

Project WAN2

- Implement two functions in file `fwdopt.cpp`

```
next_hops stack::optimize_routes(byte src)
```

where return type `next_hops` is defined as

```
typedef struct {  
    byte router[MAX_NETS+1];  
} next_hops;
```

such that `router[i]` gives the ID of the next hop router to reach destination net `i` from `src`.

- This function implements Dijkstra's algorithm.

CS 656

1

- `byte stack::forward_iface`

```
(subnet_state* thisnet, byte destnet)
```

- use `this_net->netnum` to get the ID of the calling router

- the routing table of the calling router is stored in `this_net->routes`, which is of type `next_hops`.

- To determine the cost of a link, use

```
float network::cost(this, from, to)
```

in file `cost.cpp`

- When in doubt about NW data structures, check the header file `hw.h`

CS 656

2

- ❑ To compile, `cwkb wan2`
- ❑ The outcome will be printed in the section “initial packet forwarding matrix” in `diskout.txt`
- ❑ For this project, do **NOT** use the pseudo code in Dr. Pullen’s textbook.
 - the pseudo code leads to $O(N^3)$ implementation of Dijkstra’s shortest path algorithm
- ❑ All projects are individual efforts

CS 656

3

Shortest-Path Computation: Dijkstra’s Algorithm

1. $dist[start] = 0$ and $dist[i] = \infty$ for other vertices i
2. $nhop[i] = -1$ for any vertex i (*nhop* for next hop)
3. Repeat the following until all vertices are marked.
 1. Let u be the vertex, among the unmarked, with the smallest distance; mark u .
 2. For all neighbors v of u ,
 $dist[v] = \min\{dist[v], dist[u] + cost(u,v)\}$
 3. For any neighbor v of u , if $dist[v]$ was changed in step 3.2 to $dist[u] + cost(u,v)$, then

$$nhop[v] = \begin{cases} v & \text{if } u = start \\ nhop[u] & \text{otherwise} \end{cases}$$

CS 656

4

Example

Dist

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

Nhop

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

