Decision Trees

• Looked at knapsack problem
• Each decision
  • left path (include item), right path (don’t include)
ROOTED BINARY TREE

- Top node
  - No parents
    - Root
- Non-root nodes
  - One parent
- Childless node
  - Leaf
- All nodes
  - At most two children
Building Tree Depth-First

Some “take” nodes missing, because of weight limit
TREE SIZE

• Size of tree
  • `len(items)` depth
  • At most $2^\text{level}$ nodes at each level
• Using a decision tree to solve 0/1 knapsack problem is $O(?)$
SHORTEST PATH

• Source node A
• Destination node B
• Shortest sequence of edges $A \rightarrow B$ such that
  • Edges follow in sequence
  • Destination node of one edge is the source node of another edge
Finding Shortest Path with DFS

Start by choosing one child of current node
Keep doing that until
  Reached node already seen (Why?)
  Reached goal
  Reached a node with no unvisited children
If reached node with no children, backtrack and take next child of node
  → Systematically explore all possible paths
  → Along the way keep track of best one
DFS EXAMPLE – SHORTEST PATH 0 → 5

Path traversed so far:
0
Length: 0
DFS EXAMPLE – SHORTEST PATH 0→5

Path traversed so far: 0→1
Length: 1
Path traversed so far: 0→1→2
Length: 2
DFS EXAMPLE – SHORTEST PATH 0→5

Path traversed so far: 0→1→2→3
Length: 3
DFS EXAMPLE – SHORTEST PATH 0→5

Path traversed so far:

0→1→2→3→4

Length: 4
DFS EXAMPLE – SHORTEST PATH 0 → 5

Path traversed so far: 0 → 1 → 2 → 3 → 5
Length: 4
DFS EXAMPLE – SHORTEST PATH 0→5

Path traversed so far:
0→1→2→4

Length: 3
DFS EXAMPLE – SHORTEST PATH 0→5

Path traversed so far: 0→2
Length: 1
DFS EXAMPLE – SHORTEST PATH 0→5

Path traversed so far: 0→2→3
Length: 2
DFS EXAMPLE – SHORTEST PATH $0\to5$

Path traversed so far: $0\rightarrow2\rightarrow3\rightarrow1$

Length: 3
DFS EXAMPLE – SHORTEST PATH 0 → 5

Path traversed so far:
0 → 2 → 3 → 4

Length: 3
DFS EXAMPLE – SHORTEST PATH 0→5

Path traversed so far: 0→2→3→5
Length: 3
DFS EXAMPLE – SHORTEST PATH 0→5

Path traversed so far: 0→2→4
Length: 2
DFS EXAMPLE – SHORTEST PATH 0→5

• Backtrack to 2
  • Visited all children
• Backtrack to 0
  • Visited all nodes
• Done
DFS SHORTEST PATH 0 → 5

- 0
- 0 → 1
- 0 → 1 → 2
- 0 → 1 → 2 → 3
- 0 → 1 → 2 → 3 → 4
- 0 → 1 → 2 → 3 → 5 ← length 4
- 0 → 1 → 2 → 4
- 0 → 2
- 0 → 2 → 3
- 0 → 2 → 3 → 1
- 0 → 2 → 3 → 4
- 0 → 2 → 3 → 5 ← length 3
- 0 → 2 → 4
THOUGHTS ON DFS

• To get global shortest path
  • Search entire tree for all possible paths
  • May not find global SP if stop early
  • What is complexity?

