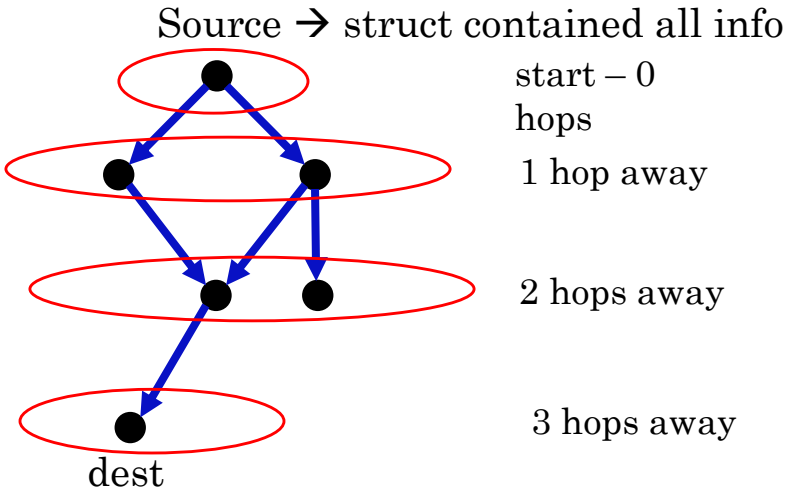
Breadth First Search

How it works

- Inspects all neighbouring nodes first
  - Then for each of those, inspects their nodes which were unvisited

- This uses a queue (First In First Out)

Breadth First Search (BFS)



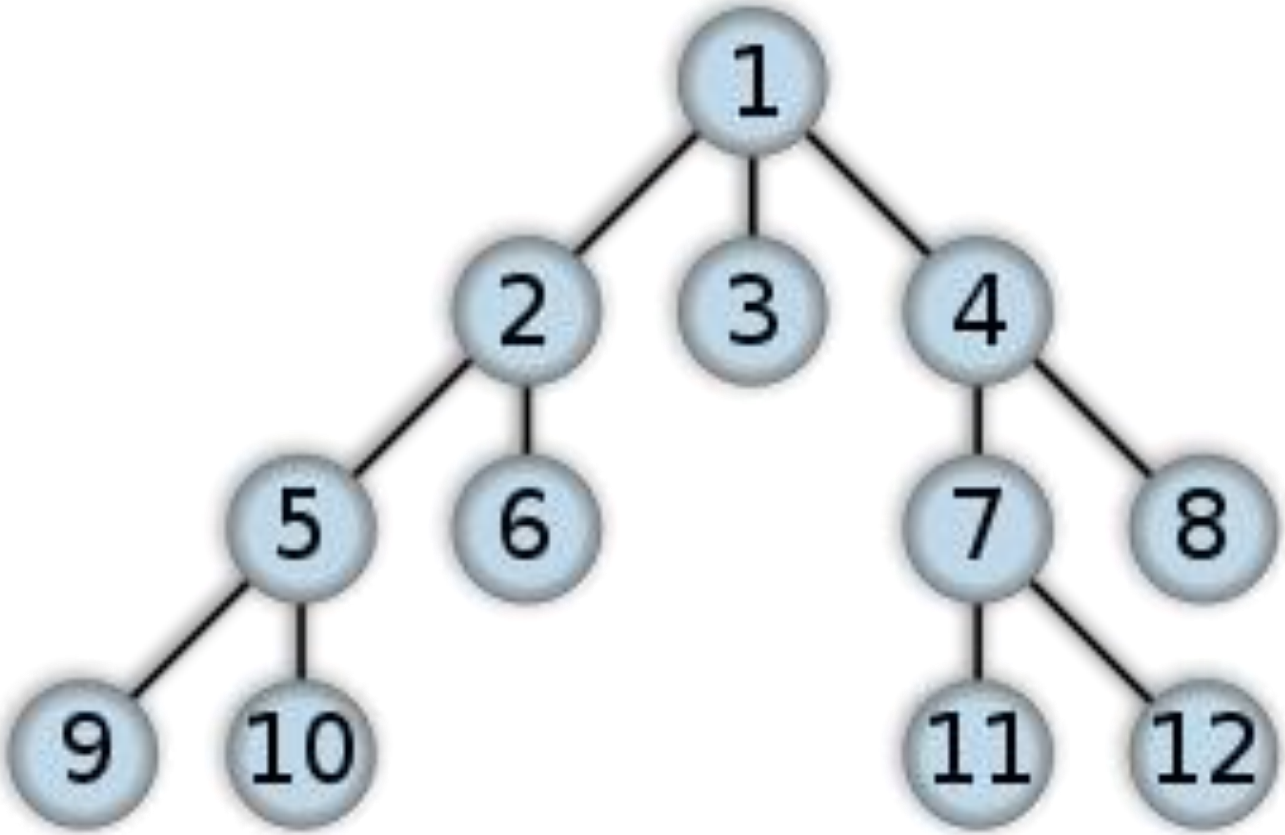

```
int main () {
    ...
    bool found = bfsPath (sourceNode, destID);
    . . .
}

bool bfsPath (Node sourceNode, int destID) {
    ...
}
```

