached ← a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s] ← child
        add child to frontier
  return failure

function EXPAND(problem, node) yields nodes
  s ← node.STATE
  for each action in problem.ACTIONS(s) do
    s' ← problem.RESULT(s, action)
    cost ← node.PATH-COST + problem.ACTION-COST(s, action, s')
    yield NODE(STATE=s', PARENT=node, ACTION=action, PATH-COST=cost)
```

Figure 3.7 The best-first search algorithm, and the function for expanding a node. The data structures used here are described in Section 3.3.2. See Appendix B for **yield**.

Example Python Code

Best-first-search python code, priority queue ordered by $f(n)$ (from R&N 4Ed)

```python
def best_first_search(problem, f):
    "Search nodes with minimum f(node) value first."
    node = Node(problem.initial)
    frontier = PriorityQueue([node], key=f)
    reached = {problem.initial: node}
    while frontier:
        node = frontier.pop()
        if problem.is_goal(node.state):
            return node
        for child in expand(problem, node):
            s = child.state
            if s not in reached or child.path_cost < reached[s].path_cost:
                reached[s] = child
                frontier.add(child)
    return failure
```

52

Priority Queue ordered by f(n)

UCS:

f(n) = g(n)

Path-cost from root to node n

GBFS:

f(n) = h(n)

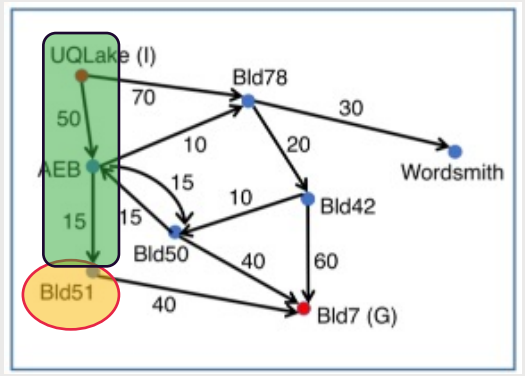
Estimated cost from node n to goal

A*:

f(n) = g(n) + h(n)