• Weighted edges
  • Will not necessarily find shortest path
  • Can modify the algorithm
BREADTH FIRST SEARCH

• Start at source node, picks every child before moving down one level

Iterate through every child of start node
Keeps doing that until
  Reached node need to get at
  If destination node not found, start nodes become each children of start node
def BFS(graph, start, end, pathQueue = []):
    # assumes graph is a Digraph
    # assumes start and end are nodes in graph
    initPath = [start]
    pathQueue.append(initPath)
    visited = [start] # to avoid going to nodes have already seen
    while len(pathQueue) != 0:
        # Get and remove oldest element in pathQueue
        tmpPath = pathQueue.pop(0)
        lastNode = tmpPath[len(tmpPath) - 1]
        print 'Current BFS path:', printPath(tmpPath)
        if lastNode == end:
            return tmpPath
        for linkNode in graph.childrenOf(lastNode):
            if linkNode not in visited:
                newPath = tmpPath + [linkNode]
                pathQueue.append(newPath)
                # uncomment this line to add node to nodes have seen
                # visited.append(linkNode)
    return None
BFS EXAMPLE – SHORTEST PATH 0→5

• Enumerate children of start node
  • 1 and 2

Path traversed so far:
  0
Length: 0
BFS EXAMPLE – SHORTEST PATH 0→5

- Go through each child
  - Is 1 the end node?
    - No
    - Keep going to other children

Path traversed so far:
0→1

Length: 1
BFS EXAMPLE – SHORTEST PATH 0 → 5

• Go through each child
  • Is 2 the end node?
    • No
• Node 0 has no other children
  • Go through children of node

Path traversed so far: 0 → 2
Length: 1
BFS EXAMPLE – SHORTEST PATH 0→5

• Node 1 has only one child
• Is 2 the end node?
  • No
• Look at other nodes on same level as 1

Path traversed so far: 0→1→2
Length: 2
BFS EXAMPLE – SHORTEST PATH 0→5

• Node 2 has 2 children
  • 3 and 4
• Is 3 the end node?
  • No

Path traversed so far:
  0→2→3
Length: 2
BFS EXAMPLE – SHORTEST PATH 0→5

• Node 2 has 2 children
  • 3 and 4
• Is 4 the end node?
  • No
• Look at nodes on next level

Path traversed so far: 0→2→4
Length: 2
BFS EXAMPLE – SHORTEST PATH 0→5

• Through node 1, node 2 has 2 children
  • 3 and 4
• Is 3 the end node?
  • No

Path traversed so far:
  0→1→2→3
Length: 3
BFS EXAMPLE – SHORTEST PATH 0→5

• Through node 1, node 2 has 2 children
  • 3 and 4
• Is 4 the end node?
  • No
• Keep going with this level

Path traversed so far:
  0→1→2→4
Length: 3
**BFS EXAMPLE – SHORTEST PATH 0→5**

- Node 3 has 3 children
  - 1, 4, 5
- Is 1 the end node?
  - No

Path traversed so far: 0→2→3→1

Length: 3
BFS EXAMPLE – SHORTEST PATH 0→5

• Node 3 has 3 children
  • 1, 4, 5
• Is 4 the end node?
  • No

Path traversed so far: 0→2→3→4
Length: 3
BFS EXAMPLE – SHORTEST PATH 0→5

• Node 3 has 3 children
  • 1, 4, 5
• Is 5 the end node?
  • Yes, stop here
  • Going farther will no longer be shortest path

Path traversed so far:
  0→2→3→5
Length: 3
BFS SHORTEST PATH 0→5

- 0
- 0→1
- 0→2
- 0→1→2
- 0→2→3
- 0→2→4
- 0→1→2→3
- 0→1→2→4
- 0→2→3→1
- 0→2→3→4
- 0→2→3→5

• Running algo with adding nodes already seen to a visited list.
- If see them again later, then skip these paths (know they will be longer than what have already seen)

← length 3
THOUGHTS ON BFS

• Finds optimal solution
  • Does not need to enumerate all paths
  • Every node reached through shortest path
• Caveat !!
  • All edge weights must be the same
  • Only true for unweighted graphs

• Weighted graphs?
  • Can’t assume node at next level is nearest
  • Keep distance estimate for each node
SUMMARY

• Graphs
  • Nodes and edges
  • Unidirectional, bidirectional, weighted
  • Adjacency matrix vs. adjacency list representation

• DFS
  • Recursive
  • Search all possible paths to find shortest

• BFS
  • Finds shortest path without searching all paths, guaranteed only on unweighted graphs