743. Network Delay Time

There are `N` network nodes, labelled `1` to `N`.

Given `times`, a list of travel times as **directed** edges `times[i] = (u, v, w)`, where `u` is the source node, `v` is the target node, and `w` is the time it takes for a signal to travel from source to target.

Now, we send a signal from a certain node `K`. How long will it take for all nodes to receive the signal? If it is impossible, return `-1`.

Note:

1. `N` will be in the range `[1, 100]`.
2. `K` will be in the range `[1, N]`.
3. The length of `times` will be in the range `[1, 6000]`.
4. All edges `times[i] = (u, v, w)` will have `1 <= u, v <= N` and `1 <= w <= 100`.

Difficulty:Medium

- Total Accepted:1.2K
- Total Submissions:4.2K
- Contributor: zestypanda
- Subscribe to see which companies asked this question.

Related Topics Digistra BFS

```
#include<sstream>
#include<vector>
#include<algorithm>
#include<unordered_map>
#include<limits.h>
#include<queue>
#include<unordered_map>
#include<unordered_set>
#include<stack>
using namespace std;
typedef struct comp
{
     bool operator()(pair<int,int> a, pair<int,int> b)
     {
          return a.first<b.first;
     }
}comp;

int networkDelayTime(vector<vector<int>>& times, int N, int K)
{
     priority_queue<pair<int,int>,vector<pair<int,int>>,comp> pq;
     vector<vector<pair<int,int>>>
adj(N+1,vector<pair<int,int>>());
     vector<int> waits(N + 1, INT_MAX);
     for(auto e:times)
     {
          adj[e[0]].push_back({e[1],e[2]});
     }
     queue<int> q;
     q.push(K);
     waits[K]=0;
     while(!q.empty())
     {
          unordered_set<int> set;
          for(int n=q.size();n>0;n--)
          {
               int u = q.front(); q.pop();
               for(auto elem:adj[u])
               {
                    int v = elem.first;
                    int w = elem.second;
                    if(waits[v]>waits[u]+w)
                    {
                         if(!set.count(v))
                         {
                              set.insert(v);
                              q.push(v);
                         }
                         waits[v] = waits[u]+w;
                    }
               }
          }
     }
     int maxwait = 0;
    for (int i = 1; i <= N; i++)
        maxwait = max(maxwait, waits[i]);
    return maxwait == INT_MAX ? -1 : maxwait;
```

```
}
```