Queue

- FIFO (First In first Out)
- Queues are objects that work just like normal queues
- #include <queue>
- queue<data type> name;
- Functions
  - name.push(data type a), pushes something into the queue
  - name.pop(), pops the top element
    - NOTE: DOES NOT RETURN
  - name.front(), returns front element
  - name.back(), returns back element

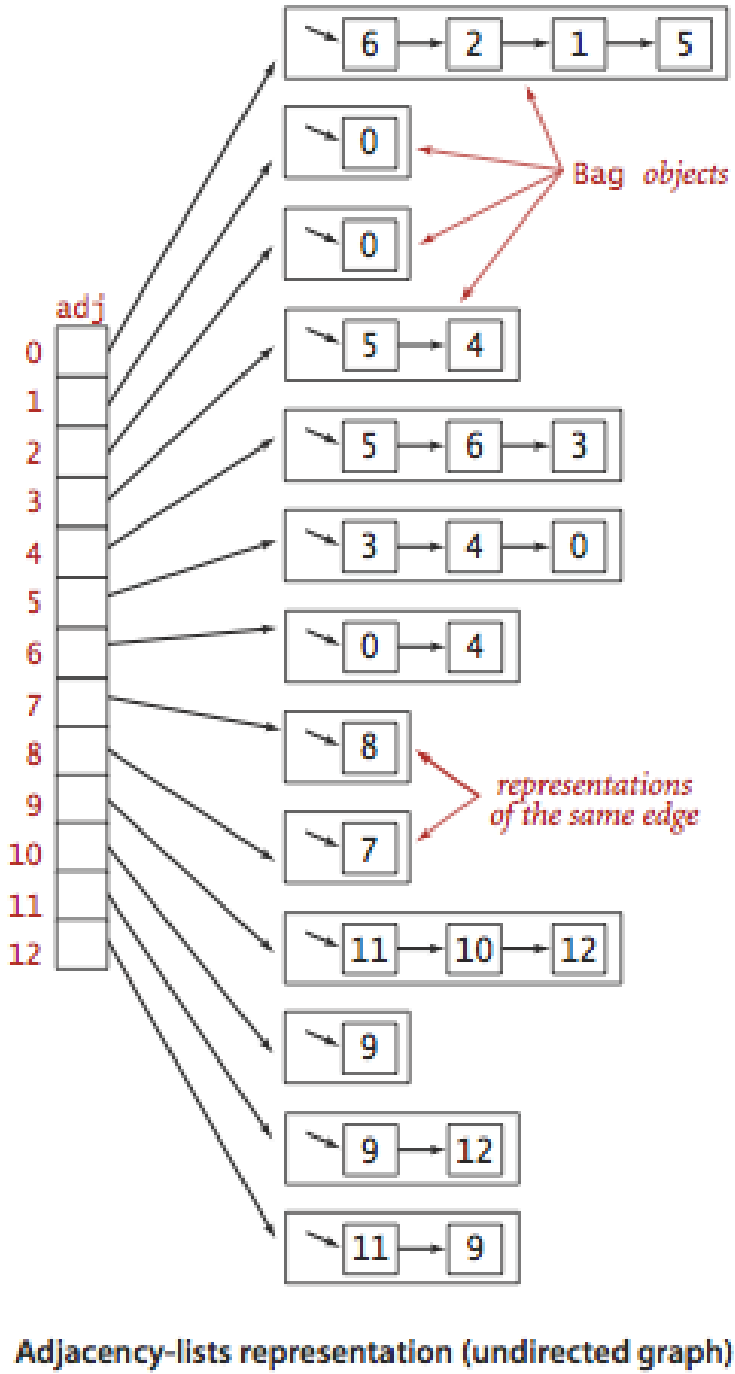
Data Structure



Data Structure

Each element:

```
struct Node{
    int id;//currentnode id
    vector<int> adjID;
    //maybe other info
    //maybe constructors
}
```



Adjacency-lists representation (undirected graph)

Breadth First Search

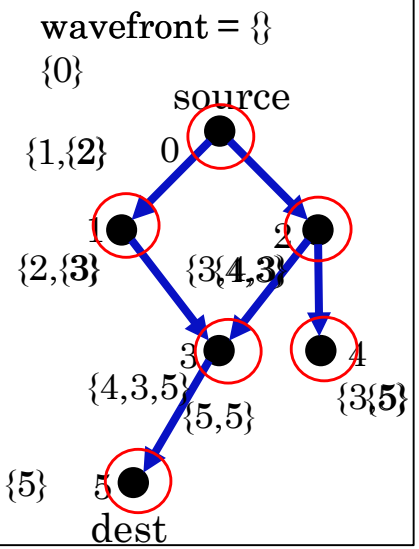
```
bool bfsPath (Node sourceNode, int destID) {
   queue<Node> wavefront;//these are the nodes need to check their
                          children nodes
   wavefront.push_back(sourceNode); //push the very first node

   while (wavefront not empty) { //if there is still node to check
      Node currNode = wavefront.front (); //get the node
      wavefront.pop_front();  // Remove node from wavefront

      if (currNode.id == destID)//if it's dest
         return (true);

      for each (children node of currentNode) {
