ENGS 104, W2010
Lecture 16
Optimization Methods for Engineering Applications

George Cybenko

Reduce 0-1 Knapsack to Partition

- Suppose we can solve the Partition Problem somehow
- How can we easily transform a 0-1 Knapsack problem into a Partition Problem whose solution solves the 0-1 Knapsack Problem

Today's agenda

- Shortest Path Problem
- Minimal Spanning Tree Problem
- Approximate solutions of NP-Complete Problems
  - Definition
  - Triangular TSP
  - Non-Triangular TSP

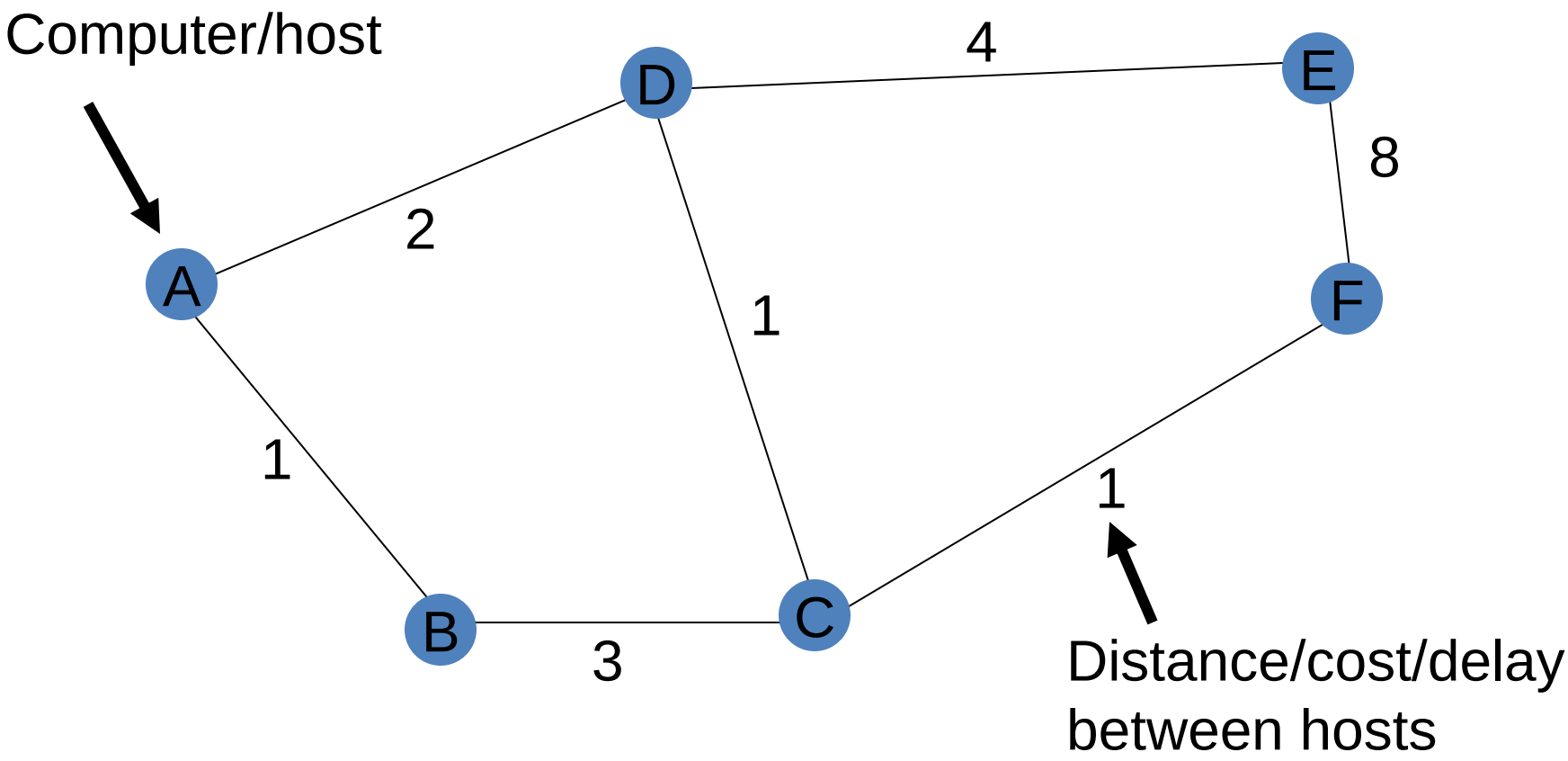
Minimal Spanning Tree Problem

- Definition
  - Tree
  - Spanning Tree
  - Cost of a tree
- A Greedy Algorithm (Kruskal)
- “Best” algorithms
- Compare with “Steiner’s Problem”