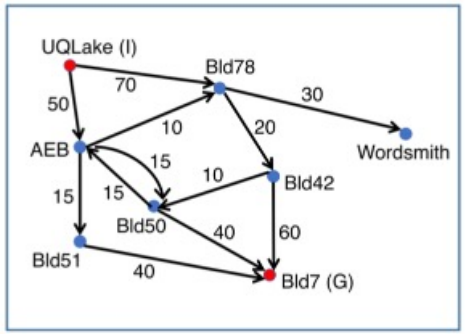
Example costs for node Bld51 (via UQLake → AEB):

g(n) = 50 + 15 = 65
h(n) = 30
g(n) + h(n) = 65 + 30 = 96



Heuristic values (to Bld7)
h(UQLake) = 100
h(Bld78) = 50
h(AEB) = 53
h(Wordsmith) = 1000
h(Bld42) = 50
h(Bld50) = 38
h(Bld51) = 30
h(Bld7) = 0

UCS



UQL

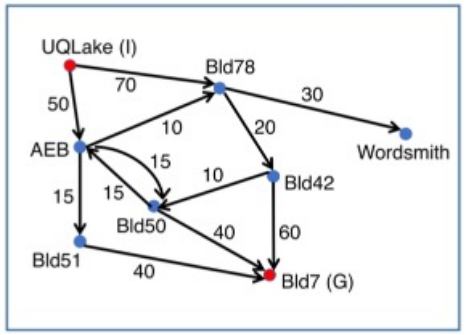
Frontier (priority queue)
Nodes with priority f(n) = g(n)

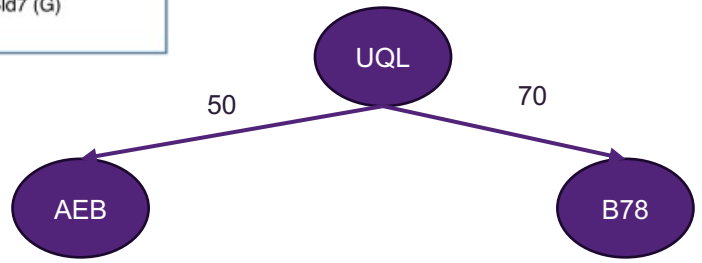
Node	f(n)=g(n)
UQL	0

Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0

UCS





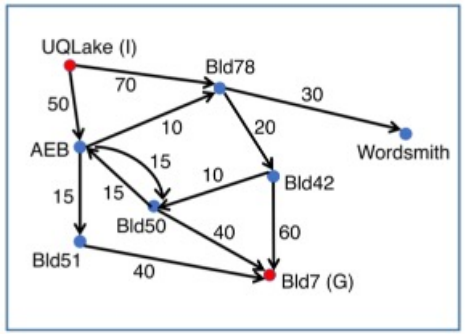
Frontier (priority queue)
Nodes with priority f(n) = g(n)

Node	f(n)=g(n)
~~UQL~~	~~0~~
AEB	50
B78	70

Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0
AEB	50
B78	70

