5 *Prove that if the Breadth-First-Search algorithm starting at v visits vertex w_1 before vertex w_2, then the distance from v to w_1 is less than or equal to the distance from v to w_2.*

```
BreadthFirstSearch(Graph G, Vertex v)
begin
   Q = empty queue;

   for each u in V
      encountered(u) = false;

   encountered(v) = true;
   Q.enqueue(v);

   while Q not empty
      w = Q.dequeue();
      Visit(w)
      for each neighbor w' of w
         if w' is not encountered
            encountered(w') = true;
            Q.enqueue(w');
         end
      end
   end
end
```

Lemma 1 *In the graph $G = (V, E)$, before the start of each iteration of the while loop of the breadth first search from v:*

Let d be the distance from v to the first vertex in the queue. Then, ly

Condition A *The vertices in the queue are either at distance d or $d+1$ from v, and the vertices at distance d precede those at distance $d+1$ in the queue.*

Condition B *All vertices at distance less than or equal to d from v have already been encountered and hence enqueued (and possibly dequeued).*

Proof: induction on i (the number of iterations of the while loop in the algorithm)

base case ($i = 0$)**:** Before the first iteration of the while loop, v is the only vertex in the queue. It is at distance 0. Thus, the inductive hypothesis is trivially satisfied.

inductive hypothesis: Assume that after $k-1$ iterations of the while loop complete, the inductive hypothesis holds, and the first vertex in the queue, say w, is at distance d from v. Thus, by hypothesis, all vertices in the queue are at distance d or $d+1$, all vertices at distance d in the queue precede all vertices at distance $d+1$ in the queue, and all vertices at distance d are either in the queue or have already been dequeued.

inductive step: We must show that the inductive hypothesis holds after k iterations of the while loop.

In the kth iteration we execute the code:

```
w = Q.dequeue();
Visit(w);
for each neighbor w' of w
   if w' is not encountered
      encountered(w') = true;
      Q.enqueue(w');
   end
end
```

Proof that Condition A holds after step k:

By the inductive hypothesis, all vertices still in the queue are distance greater than or equal to d and all vertices of distance d or less have already been enqueued. Thus, all vertices adjacent to w that have not been encountered are at distance strictly greater than d. Since these vertices are adjacent to w and w has distance d, these vertices must be at distance $d+1$. Therefore, any vertex that is added in this iteration is at distance $d+1$, and it is added at the end of the queue, and so we maintain condition A of the inductive hypothesis. (Note that the queue might not have any vertices at distance d anymore, but this is fine.)

Proof that Condition B holds after step k:

If after step k, the first vertex in the queue is still at distance d, then Condition B continues to hold trivially. Therefore, we only need consider the case where after dequeueing w, only vertices at distance $d+1$ remain in the queue. But by Condition B of the hypothesis, all vertices at distance d had already been encountered before step k. If none of these vertices remain in the queue after step k, then they have all already been dequeued and visited. Since every vertex at distance $d+1$ is adjacent to some vertex at distance d, and all the vertices at distance d have been processed, every vertex at distance $d+1$ must have already been encountered and enqueued. $\square$

The theorem now follows easily from the lemma as follows:

Theorem 2 *If the Breadth-First-Search algorithm starting at v visits vertex w_1 before vertex w_2, then the distance from v to w_1 is less than or equal to the distance from v to w_2.*

Proof:

If the algorithm visits w_1 before w_2, then w_1 is encountered, enqueued and hence dequeued before w_2. Suppose that the distance from v to w_1 is d. We need to show that the distance from v to w_2 is greater than or equal to d, which we do by considering two cases.

Case 1: At the time w_1 is dequeued, w_2 is already in the queue: Then by the Lemma, w_2's distance is either d or $d+1$.

Case 2: At the time w_1 is dequeued, w_2 has not yet been enqueued: Then w_2 has not yet been encountered, and hence by Condition B of the Lemma the distance from v to w_2 is greater than or equal to $d+1$.

In either case, we are done. $\square$