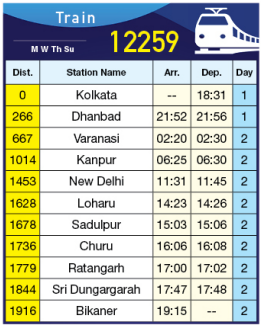
Pseudocode: Graphs — Trains

Trains

- Train dataset — information about trains and stations
  - Each train is a route list of stations
  - Each station is a route list of trains passing through

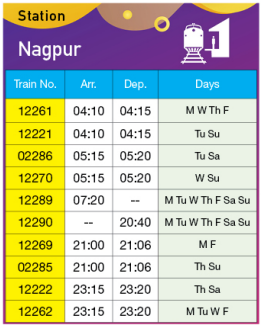


Train

M W Th Su

12259

Dist.	Station Name	Arr.	Dep.	Day
0	Kolkata	--	18:31	1
266	Dhanbad	21:52	21:56	1
667	Varanasi	02:20	02:30	2
1014	Kanpur	06:25	06:30	2
1453	New Delhi	11:31	11:45	2
1628	Loharu	14:23	14:26	2
1678	Sadulpur	15:03	15:06	2
1736	Churu	16:06	16:08	2
1779	Ratangarh	17:00	17:02	2
1844	Sri Dungargarah	17:47	17:48	2
1916	Bikaner	19:15	--	2



Station

Nagpur

Train No.	Arr.	Dep.	Days
12261	04:10	04:15	M W Th F
12221	04:10	04:15	Tu Su
02286	05:15	05:20	Tu Sa
12270	05:15	05:20	W Su
12289	07:20	--	M Tu W Th F Sa Su
12290	--	20:40	M Tu W Th F Sa Su
12269	21:00	21:06	M F
02285	21:00	21:06	Th Su
12222	23:15	23:20	Th Sa
12262	23:15	23:20	M Tu W F

Trains

- Train dataset — information about trains and stations
  - Each train is a route list of stations
  - Each station is a route list of trains passing through
- Compute pairs of stations connected by a direct train

Train

M W Th Su

12259

Dist.	Station Name	Arr.	Dep.	Day
0	Kolkata	--	18:31	1
266	Dhanbad	21:52	21:56	1
667	Varanasi	02:20	02:30	2
1014	Kanpur	06:25	06:30	2
1453	New Delhi	11:31	11:45	2
1628	Loharu	14:23	14:26	2
1678	Sadulpur	15:03	15:06	2
1736	Churu	16:06	16:08	2
1779	Ratangarh	17:00	17:02	2
1844	Sri Dungargarah	17:47	17:48	2
1916	Bikaner	19:15	--	2

Station

Nagpur

Train No.	Arr.	Dep.	Days
12261	04:10	04:15	M W Th F
12221	04:10	04:15	Tu Su
02286	05:15	05:20	Tu Sa
12270	05:15	05:20	W Su
12289	07:20	--	M Tu W Th F Sa Su
12290	--	20:40	M Tu W Th F Sa Su
12269	21:00	21:06	M F
02285	21:00	21:06	Th Su
12222	23:15	23:20	Th Sa
12262	23:15	23:20	M Tu W F

Trains

- Train dataset — information about trains and stations
  - Each train is a route list of stations
  - Each station is a route list of trains passing through
- Compute pairs of stations connected by a direct train
- Represent start and end station of trains in a nested dictionary
  - `trains[t][start]`, `trains[t][end]`

Train

M W Th Su

12259

Dist.	Station Name	Arr.	Dep.	Day
0	Kolkata	--	18:31	1
266	Dhanbad	21:52	21:56	1
667	Varanasi	02:20	02:30	2
1014	Kanpur	06:25	06:30	2
1453	New Delhi	11:31	11:45	2
1628	Loharu	14:23	14:26	2
1678	Sadulpur	15:03	15:06	2
1736	Churu	16:06	16:08	2
1779	Ratangarh	17:00	17:02	2
1844	Sri Dungargarah	17:47	17:48	2
1916	Bikaner	19:15	--	2

Station

Nagpur

Train No.	Arr.	Dep.	Days
12261	04:10	04:15	M W Th F
12221	04:10	04:15	Tu Su
02286	05:15	05:20	Tu Sa
12270	05:15	05:20	W Su
12289	07:20	--	M Tu W Th F Sa Su
12290	--	20:40	M Tu W Th F Sa Su
12269	21:00	21:06	M F
02285	21:00	21:06	Th Su
12222	23:15	23:20	Th Sa
12262	23:15	23:20	M Tu W F

