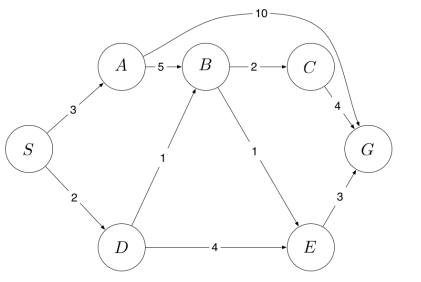
**Question 1: Find the path from S to G using the UCS algorithm for the graph below. Please show the transformation of fringe and the path found**

**To find the path from S to G using the UCS (Uniform Cost Search) algorithm, we'll maintain a priority queue called the fringe to explore nodes in order of their cumulative cost from the start node. We'll update the fringe as we explore nodes.**

Fringe at Step 1: queue [(2, 'D'), (3, 'A')]

(Expand node D and A from node S, prioritize the lowest cost path)

Fringe at Step 2: queue [(3, 'A'), (3, 'B'), (6, 'E')]

(Expand node B and E from node D, prioritize the lowest cost path)

Fringe at Step 4: queue [(3, 'B'), (6, 'E'), (8, 'B'), (13, 'G')]

(Expand node B and G from node A, prioritize the lowest cost path)

Fringe at Step 4: queue [(3, 'B'), (6, 'E'), (13, 'G')]

(Update node B)

Fringe at Step 5: queue [(4, 'E'), (5, 'C'), (6, 'E'), (13, 'G')]

(Expand node E and C from node B, prioritize the lowest cost path)

Fringe at Step 5: queue [(4, 'E'), (5, 'C'), (13, 'G')]

(Update node E)

Fringe at Step 6: queue [(5, 'C'), (7, 'G'), (13, 'G')]

(Expand node G from node E, prioritize the lowest cost path)

Fringe at Step 6: queue [(5, 'C'), (7, 'G')]

(Update node G)

Fringe at Step 7: queue [(7, 'G'), (9, 'G')]

(Expand node G from node C, prioritize the lowest cost path)

Fringe at Step 7: queue [(7, 'G')]

(Update node G)

Fringe at Step 8: queue []

Goal reach: optimal Path: ['S', 'D', 'B', 'E', 'G'] Cost: 7

The final path from S to G is: S -> D -> B -> E -> G

**Given the maze as shown below. Bold lines represent impassable wall. Find your way from s to g with the search strategies as below. Show the expanded order of each cells according to format <b1,b2,…,bN>, with bI is the expanded cell.**

**a. BFS**

Expanded cell at step 1: queue(<s,f>, <s,h>)

Expanded cell at step 2: queue(<s,h>, <s,f,p>)

Expanded cell at step 2: queue(<s,f,p>, <s,h,k>)

Expanded cell at step 3: queue(<s,h,k>, <s,f,p,q>)

Expanded cell at step 3: queue(<s,f,p,q>, <s,h,k,c>)

Expanded cell at step 4: queue(<s,h,k,c>, <s,f,p,q,r>)

Expanded cell at step 4: queue(<s,f,p,q,r>, <s,h,k,c,a>)

Expanded cell at step 5: queue(<s,h,k,c,a>, <s,f,p,q,r,t>)

Expanded cell at step 5: queue(<s,f,p,q,r,t>, <s,h,k,c,a,b>)

Expanded cell at step 6: queue(<s,h,k,c,a>, <s,f,p,q,r,t>)

Expanded cell at step 6: queue(<s,f,p,q,r,t>, <s,h,k,c,a,b>)

Expanded cell at step 7: queue(<s,h,k,c,a,b>, <s,f,p,q,r,t,g>)

Expanded cell at step 7: queue(<s,f,p,q,r,t,g>, <s,h,k,c,a,b,d>)

Expanded cell at step 8: queue(<s,h,k,c,a,b,d>)

Goal reach: so the optimal path is s->f->p->q->r->t->g

**b. DFS with state checks along the way to avoid loops. The expansion order is Right -> Bottom -> Left -> Top**

Expanded cell at step 1: stack (<s,h>)

Expanded cell at step 2: stack (<s,h,k>)

Expanded cell at step 3: stack (<s,h,k,c>)

Expanded cell at step 4: stack (<s,h,k,c,a>)

Expanded cell at step 5: stack (<s,h,k,c,a,b>)

Expanded cell at step 6: stack (<s,h,k,c,a,b,d>)

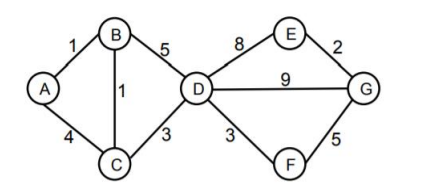
Expanded cell at step 7: stack (<s,h,k,c,a,b,d,e>, <s,h,k,c,a,b,d,m>)

