Introduction to Computer Networks
Homework 2: Routing Protocols
Announced: May 18 (Thr), Due: Jun 7 (Wed)

1. 과제의 목표

Distance Vector와 Link State routing algorithm을 이용하여 각 라우터의 라우팅 테이블을 생성하고, 네트워크의 변화에 따라 라우팅 테이블을 업데이트한다.

2. 작성해야 할 프로그램

linkstate: Link State routing algorithm을 이용하여 라우팅 테이블을 관리하는 프로그램
distvec: Distance Vector routing algorithm을 이용하여 라우팅 테이블을 관리하는 프로그램

3. 상세설명 (주의 깊게 읽고 지시사항을 따라 구현할 것)

(1) 이 과제에서는 소켓프로그래밍을 이용하여 직접 네트워크로 메시지를 전달하지 않는다. 가상의 라우터들과 그 라우터들을 잇는 회선들이 있다고 가정하고, 각 라우터의 라우팅 테이블을 계산하고 업데이트 하는 것이 이 과제에서 해야 할 일이다.

(2) linkstate와 distvec은 다음과 같이 실행시킨다. 각 프로그램은 파일 이름 세 개를 입력 인자로 받는다.

```
./linkstate topologyfile messagesfile changesfile
```

```
./distvec topologyfile messagesfile changesfile
```

프로그램 실행 예:

```
./linkstate topology.txt messages.txt changes.txt
```

(3) 프로그램에 들어가는 인자의 수가 맞지 않으면, 다음과 같은 메시지를 출력하고 프로그램을 종료한다.

(linkstate의 경우)
```
usage: linkstate topologyfile messagesfile changesfile
```

(distvec의 경우)
```
usage: distvec topologyfile messagesfile changesfile
```

(4) Input file open시 에러가 발생하면 다음과 같은 메시지를 출력한다.

```
Error: open input file.
```

(5) 프로그램 실행 성공 시 다음과 같은 메시지를 출력한다.

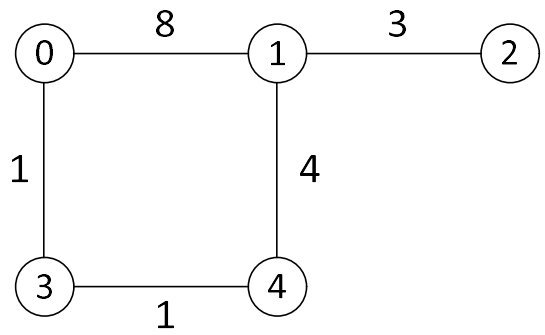
(linkstate의 경우)
```
Complete. Output file written to output_ls.txt.
```

(distvec의 경우)
```
Complete. Output file written to output_dv.txt.
```

(6) Topology file의 포맷은 다음과 같다. 먼저 첫번째 줄에는 네트워크에 있는 노드의 총 개수가 있다. 그 다음 줄부터는 네트워크에 있는 링크(회선)에 대한 정보가 있다.
- 아래는 topology file의 한 예이다.

```
5
0 1 8
1 2 3
1 4 4
3 0 1
3 4 1
```

- 이 topology file은 아래와 같은 형태의 네트워크를 표현한 것이다.



- 첫번째 줄에서 네트워크에 있는 노드의 수가 5개임을 알 수 있다. 노드의 ID는 0번부터 시작하여 연속적으로 부여된다고 가정한다. 따라서 노드 5개의 ID는 0번부터 4번까지이다.
- 두번째 줄은 "0 1 8"인데, 이는 "0번 노드와 1번 노드를 잇는 링크가 있으며 그 링크의 cost는 8"이라는 뜻이다. 방향성이 없기 때문에 "0 1 8"과 "1 0 8"은 같다.
- 두번째 줄부터는 각 줄이 하나의 링크를 의미한다.

(7) Messages file에서는 각 줄이 하나의 메시지를 의미하며, 각 메시지의 포맷은 아래와 같다.

source destination message

- 아래는 messages file의 한 예이다.

```
1 0 here is a message from 1 to 0
2 4 this one gets sent from 2 to 4!
```

- 첫번째 줄의 의미는, 노드 1번에서 노드 0번으로 메시지를 전송하였으며 그 메시지의 내용은 "here is a message from 1 to 0"이라는 것이다. 각 줄에서 맨 처음 나오는 숫자가 송신 노드의 ID이고, 두 번째로 나오는 숫자가 수신 노드의 ID이고, 그 다음부터 메시지의 내용이 된다.