UCS

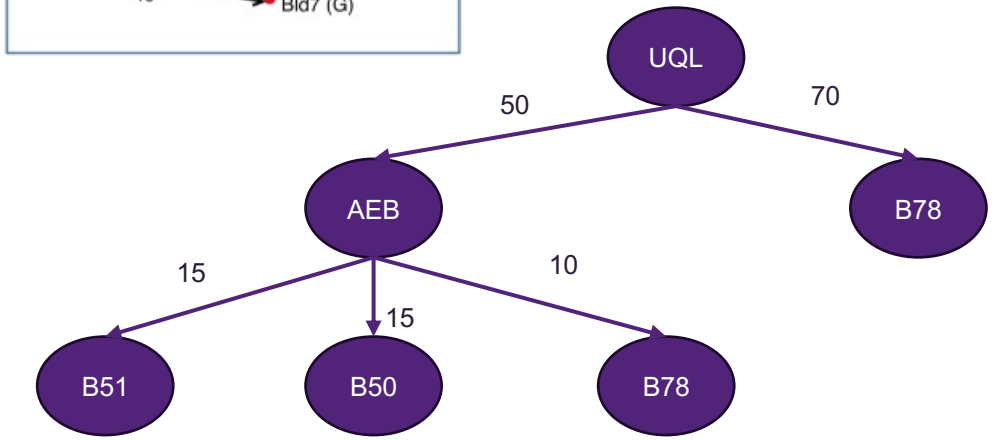


Frontier (priority queue)
Nodes with priority f(n) = g(n)

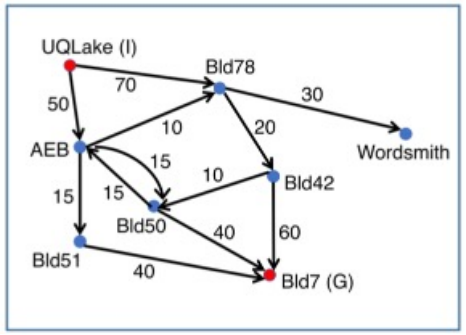
Node	f(n)=g(n)
~~UQL~~	~~0~~
~~AEB~~	~~50~~
B78	70
B51	65
B50	65
B78	60

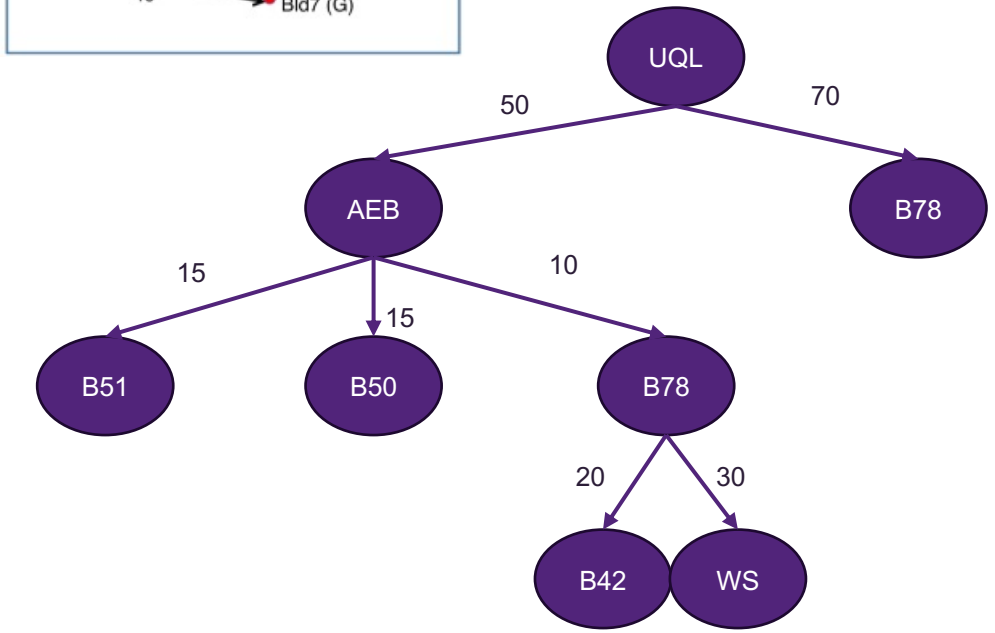
Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0
AEB	50
B78	~~70~~ 60
B51	65
B50	65



UCS





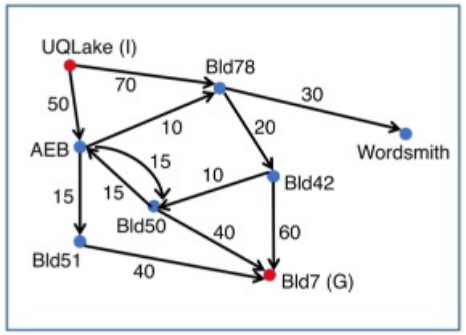
Frontier (priority queue)
Nodes with priority f(n) = g(n)

Node	f(n)=g(n)
~~UQL~~	~~0~~
~~AEB~~	~~50~~
B78	70
B51	65
B50	65
~~B78~~	~~60~~
B42	80
WS	90

Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0
AEB	50
B78	~~70~~ 60
B51	65
B50	65
B42	80
WS	90

UCS

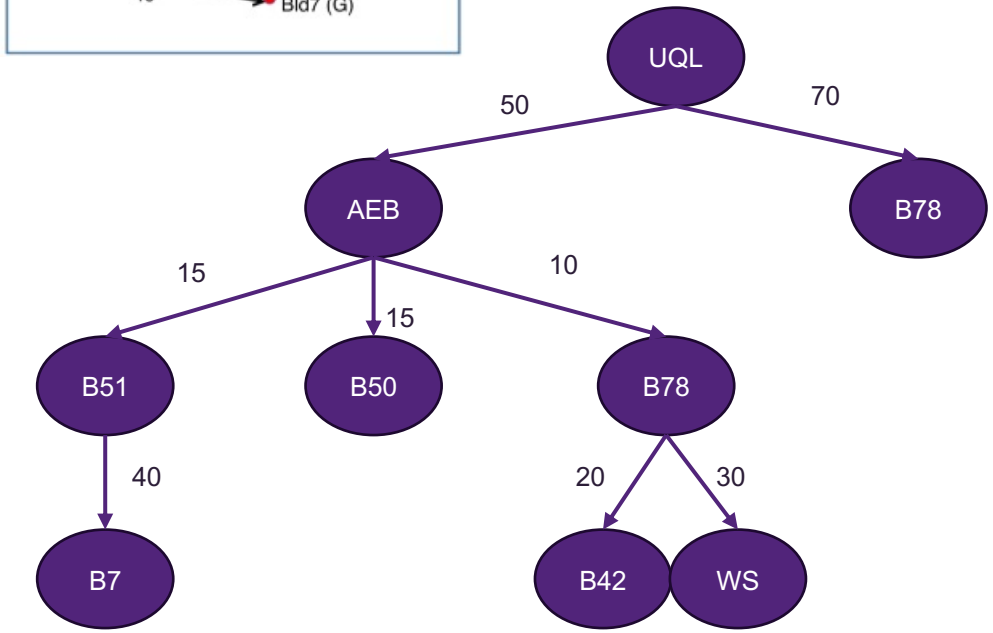


Frontier (priority queue)