Expanded cell at step 8: stack (<s,h,k,c,a,b,d,e>, <s,h,k,c,a,b,d,m,n>, <s,h,k,c,a,b,d,m,g>)

Expanded cell at step 9: stack (<s,h,k,c,a,b,d,e>, <s,h,k,c,a,b,d,m,n>)

Goal reach: so the optimal path is s->h->k->c->a->b->d->m->g

**Question 3: Find the way from A to G. Show the transformation steps of fringe (stack, queue).**

****

**Assuming the algorithm check the visited state of node to avoid loop and it will prioritize the path with lowest cost**

1. **BFS**

Fringe at Step 1: queue ([('B', ['A', 'B']), ('C', ['A', 'C'])])

(Expand node B and C from node A, prioritize the lowest cost path)

Fringe at Step 2: queue ([('C', ['A', 'C']), ('A', ['A', 'B', 'A']), ('C', ['A', 'B', 'C']), ('D', ['A', 'B', 'D'])])

(Expand node A, C, D from node B, prioritize the lowest cost path)

Fringe at Step 3: queue ([('A', ['A', 'B', 'A']), ('C', ['A', 'B', 'C']), ('D', ['A', 'B', 'D']), ('B', ['A', 'C', 'B']), ('D', ['A', 'C', 'D']), ('A', ['A', 'C', 'A'])])

(Expand node B, D, A from node C, prioritize the lowest cost path)

Fringe at Step 3: queue ([('C', ['A', 'B', 'C']), ('D', ['A', 'B', 'D']), ('B', ['A', 'C', 'B']), ('D', ['A', 'C', 'D']), ('A', ['A', 'C', 'A'])])

(Node A is visited)

Fringe at Step 3: queue ([('D', ['A', 'B', 'D']), ('B', ['A', 'C', 'B']), ('D', ['A', 'C', 'D']), ('A', ['A', 'C', 'A'])])

(Node C is visited)

Fringe at Step 4: queue ([('B', ['A', 'C', 'B']), ('D', ['A', 'C', 'D']), ('A', ['A', 'C', 'A']), ('C', ['A', 'B', 'D', 'C']), ('F', ['A', 'B', 'D', 'F']), ('B', ['A', 'B', 'D', 'B']), ('E', ['A', 'B', 'D', 'E']), ('G', ['A', 'B', 'D', 'G'])])

(Expand node C, F, B, E, G from node D, prioritize the lowest cost path)

Fringe at Step 4: queue ([('D', ['A', 'C', 'D']), ('A', ['A', 'C', 'A']), ('C', ['A', 'B', 'D', 'C']), ('F', ['A', 'B', 'D', 'F']), ('B', ['A', 'B', 'D', 'B']), ('E', ['A', 'B', 'D', 'E']), ('G', ['A', 'B', 'D', 'G'])])

(Node B is visited)

Fringe at Step 4: queue ([('A', ['A', 'C', 'A']), ('C', ['A', 'B', 'D', 'C']), ('F', ['A', 'B', 'D', 'F']), ('B', ['A', 'B', 'D', 'B']), ('E', ['A', 'B', 'D', 'E']), ('G', ['A', 'B', 'D', 'G'])])

(Node D is visited)

Fringe at Step 4: queue ([('C', ['A', 'B', 'D', 'C']), ('F', ['A', 'B', 'D', 'F']), ('B', ['A', 'B', 'D', 'B']), ('E', ['A', 'B', 'D', 'E']), ('G', ['A', 'B', 'D', 'G'])])

(Node A is visited)

Fringe at Step 4: queue ([('F', ['A', 'B', 'D', 'F']), ('B', ['A', 'B', 'D', 'B']), ('E', ['A', 'B', 'D', 'E']), ('G', ['A', 'B', 'D', 'G'])])

(Node C is visited)

Fringe at Step 5: queue ([('B', ['A', 'B', 'D', 'B']), ('E', ['A', 'B', 'D', 'E']), ('G', ['A', 'B', 'D', 'G']), ('D', ['A', 'B', 'D', 'F', 'D']), ('G', ['A', 'B', 'D', 'F', 'G'])])

(Expand node D, G from node F, prioritize the lowest cost path)

Fringe at Step 5: queue ([('E', ['A', 'B', 'D', 'E']), ('G', ['A', 'B', 'D', 'G']), ('D', ['A', 'B', 'D', 'F', 'D']), ('G', ['A', 'B', 'D', 'F', 'G'])])

(Node B is visited)