(8) Changes file에는 네트워크의 상태 변화가 적혀 있다. 아래는 changes file의 한 예를 보여준다.

```
1 3 1
1 3 -999
```

- 위와 같이 두 줄이 있으면, 이 네트워크에서는 두 번의 변화가 있었다는 뜻이다.
- 첫번째 줄의 "1 3 1"은, 1번 노드와 3번 노드를 잇는 링크의 cost가 1로 바뀌었다는 뜻이다.
- 링크의 cost가 -999로 되어 있으면, 그 노드 사이에는 링크가 없는 것으로 간주한다. 위의 changes file에서 두번째 줄에 있는 "1 3 -999"는 1번과 3번 사이의 링크가 끊어졌다는 의미가 된다.

(9) 프로그램에서 하는 일은 다음과 같다. 가장 먼저 topology file을 읽어 들여서 네트워크를 구성한다. 그런 다음 link state또는 distance vector 알고리즘을 이용하여 각 노드의 라우팅 테이블을 구성하는 것이다.
- 그리고 난 다음, messages file에 있는 각 줄의 message를 라우팅 테이블을 통해 송신자부터 수신자까지 전달하는 과정을 simulation한다. 여기 까지가 step 0이다.
- Step 0을 마쳤으면, changes file 첫번째 줄에 있는 상태 변화를 적용한다. 적용하고 나면 특정 링크의 cost가 바뀐다. (없던 링크가 생기거나 있던 링크가 없어질 수도 있다.)
- 그리고는 다시 link state와 distance vector 알고리즘을 이용하여 각 노드의 라우팅 테이블을 업데이트한다.
- 그리고 난 다음, messages file에 있는 각 줄의 message를 라우팅 테이블을 통해 송신자부터 수신자까지 전달하는 과정을 simulation한다. 여기 까지가 step 1이다.
- 다시 이번에는 changes file의 두 번째 줄에 있는 엔트리를 이용하여 네트워크를 변화시킨다.
- 그런 다음에는 다시 라우팅 테이블을 업데이트하고, message file의 메시지들을 처리하고, changes file에 의해 네트워크를 변형한다. 이 과정은 changes file의 맨 마지막 줄까지 네트워크에 반영이 되면 끝난다.

(10) distvec의 출력파일은 output_dv.txt로 하고, linkstate의 출력파일은 output_ls.txt로 한다.
- 출력파일의 포맷은 distvec과 linkstate가 공통이며, 다음과 같다.
- 맨 먼저, changes file의 내용을 적용시키기 전, topology file에서 제시된 네트워크 토폴로지에 대하여 converge된 라우팅 테이블을 출력한다. 라우팅 테이블 출력은 0번 노드의 테이블부터 출력하며, 각각의 테이블 사이에는 공백 줄 한 줄을 넣는다.
- 라우팅 테이블은 각 줄에 **목적지 다음 거리**와 같은 형태로 출력하면 되며, 목적지 ID로 정렬해서 출력한다.
- 아래는 위의 그림에 나타난 네트워크에 대한 라우팅 테이블을 출력한 것이다.

```
0 0 0
1 3 6
2 3 9
3 3 1
4 3 2

0 4 6
1 1 0
2 2 3
3 4 5
4 4 4

0 1 9
1 1 3
2 2 0
3 1 8
4 1 7

0 0 1
1 4 5
2 4 8
3 3 0
4 4 1

0 3 2
1 1 4
2 1 7
3 3 1
4 4 0
```

- 이렇게 라우팅 테이블 출력이 끝나고 나면, messages file에 있는 메시지를 송신자부터 수신자로 전달한다고 가정하고, 그 내용을 출력한다.
- message에 대한 출력 포맷은 다음과 같다. 만약 송신자 x로부터 수신자 y로의 경로가 존재한다고 하면,

```
from <x> to <y> cost <path_cost> hops <hop1> <hop2> <...> message <message>
```

- 예를 들어, messages file의 첫번째 라인이 "1 0 here is a message from 1 to 0"었다고 하자. 그럼 "here is a message from 1 to 0"이라는 문자열을 1번 노드로부터 0번 노드로 전송한다고 가정한다. 이것에 대한 출력 내용은 아래와 같다.

```
from 1 to 0 cost 6 hops 1 4 3 message here is a message from 1 to 0
```

- 위의 출력 내용에서 hops 다음에 출력하는 내용은 메시지의 전달 경로 상에 있는 노드들의 ID 이다. 메시지는 1번 노드에서 출발하여 4번, 3번을 거쳐서 목적지인 0번 노드에 도달하게 된다. 이 출력에서는 송신자는 포함하고 수신자는 포함시키지 않는다.