         wavefront.push_back (childrenNode);
      }
   }

   return (false);   // No path exits!
}
```



BFS: How Do I Print Out the Path?

Need more information!

```
struct waveElem {
   Node node;
   int lastElementID; // last nodeID used to reach this node
    //need a constructor
   waveElem(Node _node, int _lastElementID){
        node = _node;
        lastElementID = _lastElementID;
   }
};

//no lastelementID = -1
```

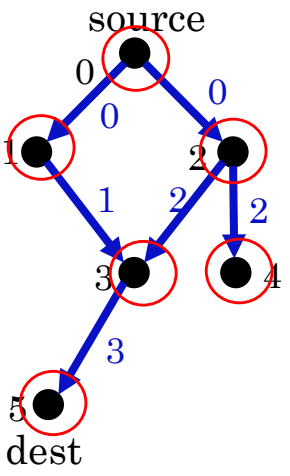
BFS: How Do I Print Out the Path?

```
bool bfsPath (Node sourceNode, int destID) {
   queue<waveElem> wavefront;
   wavefront.push_back (waveElem (sourceNode, -1)); //no reaching ID

   while (wavefront not empty) {
      waveElem curr = wavefront.front (); //get the next node to check
      wavefront.pop_front();  // Remove node from wavefront

      if (curr.node.id == destID) //check if it's the dest
         return (true);

      for each (childrenNode of currNode) {
         wavefront.push_back (
              waveElem(childrenNode, curr.node.id);
         //change the actual node's reaching ID
         childrenNode.reachingID = curr.node.id;
      }
   }
   return (false);   // No path exits!
}
```