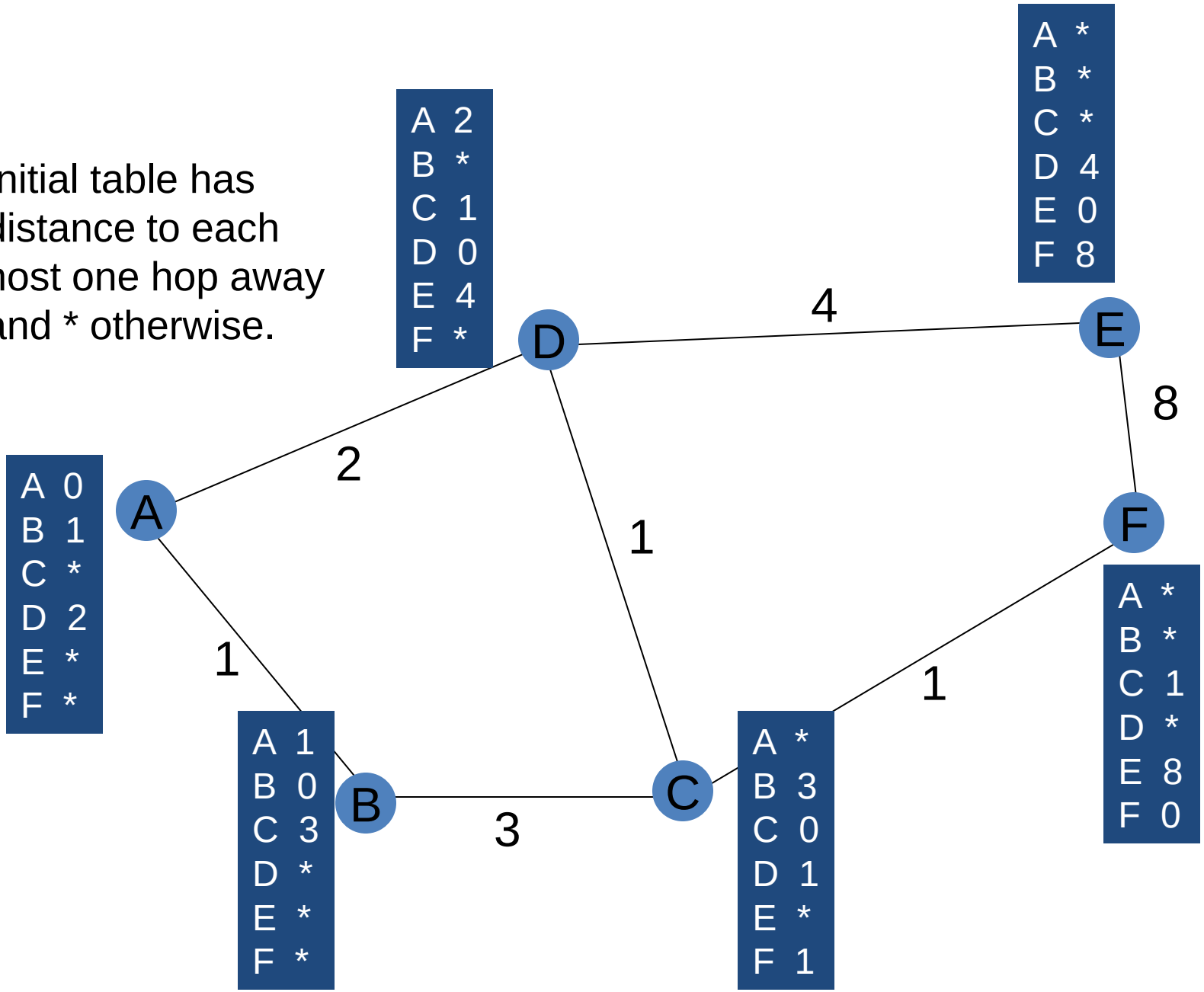
Shortest Path Problem

- Definition
- Basic Property (basis of dynamic programming)