Direct routes

- Stations A and B such that a train starts at A and ends at B

```
Procedure DirectRoutes(trains)
```

```
End DirectRoutes
```

Direct routes

- Stations A and B such that a train starts at A and ends at B
- First, compile the list of stations from `trains`

```
Procedure DirectRoutes(trains)
  stations = {}
  foreach t in keys(trains) {
    stations[trains[t][start]] = True
    stations[trains[t][end]] = True
  }
```

```
End DirectRoutes
```

Direct routes

- Stations A and B such that a train starts at A and ends at B
- First, compile the list of stations from `trains`
- Create a matrix to record direct routes

```
Procedure DirectRoutes(trains)
  stations = {}
  foreach t in keys(trains) {
    stations[trains[t][start]] = True
    stations[trains[t][end]] = True
  }
  n = length(keys(stations))
  direct = CreateMatrix(n,n)
```

```
End DirectRoutes
```

Direct routes

- Stations A and B such that a train starts at A and ends at B
- First, compile the list of stations from `trains`
- Create a matrix to record direct routes
- Map station names to row and column indices

```
Procedure DirectRoutes(trains)
  stations = {}
  foreach t in keys(trains) {
    stations[trains[t][start]] = True
    stations[trains[t][end]] = True
  }
  n = length(keys(stations))
  direct = CreateMatrix(n,n)
  stnindex = {}
  i = 0
  foreach s in keys(stations) {
    stnindex[s] = i
    i = i+1
  }
```

```
End DirectRoutes
```

Direct routes

- Stations A and B such that a train starts at A and ends at B
- First, compile the list of stations from `trains`
- Create a matrix to record direct routes
- Map station names to row and column indices
- Populate the matrix

```
Procedure DirectRoutes(trains)
  stations = {}
  foreach t in keys(trains) {
    stations[trains[t][start]] = True
    stations[trains[t][end]] = True
  }
  n = length(keys(stations))
  direct = CreateMatrix(n,n)
  stnindex = {}
  i = 0
  foreach s in keys(stations) {
    stnindex[s] = i
    i = i+1
  }
  foreach t in keys(trains){
    i = stnindex[trains[t][start]]
    j = stnindex[trains[t][end]]
    direct[i][j] = 1
  }
  return(direct)
End DirectRoutes
```

One hop routes

- Stations A and B such that you can reach B from A by changing one train

One hop routes

- Stations A and B such that you can reach B from A by changing one train
- Iterate through intermediate stations
  - Set `onehop[i][j] = 1` if there is a connection via intermediate station `k`

```
Procedure OneHop(direct)
  n = length(keys(direct))
  onehop = CreateMatrix(n,n)
  foreach i in rows(direct) {
    foreach j in columns(direct) {
      foreach k in columns(direct) {
        if (direct[i][k] == 1 and
              direct[k][j] == 1) {
          onehop[i][j] = 1
        }
      }
    }
  }
  return(onehop)
End OneHop
```

One hop routes

- Stations A and B such that you can reach B from A by changing one train
- Iterate through intermediate stations
  - Set `onehop[i][j] = 1` if there is a connection via intermediate station `k`
- More useful to let `onehop[i][j]` mean "connected with at most one hop"
  - Initialize `onehop` to include direct routes

```
Procedure WithinOneHop(direct)
  n = length(keys(direct))
  onehop = CreateMatrix(n,n)
  foreach i in rows(direct) {
    foreach j in columns(direct) {
      onehop[i][j] = direct[i][j]
      foreach k in columns(direct) {
        if (direct[i][k] == 1 and
            direct[k][j] == 1) {
          onehop[i][j] = 1
        }
      }
    }
  }
  return(onehop)
End WithinOneHop
```

Two hop routes

- Stations A and B such that you can reach B from A by changing at most two trains

Two hop routes

- Stations A and B such that you can reach B from A by changing at most two trains
- Extend a one hop route from `i` to `k` by a direct train from `k` to `j`