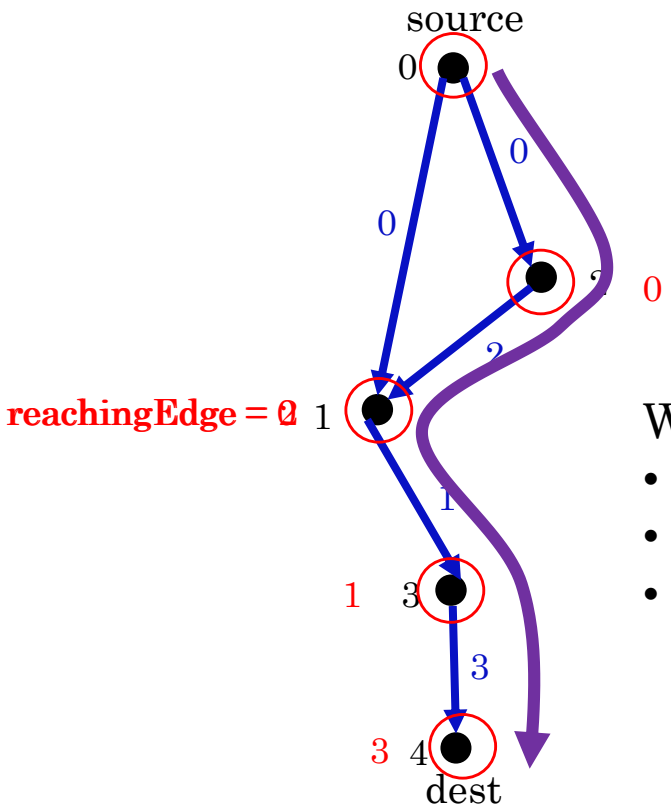
BFS: How Do I Print Out the Path?

```
int main () {
   Node sourceNode = … ;
   bool found = bfsPath (sourceNode, destID);
   if (found)
      list<int id> path = bfsTraceBack (destID);
}

list<int id> bfsTraceBack (int destID) {
   list<int id> path;
   Node destNode = getNodebyID(destID);
   Node currNode = destNode;

   while(currNode.reachingID != -1){
      path.push_back(currNode.reachingID);
      currNode = getNodebyID(currNode.reachingID);
   }
   return (path);
}
```

Min. Path Issues

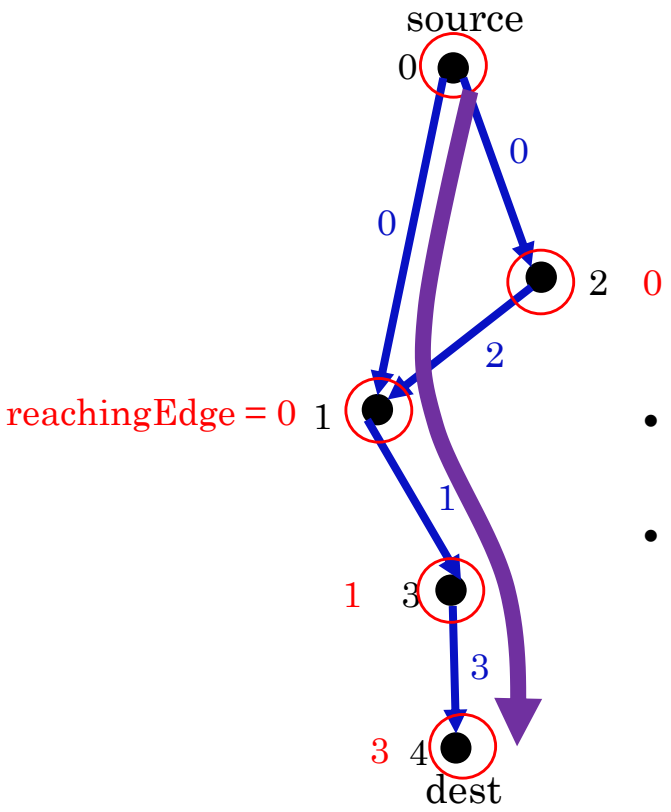


wavefront = {1, 3}

What happened?

- Node 1 was re-expanded
- Overwrote the reachingEdge
- Messed up the path traceback

Solution?



wavefront = {1, 2, 3}

- Mark nodes as visited when you reach them
- Never allow them to be visited again