Breadth-first search, the problem

We want to find the minimum length of a path from node $u$ to node $v$. We will actually find such length to all nodes, this is single source shortest paths problem. We measure the length of a path in the number of edges.

Later we will look at more difficult versions, when each edge has a cost and we measure the length of a path as the sum of these costs.

If a path $(u = v_0, v_1, \ldots, v_k = v)$ is a shortest path to $v$, then either $k = 0$ and $v = u$, the basic case, or this is a one edge extension of another shortest path, from $u$ to $v_{k-1}$. This suggest a way to compute:

Compute the basic case first.

After paths of length $k - 1$ are computed, find their one edge extensions and accept if they provide paths to new targets. A new target does not have a shorter path because it would be computed earlier.

To do it methodically, we maintain a queue of targets that we have reached.
queue Q;
makeempty(Q);
Dist[u] = 0;
enqueue(u,Q);

for (every node v)
   Dist[v] = n;  // n is like infinity
while (!empty(Q)) {
   v = dequeue(Q);
   for (every node w adjacent to v)
      if (Dist[w] == n)
         Dist[w] = Dist[v]+1,
         Predecessor[w] = v;
      enqueue(w,Q);
}

Implementing queue is very simple in this case because each node is put on the queue but once.

// making empty queue
int *Q = (int*) malloc(n*sizeof(n));
Qfront = Qtail = 0;

// test for empty queue
Qfront == Qtrail

// enqueue w
Q[Qtail++] = w;

// dequeue
v = Q[Qfront++];
Search for fake coin.

Now we look at problems when we can reduce size of an input instance to an instance smaller by a constant factor.

We have $n$ coins, of equal weight, except for a fake coin that is lighter.
We also have a scale.
Find the fake coin.

Basic case: $n = 1$. Our coin is fake.

Put $\kappa$ coins on one scale, $\kappa$ on the other. The heavier pile does not contain the fake, so we remove it from further consideration.

Use largest $\kappa$ possible, i.e. $\lfloor n/2 \rfloor$. We get the recurrence for the running time the same as for binary search.
**Josephus problem (Jewish rebellion against the Rome)**

We have $n$ rebels who have a suicide pact. They will stand in the circle, with positions 0 to $n - 1$, and the leader will start the grim process. They use a single sword.

Whoever has the sword, stabs the next person, who falls out of the circle and is no longer next. Then the sword is passed to the next person. At the end, there is no one to pass the sword to: the last person is standing.

Who is the last person? He is on position $J(n)$. Can we compute quickly? By taking this position in the circle we can either survive, or commit suicide by a more painful method of falling on our own sword.

Even case, $n = 2m$: the sword returns to the leader, now a rebel who started at position $2k$ is on position $k$. Thus survivor on position $J(m)$ started at position $J(2m) = 2J(m)$.

Odd case, $n = 2m + 1$: positions from 0 to $2m$, and $2m$ kills the leader, and the sword passes to rebel who was at position 2. So now 2 is a momentary leader with $m$ rebels remaining, and a rebel at position $k$ was at position $2(k + 1)$. Thus survivor at position $J(m)$ started at $J(2m + 1) = 2J(m) + 2$. 

Next time:

Median and order statistics, and start of Chapter 6.