Nodes with priority f(n) = g(n)

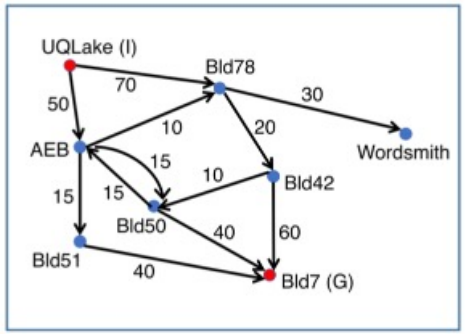
Node	f(n)=g(n)
~~UQL~~	~~0~~
~~AEB~~	~~50~~
B78	70
~~B51~~	~~65~~
B50	65
~~B78~~	~~60~~
B42	80
WS	90
B7	105

Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0
AEB	50
B78	~~70~~ 60
B51	65
B50	65
B42	80
WS	90
B7	105



UCS

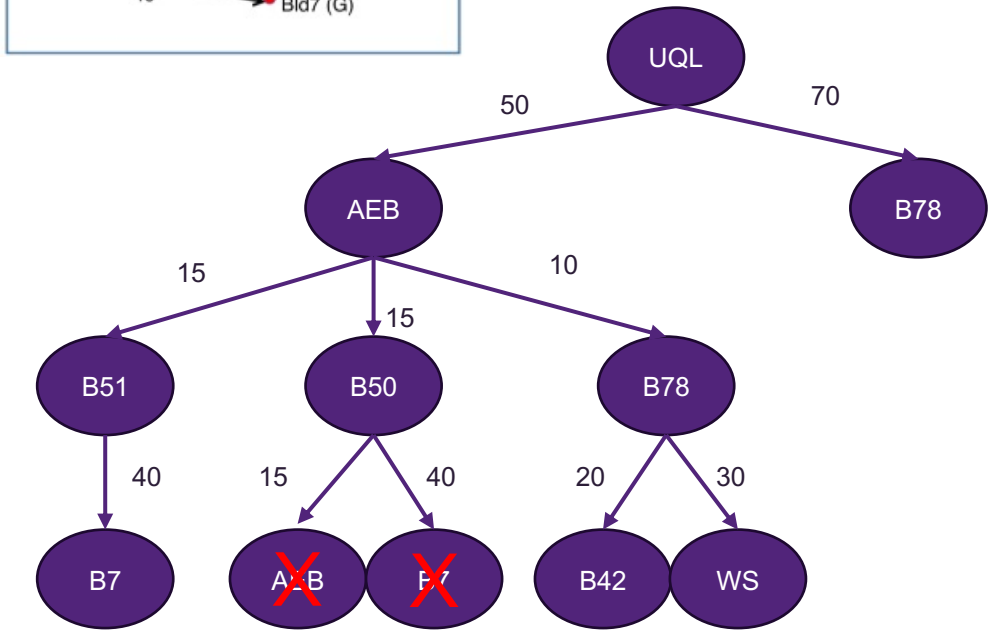


Frontier (priority queue)
Nodes with priority f(n) = g(n)

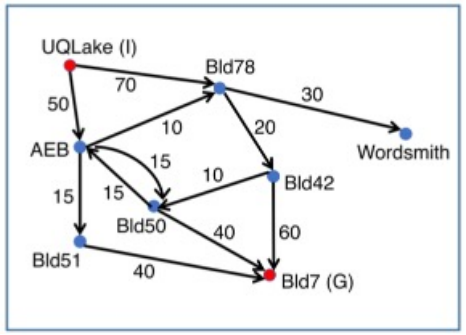
Node	f(n)=g(n)
~~UQL~~	~~0~~
~~AEB~~	~~50~~
B78	70
~~B51~~	~~65~~
~~B50~~	~~65~~
~~B78~~	~~60~~
B42	80
WS	90
B7	105

Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0
AEB	50
B78	~~70~~ 60
B51	65
B50	65
B42	80
WS	90
B7	105



UCS

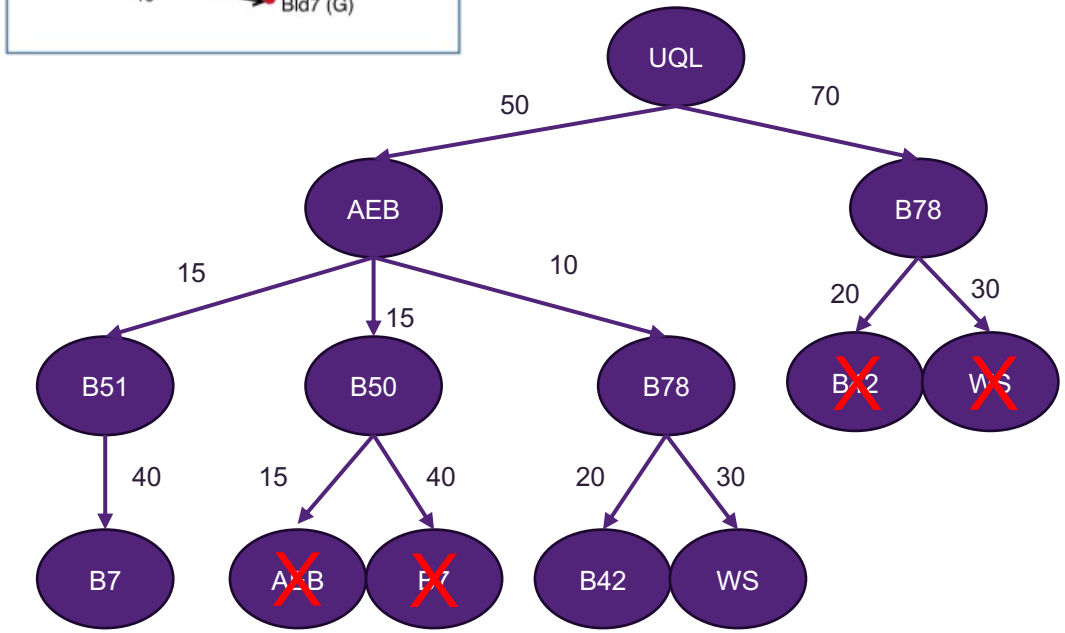