Fringe at Step 6: queue ([('G', ['A', 'B', 'D', 'G']), ('D', ['A', 'B', 'D', 'F', 'D']), ('G', ['A', 'B', 'D', 'F', 'G']), ('G', ['A', 'B', 'D', 'E', 'G']), ('D', ['A', 'B', 'D', 'E', 'D'])])

(Node E is visited)

Fringe at Step 6: queue ([('D', ['A', 'B', 'D', 'F', 'D']), ('G', ['A', 'B', 'D', 'F', 'G']), ('G', ['A', 'B', 'D', 'E', 'G']), ('D', ['A', 'B', 'D', 'E', 'D'])])

Goal reach: so the optimal path is: A->B->D->G

1. **DFS**

Stack at Step 1: stack [('C', ['A', 'C']), ('B', ['A', 'B'])]

(Expand node B and C from node A, prioritize the lowest cost path)

Stack at Step 2: stack [('C', ['A', 'C']), ('D', ['A', 'B', 'D']), ('A', ['A', 'B', 'A']), ('C', ['A', 'B', 'C'])]

(Expand node D, A and C from node B, prioritize the lowest cost path)

Stack at Step 3: stack [('C', ['A', 'C']), ('D', ['A', 'B', 'D']), ('A', ['A', 'B', 'A']), ('A', ['A', 'B', 'C', 'A']), ('D', ['A', 'B', 'C', 'D']), ('B', ['A', 'B', 'C', 'B'])]

(Expand node A, D and B from node C, prioritize the lowest cost path)

Stack at Step 3: stack [('C', ['A', 'C']), ('D', ['A', 'B', 'D']), ('A', ['A', 'B', 'A']), ('A', ['A', 'B', 'C', 'A']), ('D', ['A', 'B', 'C', 'D'])]

(Node B is visited)

Stack at Step 4: [('C', ['A', 'C']), ('D', ['A', 'B', 'D']), ('A', ['A', 'B', 'A']), ('A', ['A', 'B', 'C', 'A']), ('G', ['A', 'B', 'C', 'D', 'G']), ('E', ['A', 'B', 'C', 'D', 'E']), ('B', ['A', 'B', 'C', 'D', 'B']), ('C', ['A', 'B', 'C', 'D', 'C']), ('F', ['A', 'B', 'C', 'D', 'F'])]

(Expand node G, E, B, C and F from node D, prioritize the lowest cost path)

Stack at Step 5: [('C', ['A', 'C']), ('D', ['A', 'B', 'D']), ('A', ['A', 'B', 'A']), ('A', ['A', 'B', 'C', 'A']), ('G', ['A', 'B', 'C', 'D', 'G']), ('E', ['A', 'B', 'C', 'D', 'E']), ('B', ['A', 'B', 'C', 'D', 'B']), ('C', ['A', 'B', 'C', 'D', 'C']), ('G', ['A', 'B', 'C', 'D', 'F', 'G']), ('D', ['A', 'B', 'C', 'D', 'F', 'D'])]

(Expand node G, D from node F, prioritize the lowest cost path)

Stack at Step 5: [('C', ['A', 'C']), ('D', ['A', 'B', 'D']), ('A', ['A', 'B', 'A']), ('A', ['A', 'B', 'C', 'A']), ('G', ['A', 'B', 'C', 'D', 'G']), ('E', ['A', 'B', 'C', 'D', 'E']), ('B', ['A', 'B', 'C', 'D', 'B']), ('C', ['A', 'B', 'C', 'D', 'C']), ('G', ['A', 'B', 'C', 'D', 'F', 'G'])]

(Node D is visited)

Stack at Step 5: [('C', ['A', 'C']), ('D', ['A', 'B', 'D']), ('A', ['A', 'B', 'A']), ('A', ['A', 'B', 'C', 'A']), ('G', ['A', 'B', 'C', 'D', 'G']), ('E', ['A', 'B', 'C', 'D', 'E']), ('B', ['A', 'B', 'C', 'D', 'B']), ('C', ['A', 'B', 'C', 'D', 'C'])]

Goal reach: so the optimal path is: A->B->C->D->F->G

1. **UCS**

Fringe at Step 1: queue [(1, 'B', ['A', 'B']), (4, 'C', ['A', 'C'])]

(Expand node B, C from node A, prioritize the lowest cost path)

Fringe at Step 2: queue [(2, 'A', ['A', 'B', 'A']), (2, 'C', ['A', 'B', 'C']), (4, 'C', ['A', 'C']), (6, 'D', ['A', 'B', 'D'])]

(Expand node A, C, D from node B, prioritize the lowest cost path)