Two hop routes

- Stations A and B such that you can reach B from A by changing at most two trains
- Extend a one hop route from `i` to `k` by a direct train from `k` to `j`
- Iterate through intermediate stations
  - Combine information in `direct` and `onehop`
  - Check `onehop[i][k]` and `direct[k][j]`

```
Procedure WithinTwoHops(direct,onehop)
  n = length(keys(direct))
  twohops = CreateMatrix(n,n)
  foreach i in rows(direct) {
    foreach j in columns(direct) {
      twohops[i][j] = onehop[i][j]
      foreach k in columns(direct) {
        if (onehop[i][k] == 1 and
            direct[k][j] == 1) {
          twohops[i][j] = 1
        }
      }
    }
  }
  return(twohops)
End WithinTwoHops
```

n hop routes

- Let `nhops` record connections from station A to station B by changing at most *n* trains
- Want to extend `nhops` to allow one more hop

n hop routes

- Let `nhops` record connections from station A to station B by changing at most n trains
- Want to extend `nhops` to allow one more hop
- Iterate through intermediate stations
- Combine information `direct` and `nhops`
  - Extend an n hop route from `i` to `k` by a direct train from `k` to `j`
  - Check `nhops[i][k]` and `direct[k][j]`

```
Procedure OneMoreHop(direct,nhops)
  n = length(keys(direct))
  onemorehop = CreateMatrix(n,n)
  foreach i in rows(direct) {
    foreach j in columns(direct) {
      onemorehop[i][j] = nhops[i][j]
      foreach k in columns(direct) {
        if (nhops[i][k] == 1 and
              direct[k][j] == 1) {
          onemorehop[i][j] = 1
        }
      }
    }
  }
  return(onemorehop)
End OneMoreHop
```

Discovering paths

- Path: sequence of edges from A to B
  - A direct edge is a path of length 1

```
Procedure OneMoreHop(direct,nhops)
  n = length(keys(direct))
  onemorehop = CreateMatrix(n,n)
  foreach i in rows(direct) {
    foreach j in columns(direct) {
      onemorehop[i][j] = nhops[i][j]
      foreach k in columns(direct) {
        if (nhops[i][k] == 1 and
              direct[k][j] == 1) {
          onemorehop[i][j] = 1
        }
      }
    }
  }
  return(onemorehop)
End OneMoreHop
```

Discovering paths

- Path: sequence of edges from A to B
  - A direct edge is a path of length 1
- Procedure OneMoreHop extends paths of length n to paths of length $n + 1$

```
Procedure OneMoreHop(direct,nhops)
  n = length(keys(direct))
  onemorehop = CreateMatrix(n,n)
  foreach i in rows(direct) {
    foreach j in columns(direct) {
      onemorehop[i][j] = nhops[i][j]
      foreach k in columns(direct) {
        if (nhops[i][k] == 1 and
            direct[k][j] == 1) {
          onemorehop[i][j] = 1
        }
      }
    }
  }
  return(onemorehop)
End OneMoreHop
```

Discovering paths

- Path: sequence of edges from A to B
  - A direct edge is a path of length 1
- Procedure OneMoreHop extends paths of length n to paths of length $n+1$
- N nodes — shortest path from A to B has at most $N-1$ edges
  - A longer path would visit a node twice — unnecessary loop

```
Procedure OneMoreHop(direct,nhops)
  n = length(keys(direct))
  onemorehop = CreateMatrix(n,n)
  foreach i in rows(direct) {
    foreach j in columns(direct) {
      onemorehop[i][j] = nhops[i][j]
      foreach k in columns(direct) {
        if (nhops[i][k] == 1 and
            direct[k][j] == 1) {
          onemorehop[i][j] = 1
        }
      }
    }
  }
  return(onemorehop)
End OneMoreHop
```

Discovering paths

- Path: sequence of edges from `A` to `B`
  - A direct edge is a path of length 1
- Procedure `OneMoreHop` extends paths of length n to paths of length $n+1$
- N nodes — shortest path from `A` to `B` has at most $N-1$ edges
  - A longer path would visit a node twice — unnecessary loop
- Transitive closure — pairs of nodes connected by a path
  - Repeat `OneMoreHop` $N-1$ times starting with `direct`

```
Procedure OneMoreHop(direct,nhops)
  n = length(keys(direct))
  onemorehop = CreateMatrix(n,n)
  foreach i in rows(direct) {
    foreach j in columns(direct) {
      onemorehop[i][j] = nhops[i][j]
      foreach k in columns(direct) {
        if (nhops[i][k] == 1 and
            direct[k][j] == 1) {
          onemorehop[i][j] = 1
        }
      }
    }
  }
  return(onemorehop)
End OneMoreHop
```

Summary

- Graphs are useful to represent connectivity in a network
- A path is a sequence of edges
- Starting with direct edges, we can iteratively find longer and longer paths
- Compute the transitive closure of the edge relation
  - Stop with paths of length $N - 1$ for an N node graph