- Some algorithms

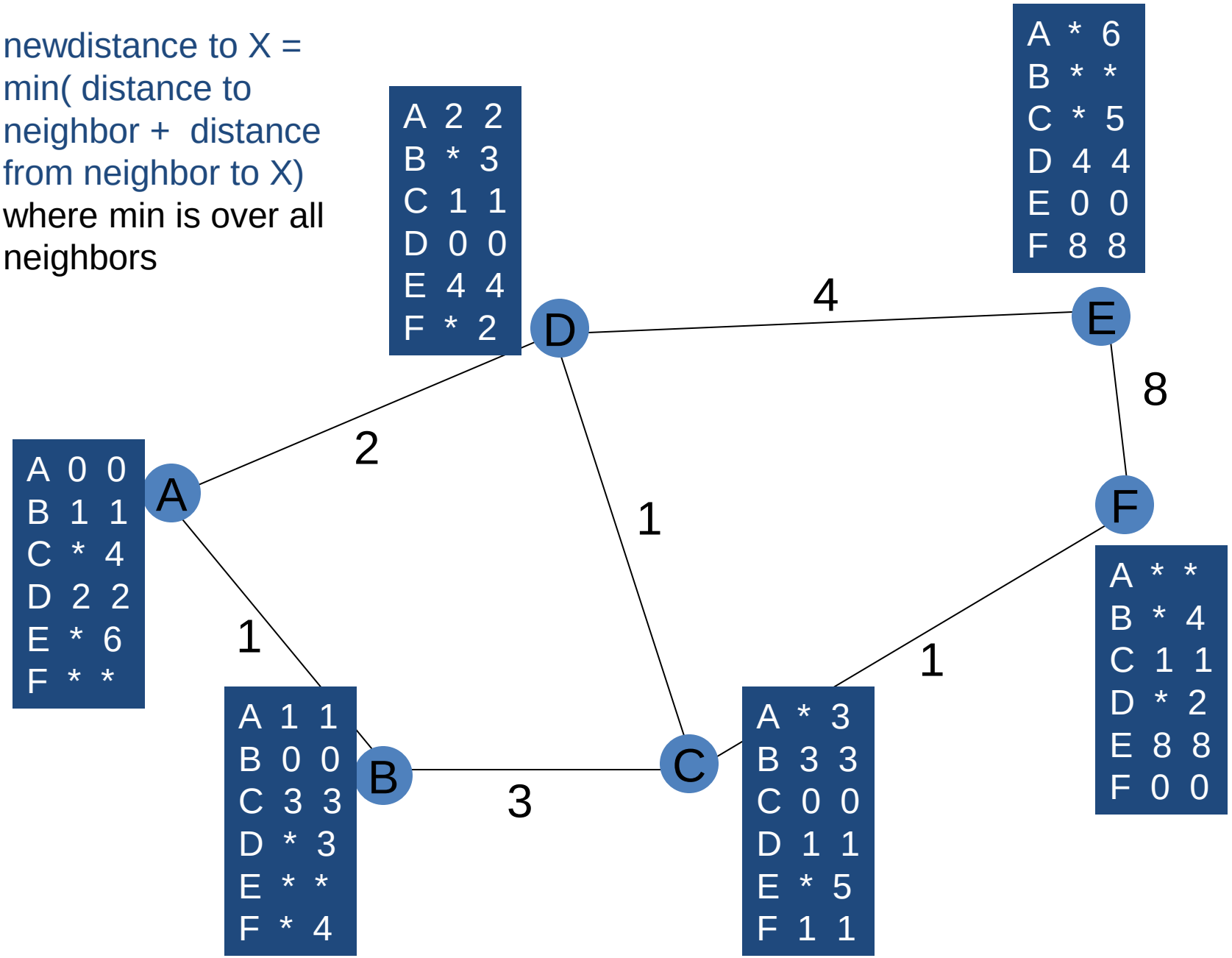
Bellman-Ford Routing

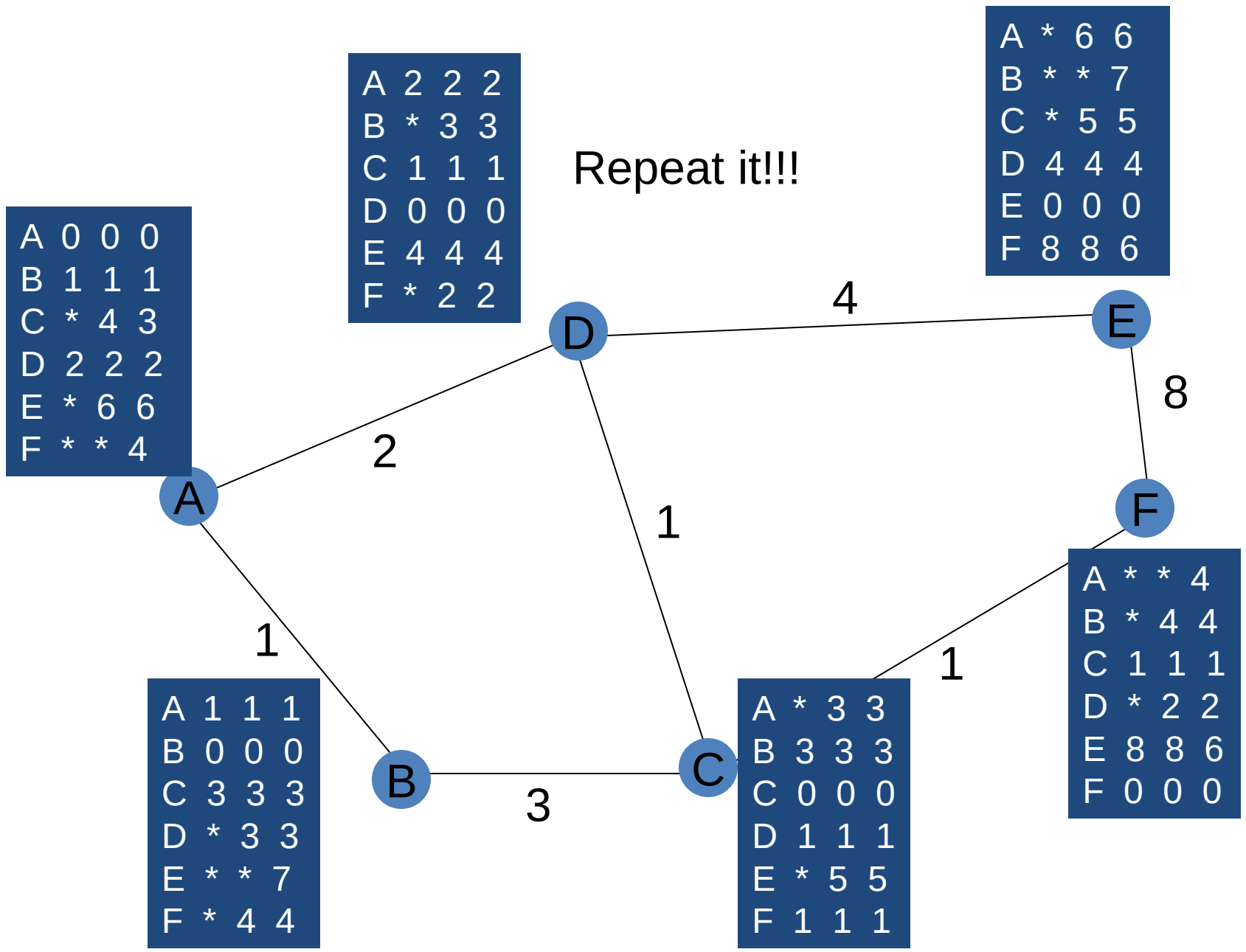


