Shortest Path Algorithms

TAI, WEI HSUAN

July 30, 2025

Outline

- Dijkstra's Algorithm
- Bellman-Ford Algorithm
- Floyd-Warshall Algorithm
- More about it

Introduction

To design a network, we need to consider the shortest path between nodes so that we can optimize the flow of data or resources.
The shortest path algorithms can help us find the most efficient route in a network (or in a navigation system).

Dijkstra's Algorithm

Dijkstra's algorithm is a greedy algorithm that finds the shortest path from a source node to all other nodes in a weighted graph.
The following steps outline the algorithm:

1. Initialize the distance to the source node as 0 and all other nodes as infinity.
2. Add the nearest unvisited node to the set of visited nodes.
3. Update the distances to the neighboring nodes of the current node.
4. Repeat until the destination node is reached.

Dijkstra's algorithm is efficient for graphs with non-negative weights and can be implemented using a priority queue.

Example for Dijkstra's Algorithm

6. Consider the network shown in Figure 5. With the indicated link costs, use Dijkstra' s shortest-path algorithm to compute the shortest path from x to all network nodes. Show how the algorithm works by computing a table similar to Table I. (10%)

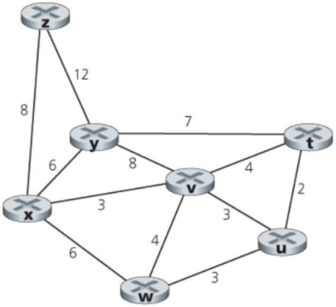


Figure 5

Code for Dijkstra's Algorithm

```
struct edge{
    int v, w;
};
vector<edge> e[N];
int dis[N], vis[N];
void dijkstra(int n, int s){
    fill(dis, dis+N, inf);
    dis[s]=0;
    for(int i=1;i<=n;++i){
        int u=0, min_dis=inf;
        for(int j=1;j<=n;++j)
            if(!vis[j] and dis[j]<min_dis)
                u=j, min_dis=dis[j];
        vis[u]=1;
        for(auto ed:e[u]){
            int v=ed.v, w=ed.w;
            if(dis[v]>dis[u]+w)dis[v]=dis[u]+w;
        }
    }
}
```

What is the time complexity of the code above?

What is the time complexity of the code above?

Answer

We use double loop to find the minimum distance node, so the time complexity is $O(n^2)$.

If we use a priority queue to store the nodes, the time complexity can be reduced to $O((n+m)\log n)$,

Practice

- CSES 1671 - Pure Dijkstra

Summary

Dijkstra's algorithm ...

- is an $O((n+m)\log n)$, greedy algorithm.
- can help us find the minimum distance from **a source node** to all other nodes.
- can't handle negative weights.

Bellman-Ford Algorithm

If there are negative weights in the graph, the Dijkstra's algorithm may not work correctly. In this case, we can use the Bellman-Ford algorithm, a dynamic programming algorithm based on the principle of relaxation.
The following steps outline the algorithm:

1. Initialize the distance to the source node as 0 and all other nodes as infinity.
2. **Relax** all edges repeatedly for $n-1$ iterations, where n is the number of nodes.
3. Check for negative weight cycles by relaxing all edges one more time.

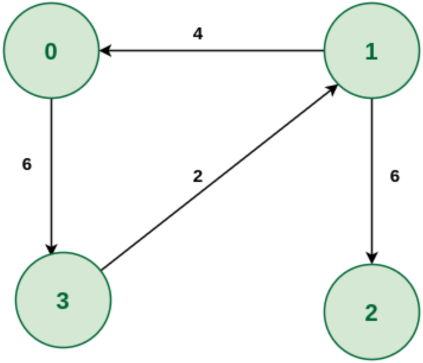
TL;DR:The Bellman-Ford algorithm can handle graphs with negative weights and is useful for detecting negative weight cycles.

Relaxation

Define `dis(u, v)` as the current shortest distance from node `u` to node `v`, and `w(u, v)` as the weight of the edge from `u` to `v`.
The relaxation step can be defined as:

```
dis(u, v) = min(dis(u, v), dis(u)+w(u, v))
```

Negative Cycle



Code for Bellman-Ford Algorithm-Initialize

```
struct edge{
    int u, v, w;
};
vector<edge> e;
int dis[N];
```

Code for Bellman-Ford Algorithm-Function

```
bool bellman_ford(int n, int s){
    fill(dis, dis+N, inf);
    dis[s]=0;
    bool relax=0;
    for(int i=1;i<=n;++i){
        relax=0;
        for(auto ed:e){
            int u=ed.u, v=ed.v, w=ed.w;
            if(dis[u]==inf)continue;
            if(dis[v]>dis[u]+w){
                dis[v]=dis[u]+w;
                relax=1;
            }
        }
        if(!relax)break;
    }
    return relax;
}
```

What is the time complexity of the code above?

What is the time complexity of the code above?

Answer

We use double loop to find the minimum distance node, so the time complexity is $O(nm)$, where n is the number of nodes and m is the number of edges.

Practice

- CSES 1672 - Pure Bellman-Ford
- CSES 1673 - Change the relax function

Summary

Bellman-Ford algorithm ...

- is an $O(nm)$, dynamic programming algorithm.
- can help us find the minimum distance from **a source node** to all other nodes.
- can handle negative weights.

If the graph doesn't contain any negative weights, Dijkstra's algorithm is a better choice.

Floyd-Warshall Algorithm

To find the shortest path between all pairs of nodes in a graph, we can use the Floyd-Warshall algorithm, which is a dynamic programming algorithm.
The following steps outline the algorithm:

1. Initialize a distance matrix with the weights of the edges.
2. Iterate through all pairs of nodes and update the distance matrix by considering each node as an intermediate node.
3. If the distance from node `u` to node `v` through node `k` is shorter than the current distance, update the distance.
4. Repeat until all pairs of nodes have been considered.

Code for Floyd-Warshall Algorithm

```
int mp[1005][1005][1005];
int n,m;

inline void warshall(){
    for(int k=0;k<n;++k)
    for(int i=0;i<n;++i)
    for(int j=0;j<n;++j)
        mp[k][i][j]=min(mp[k-1][i][j],mp[k-1][i][k]+mp[k-1][k][j]);
}
```

Code for Floyd-Warshall algorithm

We found that k can be neglected in the code above.

```
int mp[1005][1005];
int n,m;

inline void warshall(){
    for(int k=0;k<n;++k)
    for(int i=0;i<n;++i)
    for(int j=0;j<n;++j)
        mp[i][j]=min(mp[i][j],mp[i][k]+mp[k][j]);
}
```

What is the time complexity of the code above?

What is the time complexity of the code above?

Answer

We use triple loop to find the minimum distance node, so the time complexity is $O(n^3)$, where n is the number of nodes.

Summary

- Floyd-Warshall algorithm is simple but slow, it also occupy lots of memory.
- We can use Dijkstra or Bellman-Ford algorithm to replace it in most cases.
- But if the graph is hugely dense, the Floyd-Warshall algorithm can be more efficient than Dijkstra or Bellman-Ford because it can find the shortest path between all pairs of nodes in one go.
- Think about it, how to detect negative cycles in the graph with Floyd-Warshall algorithm?

Practice

- CSES 1197
- (optional)Luogu P3403 - implicit graph
- (optional)CF 1540A - Construct a Graph

More about it

- kth shortest path problem
- Johnson's algorithm
- A* algorithm
- implicit graph(i.e. measuring jugs problem)