Frontier (priority queue)
Nodes with priority f(n) = g(n)

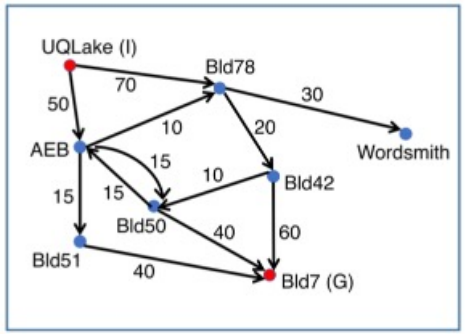
Node	f(n)=g(n)
~~UQL~~	~~0~~
~~AEB~~	~~50~~
~~B78~~	~~70~~
~~B51~~	~~65~~
~~B50~~	~~65~~
~~B78~~	~~60~~
B42	80
WS	90
B7	105

Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0
AEB	50
B78	~~70~~ 60
B51	65
B50	65
B42	80
WS	90
B7	105



UCS

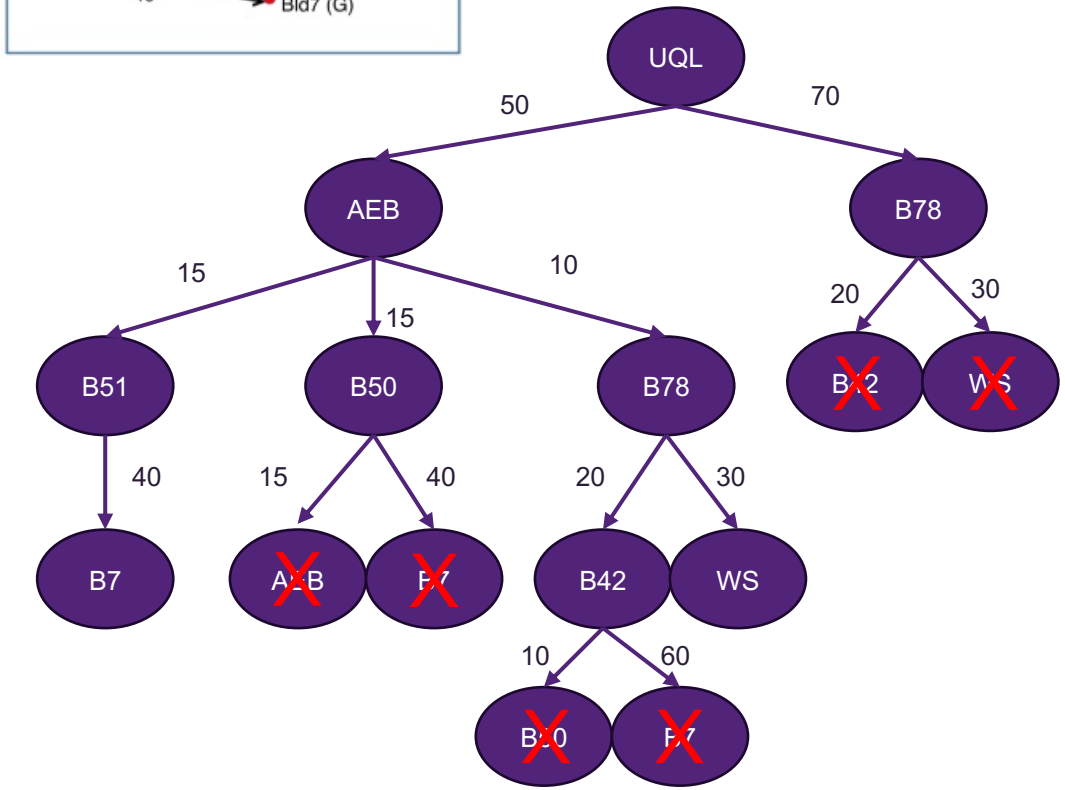


Frontier (priority queue)
Nodes with priority f(n) = g(n)

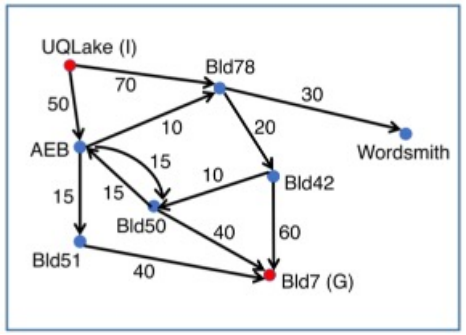
Node	f(n)=g(n)
~~UQL~~	~~0~~
~~AEB~~	~~50~~
~~B78~~	~~70~~
~~B51~~	~~65~~
~~B50~~	~~65~~
~~B78~~	~~60~~
~~B42~~	~~80~~
WS	90
B7	105

Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0
AEB	50
B78	~~70~~ 60
B51	65
B50	65
B42	80
WS	90
B7	105



UCS