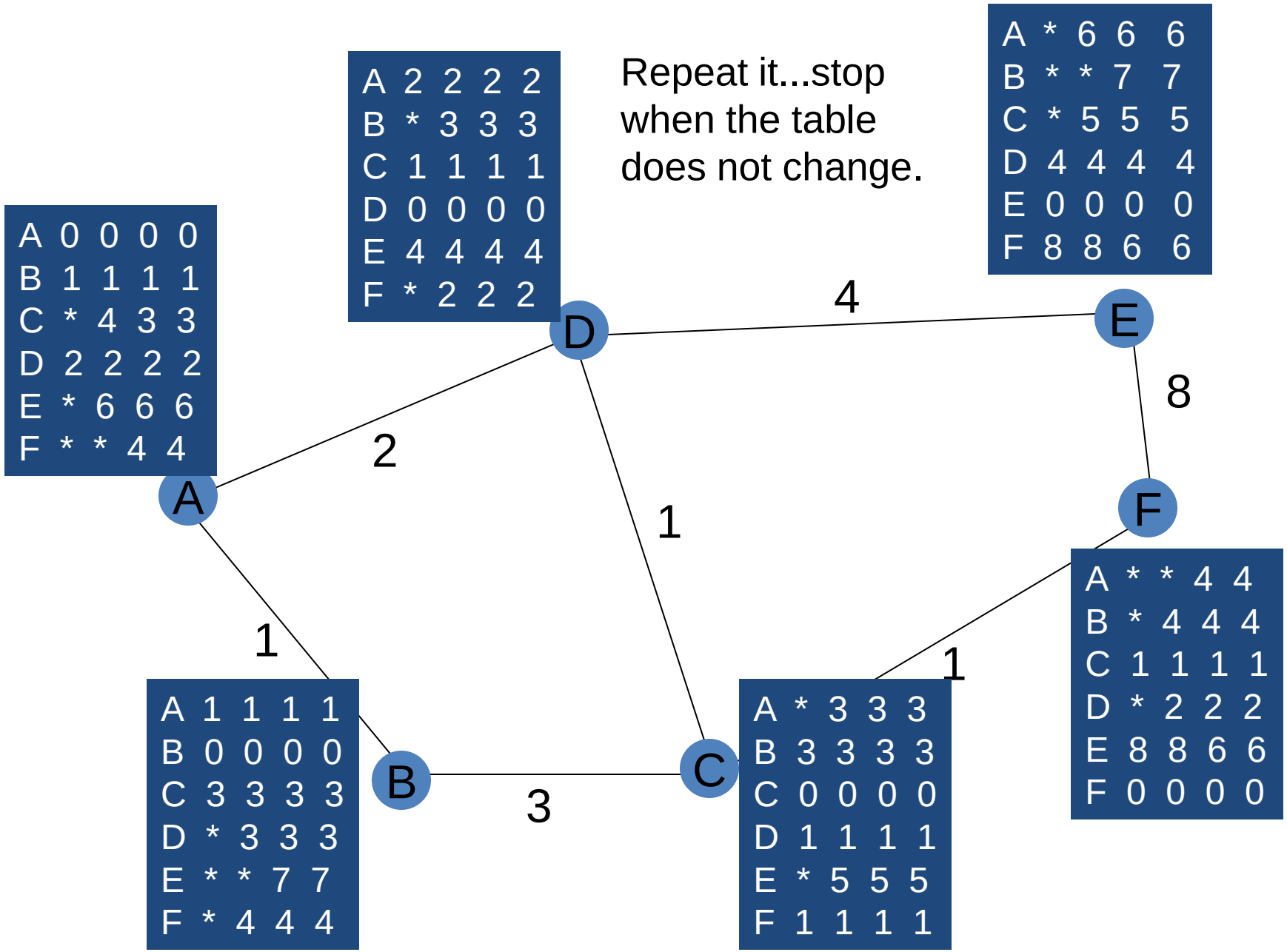
Initial table has distance to each host one hop away and * otherwise.