- 만약 송신자 x로부터 수신자 y로의 경로가 존재하지 않으면, 다음과 같이 출력한다.

```
from <x> to <y> cost infinite hops unreachable message <message>
```

- 이런 식으로 messages file 각 라인을 처리한다.

- messages file에 대한 처리가 끝나고 나면, changes file의 네트워크 변경 내용 한 줄을 적용한다.

- 예를 들어, changes file의 첫번째 줄이 "1 3 1" 이라고 하면, 이는 1번 노드와 3번 노드 사이에 링크가 있고 그 cost가 1이라는 뜻이다. 초기 네트워크에서는 1번과 3번 사이에 링크가 없었는데, 이 변경사항을 적용하고 나면 1번 노드와 3번 노드 사이에 새로운 링크가 생기는 것이다.
- 그러면 이것을 반영하여 distvec이나 linkstate 모두 각자의 라우팅 알고리즘으로 새로운 네트워크에 대하여 라우팅 테이블을 업데이트 하면 된다.
- 업데이트가 끝나고 나면, 다시 출력 파일에 각 라우터의 라우팅 테이블을 출력하고, 그 다음에 message file에 있는 라인들을 출력해 주면 된다.
- 라우팅 테이블에 경로가 없으면 출력하지 않는다.

(11) 아래는 예제로 주어지는 topology.txt, messages.txt, changes.txt 파일의 내용과, 이를 이용한 distvec과 linkstate의 출력 파일의 내용이다. (message나 change는 빈 파일이 input으로 주어질 수 있다.)

```
[topology.txt]
5
0 1 8
1 2 3
1 4 4
3 0 1
3 4 1

[messages.txt]
1 0 here is a message from 1 to 0
2 4 this one gets sent from 2 to 4!

[changes.txt]
1 3 1
1 3 -999

[output_dv.txt] [output_ls.txt] (공통)
```

```
0 0 0
1 3 6
2 3 9
3 3 1
4 3 2

0 4 6
1 1 0
2 2 3
3 4 5
4 4 4

0 1 9
1 1 3
2 2 0
3 1 8
4 1 7

0 0 1
1 4 5
2 4 8
3 3 0
4 4 1

0 3 2
1 1 4
2 1 7
3 3 1
4 4 0

from 1 to 0 cost 6 hops 1 4 3 message here is a message from 1 to 0
from 2 to 4 cost 7 hops 2 1 message this one gets sent from 2 to 4!

0 0 0
1 3 2
2 3 5
3 3 1
4 3 2

0 3 2
1 1 0
2 2 3
3 3 1
4 3 2

0 1 5
1 1 3
2 2 0
3 1 4
4 1 5

0 0 1
1 1 1
2 1 4
3 3 0
4 4 1

0 3 2
```

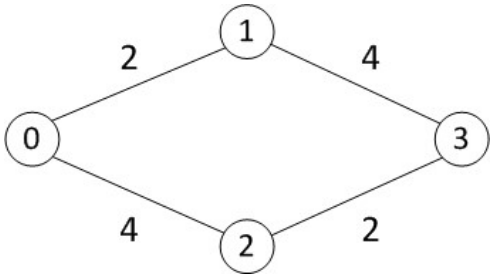
```
1 3 2
2 3 5
3 3 1
4 4 0

from 1 to 0 cost 2 hops 1 3 message here is a message from 1 to 0
from 2 to 4 cost 5 hops 2 1 3 message this one gets sent from 2 to 4!

0 0 0
1 3 6
2 3 9
3 3 1
4 3 2

0 4 6
1 1 0
2 2 3
3 4 5
4 4 4

0 1 9
1 1 3
2 2 0
3 1 8
4 1 7

0 0 1
1 4 5
2 4 8
3 3 0
4 4 1

0 3 2
1 1 4
2 1 7
3 3 1
4 4 0

from 1 to 0 cost 6 hops 1 4 3 message here is a message from 1 to 0
from 2 to 4 cost 7 hops 2 1 message this one gets sent from 2 to 4!
```

- 위의 예에서는 distvec의 출력과 linkstate의 출력이 동일하였다. 하지만 **반드시 그렇지는 않은데**, 그 이유는 다음에 설명할 tie-breaking rule 때문이다.

(12) Distance vector와 link state 알고리즘을 실행하다 보면 tie-breaking이 필요한 부분이 세 군데 있다. 결과의 일관성을 위하여 반드시 다음의 tie-breaking rule을 적용해서 경로를 계산하도록 한다.