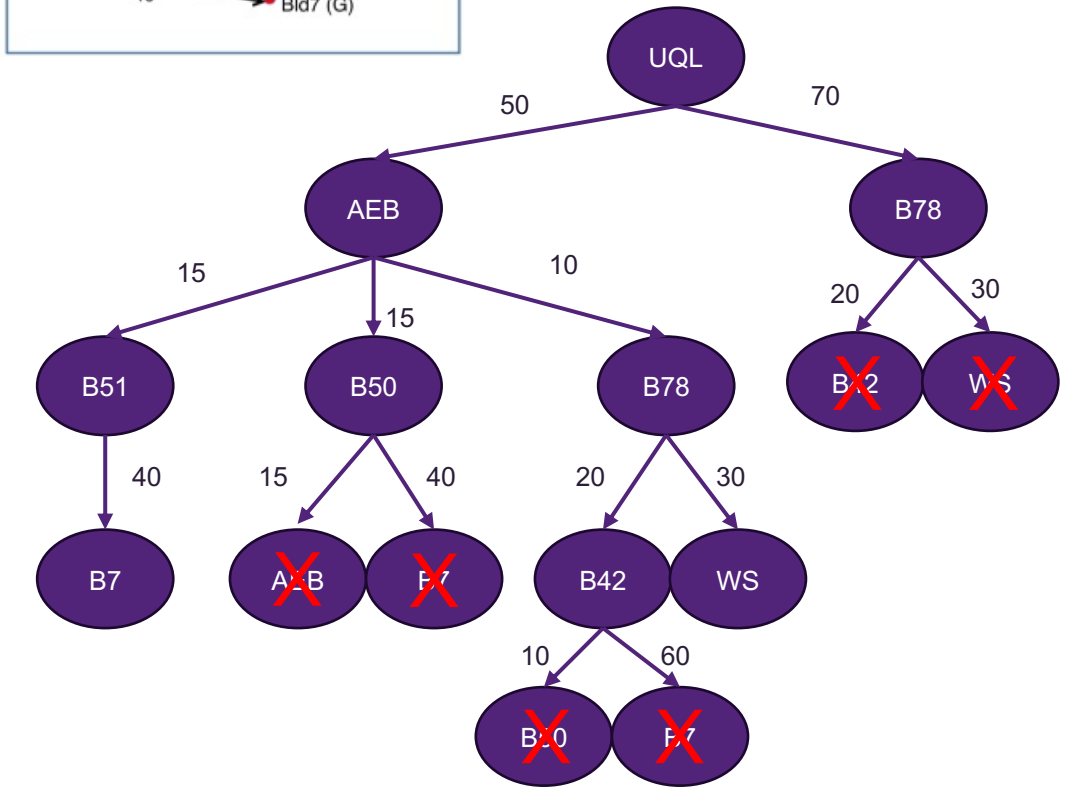


Frontier (priority queue)
Nodes with priority f(n) = g(n)

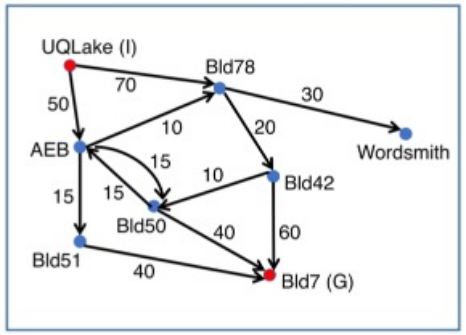
Node	f(n)=g(n)
~~UQL~~	~~0~~
~~AEB~~	~~50~~
~~B78~~	~~70~~
~~B51~~	~~65~~
~~B50~~	~~65~~
~~B78~~	~~60~~
~~B42~~	~~80~~
~~WS~~	~~90~~
B7	105

Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0
AEB	50
B78	~~70~~ 60
B51	65
B50	65
B42	80
WS	90
B7	105



UCS



Frontier (priority queue)
Nodes with priority f(n) = g(n)

Node	f(n)=g(n)
~~UQL~~	~~0~~
~~AEB~~	~~50~~
~~B78~~	~~70~~
~~B51~~	~~65~~
~~B50~~	~~65~~
~~B78~~	~~60~~
~~B42~~	~~80~~
~~WS~~	~~90~~
~~B7~~	~~105~~

Visited set/dictionary
Key = state
Value = path-cost, g(n)

State	g(n)
UQL	0
AEB	50
B78	~~70~~ 60
B51	65
B50	65
B42	80
WS	90
B7	105