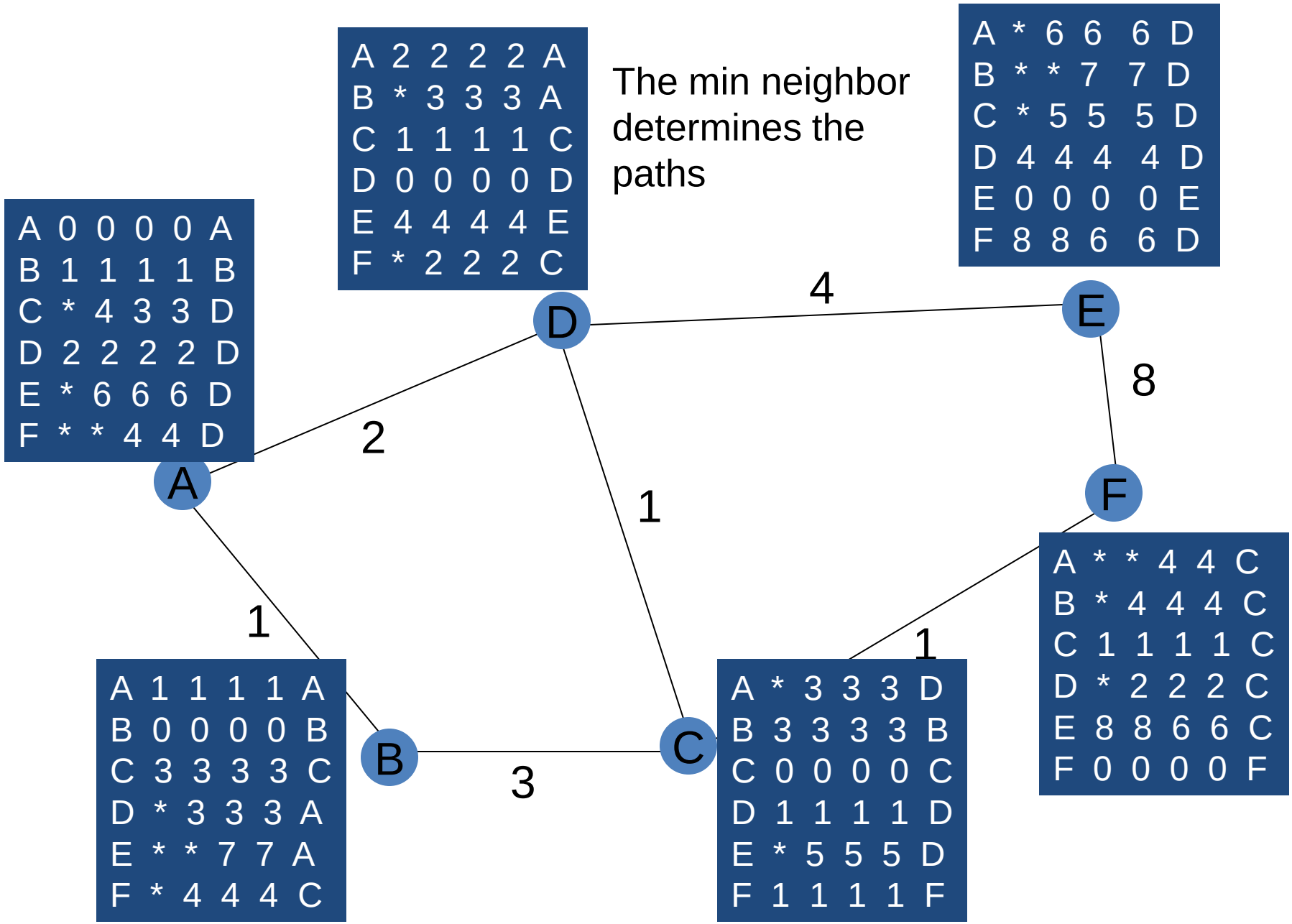


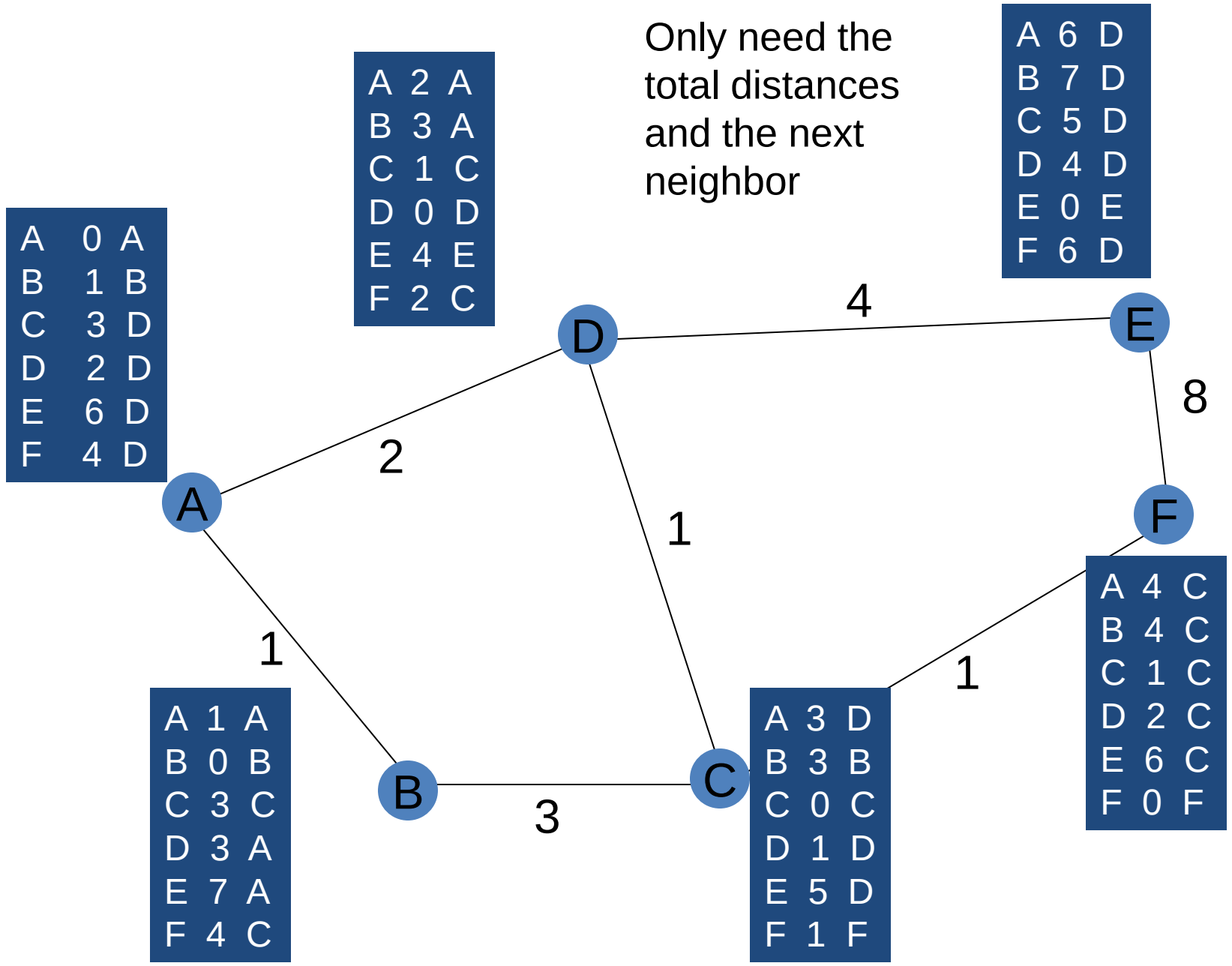
newdistance to X =
min(distance to neighbor + distance from neighbor to X)
where min is over all neighbors