Fringe at Step 2: queue [(2, 'C', ['A', 'B', 'C']), (4, 'C', ['A', 'C']), (6, 'D', ['A', 'B', 'D'])]

(Node A is visited)

Fringe at Step 3: queue [(3, 'B', ['A', 'B', 'C', 'B']), (4, 'C', ['A', 'C']), (5, 'D', ['A', 'B', 'C', 'D']), (6, 'A', ['A', 'B', 'C', 'A']), (6, 'D', ['A', 'B', 'D'])]

(Expand node A, B, D from node C, prioritize the lowest cost path)

Fringe at Step 3: queue [(4, 'C', ['A', 'C']), (5, 'D', ['A', 'B', 'C', 'D']), (6, 'A', ['A', 'B', 'C', 'A']), (6, 'D', ['A', 'B', 'D'])]

(Node B is visited)

Fringe at Step 3: queue [(5, 'D', ['A', 'B', 'C', 'D']), (6, 'D', ['A', 'B', 'D']), (6, 'A', ['A', 'B', 'C', 'A'])]

(Node C is visited)

Fringe at Step 4: queue [(6, 'A', ['A', 'B', 'C', 'A']), (6, 'D', ['A', 'B', 'D']), (8, 'F', ['A', 'B', 'C', 'D', 'F']), (8, 'C', ['A', 'B', 'C', 'D', 'C']), (13, 'E', ['A', 'B', 'C', 'D', 'E']), (10, 'B', ['A', 'B', 'C', 'D', 'B']), (14, 'G', ['A', 'B', 'C', 'D', 'G'])]

(Expand node B, C, E, F and G from node D, prioritize the lowest cost path)

Fringe at Step 4: queue [(6, 'D', ['A', 'B', 'D']), (8, 'C', ['A', 'B', 'C', 'D', 'C']), (8, 'F', ['A', 'B', 'C', 'D', 'F']), (14, 'G', ['A', 'B', 'C', 'D', 'G']), (13, 'E', ['A', 'B', 'C', 'D', 'E']), (10, 'B', ['A', 'B', 'C', 'D', 'B'])]

(Node A is visited)

Fringe at Step 4: queue [(8, 'C', ['A', 'B', 'C', 'D', 'C']), (10, 'B', ['A', 'B', 'C', 'D', 'B']), (8, 'F', ['A', 'B', 'C', 'D', 'F']), (14, 'G', ['A', 'B', 'C', 'D', 'G']), (13, 'E', ['A', 'B', 'C', 'D', 'E'])]

(Node D is visited)

Fringe at Step 4: queue [(8, 'F', ['A', 'B', 'C', 'D', 'F']), (10, 'B', ['A', 'B', 'C', 'D', 'B']), (13, 'E', ['A', 'B', 'C', 'D', 'E']), (14, 'G', ['A', 'B', 'C', 'D', 'G'])]

(Node C is visited)

Fringe at Step 5: queue [(10, 'B', ['A', 'B', 'C', 'D', 'B']), (11, 'D', ['A', 'B', 'C', 'D', 'F', 'D']), (13, 'E', ['A', 'B', 'C', 'D', 'E']), (14, 'G', ['A', 'B', 'C', 'D', 'G']), (13, 'G', ['A', 'B', 'C', 'D', 'F', 'G'])]

(Expand node G from node F, prioritize the lowest cost path)

Fringe at Step 5: queue [(11, 'D', ['A', 'B', 'C', 'D', 'F', 'D']), (13, 'G', ['A', 'B', 'C', 'D', 'F', 'G']), (13, 'E', ['A', 'B', 'C', 'D', 'E']), (14, 'G', ['A', 'B', 'C', 'D', 'G'])]

(Node B is visited)

Fringe at Step 5: queue [(13, 'E', ['A', 'B', 'C', 'D', 'E']), (13, 'G', ['A', 'B', 'C', 'D', 'F', 'G']), (14, 'G', ['A', 'B', 'C', 'D', 'G'])]

(Node D is visited)

Fringe at Step 6: queue [(13, 'G', ['A', 'B', 'C', 'D', 'F', 'G']), (14, 'G', ['A', 'B', 'C', 'D', 'G']), (21, 'D', ['A', 'B', 'C', 'D', 'E', 'D']), (15, 'G', ['A', 'B', 'C', 'D', 'E', 'G'])]

(Expand node G from node E, prioritize the lowest cost path)

Fringe at Step 6: queue [(14, 'G', ['A', 'B', 'C', 'D', 'G']), (21, 'D', ['A', 'B', 'C', 'D', 'E', 'D']), (15, 'G', ['A', 'B', 'C', 'D', 'E', 'G'])]

Goal reach: so the optimal path is: A->B->C->D->F->G