- Tie-breaking rule 1 (distance-vector): distance vector에서는 이웃 노드로부터 라우팅 테이블을 받아서, 어떤 목적지에 대하여 지금 현재 내가 가지고 있는 경로보다 더 cost가 적은 경로가 있는지를 확인하게 된다. 예를 들어, 아래 그림에서 0번 노드가 3번 노드로 메시지를 전송하려면 1번

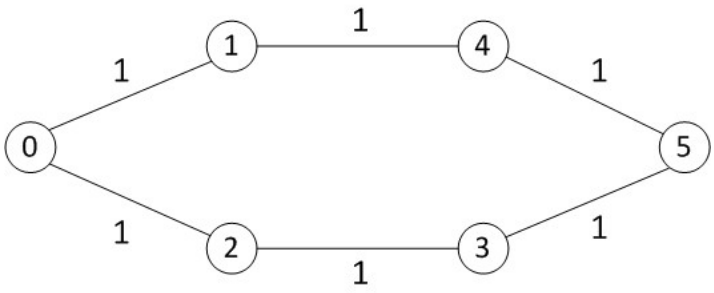
노드를 다음 노드로 해도 되고 2번 노드를 다음 노드로 해도 된다. 이런 경우에는 **ID 값이 작은 1번 노드를 다음 노드로 선택**한다.



- Tie-breaking rule 2 (link-state): link state routing에서 Dijkstra's algorithm을 실행하다 보면, 각 stage에서 SPT에 넣을 노드를 선택해야 한다. 이 때 source로부터 distance가 가장 적은 노드를 선택하는데, distance가 가장 적은 노드가 여러 개 있을 경우에는 **ID 값이 작은 노드를 우선적으로 선택하여 SPT에 넣는다**.

- Tie-breaking rule 3 (link-state): link state routing에서도 각 노드가 현재의 경로와 새로운 경로 중에서 어떤 것을 선택할지 판단하는 장면이 있다. 즉, 현재의 parent 노드와 새로운 parent 노드 중에서 distance 값이 작아지는 parent를 선택해야 하는데, 이 때 distance가 같은 parent가 여러 개 있으면 **ID 값이 작은 노드를 parent로 선택한다**.

- 이러한 tie-breaking rule을 적용하면, 대부분의 경우에는 distvec과 linkstate의 결과가 동일하지만, 그렇지 않은 경우가 있다. 아래 그림에 있는 네트워크가 그렇다.



(13) 입력 파일 중 messages file과 changes file은 빈 파일일 수 있다. Messages file이 빈 파일이면 전송하는 메시지가 없다는 뜻이고, changes file에 내용이 없으면 네트워크 변경이 없다고 보면 된다.

(14) 라우팅 테이블과 메시지 전달은 changes file의 라인을 적용하기 전에 한번 출력하고, changes file의 각 라인을 적용할 때마다 출력해야 한다. 다시 말해 changes file에 라인이 두 줄 있으면 다음과 같이 출력되어야 한다.

routing tables ...

messages ...

routing tables ...

messages ...

routing tables ...

messages ...

(15) 노드의 개수는 최대 100개이며, 링크 cost의 최대 값은 100, 메시지 스트링의 최대 길이는 1000 byte로 가정하면 된다. Messages와 changes file은 100라인을 넘지 않는다.

(16) 입력된 노드는 모두 하나의 네트워크만을 이룬다고 가정한다.

4. 과제 제출

(1) 제출해야 하는 파일은 두 개이며, 그 이름은 아래와 같다. 빨간 색 부분은 자기 학번으로 변경한다.

```
distvec_20200001.cc
linkstate_20200001.cc
```

(2) 이 파일은 cspro에서 g++ 컴파일러로 다음과 같이 컴파일 될 예정이다.

```
g++ -o -Wall distvec_20200001 distvec_20200001.cc
g++ -o -Wall linkstate_20200001 linkstate_20200001.cc
```

- 제출하기 전에 컴파일과 실행이 잘 되는지 테스트해보고 제출한다.
- 컴파일시 경고문이 발생해서는 안 된다.

(3) 아이디어는 상의할 수 있으나, 코드를 표절하지 말 것. 수강생들 사이뿐만 아니라 인터넷에 있는 코드도 동일하게 사용하지 말아야 한다. 표절인 경우에는 0점 처리된다.

(4) 지각 제출은 마감일 후 3일까지 허용하며, 하루에 10%씩 감점한다.