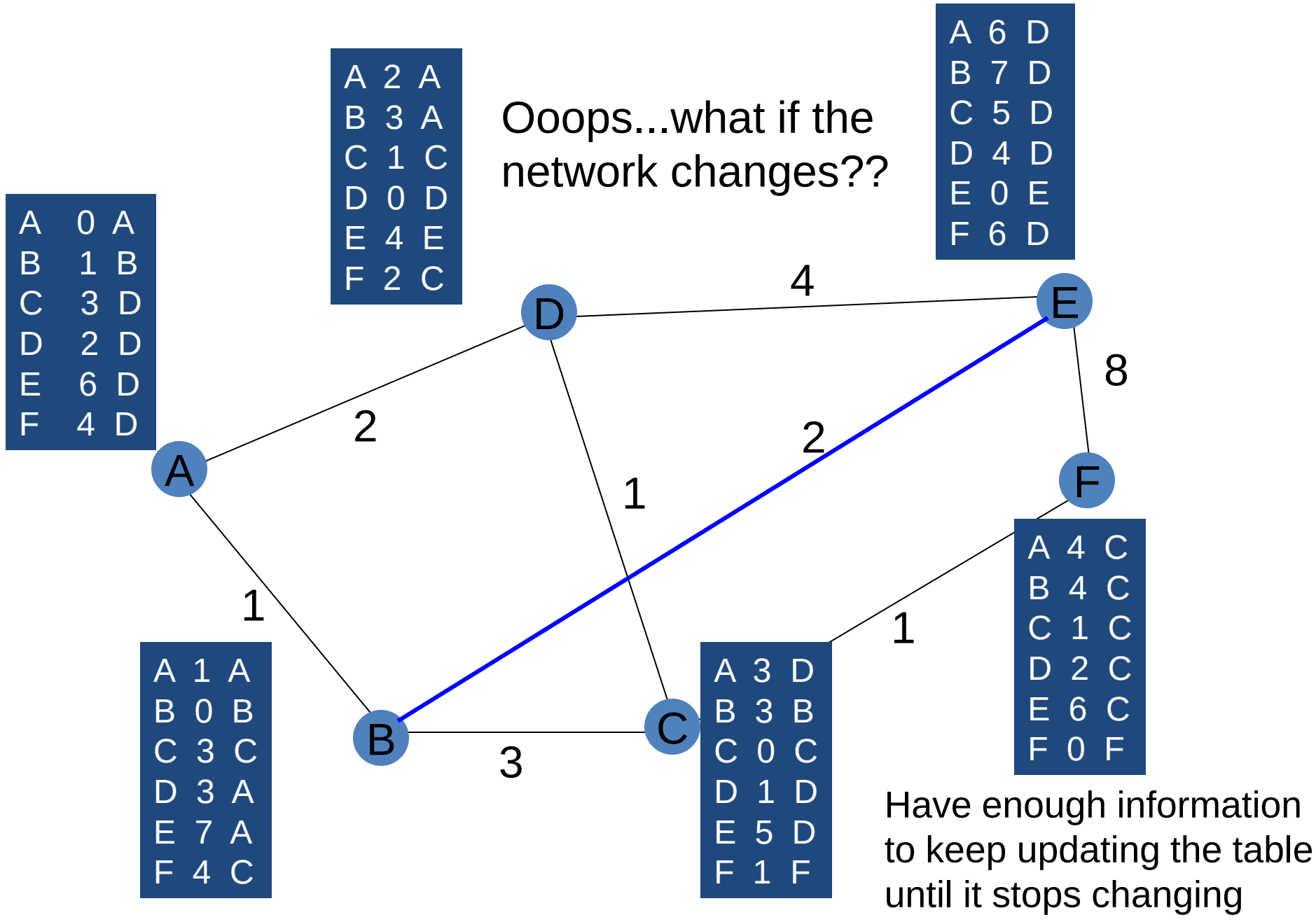



A 0 0 0
B 1 1 1
C * 4 3
D 2 2 2
E * 6 6
F * * 4
A 2 2 2
B * 3 3
C 1 1 1
D 0 0 0
E 4 4 4
F * 2 2
Repeat it!!!
A * 6 6
B * * 7
C * 5 5
D 4 4 4
E 0 0 0
F 8 8 6
A
B
C
D
E
F
2
4
8
1
1
3
1
A 1 1 1
B 0 0 0
C 3 3 3
D * 3 3
E * * 7
F * 4 4
A * 3 3
B 3 3 3
C 0 0 0
D 1 1 1
E * 5 5
F 1 1 1
A * * 4
B * 4 4
C 1 1 1
D * 2 2
E 8 8 6
F 0 0 0


A 0 0 0 0
B 1 1 1 1
C * 4 3 3
D 2 2 2 2
E * 6 6 6
F * * 4 4
A 2 2 2 2
B * 3 3 3
C 1 1 1 1
D 0 0 0 0
E 4 4 4 4
F * 2 2 2
Repeat it...stop when the table does not change.
A * 6 6 6
B * * 7 7
C * 5 5 5
D 4 4 4 4
E 0 0 0 0
F 8 8 6 6
A 1 1 1 1
B 0 0 0 0
C 3 3 3 3
D * 3 3 3
E * * 7 7
F * 4 4 4
A * 3 3 3
B 3 3 3 3
C 0 0 0 0
D 1 1 1 1
E * 5 5 5
F 1 1 1 1
A * * 4 4
B * 4 4 4
C 1 1 1 1
D * 2 2 2
E 8 8 6 6
F 0 0 0 0
A
B
C
D
E
F
2
4
8
1
1
1
3


A 0 0 0 0 A
B 1 1 1 1 B
C * 4 3 3 D
D 2 2 2 2 D
E * 6 6 6 D
F * * 4 4 D
A 2 2 2 2 A
B * 3 3 3 A
C 1 1 1 1 C
D 0 0 0 0 D
E 4 4 4 4 E
F * 2 2 2 C
The min neighbor determines the paths
A * 6 6 6 D
B * * 7 7 D
C * 5 5 5 D
D 4 4 4 4 D
E 0 0 0 0 E
F 8 8 6 6 D
A
B
C
D
E
F
4
8
2
1
1
3
1
A 1 1 1 1 A
B 0 0 0 0 B
C 3 3 3 3 C
D * 3 3 3 A
E * * 7 7 A
F * 4 4 4 C
A * 3 3 3 D
B 3 3 3 3 B
C 0 0 0 0 C
D 1 1 1 1 D
E * 5 5 5 D
F 1 1 1 1 F
A * * 4 4 C
B * 4 4 4 C
C 1 1 1 1 C
D * 2 2 2 C
E 8 8 6 6 C
F 0 0 0 0 F


Only need the total distances and the next neighbor
A 0 A
B 1 B
C 3 D
D 2 D
E 6 D
F 4 D
A 2 A
B 3 A
C 1 C
D 0 D
E 4 E
F 2 C
A 6 D
B 7 D
C 5 D
D 4 D
E 0 E
F 6 D
A 1 A
B 0 B
C 3 C
D 3 A
E 7 A
F 4 C
A 3 D
B 3 B
C 0 C
D 1 D
E 5 D
F 1 F
A 4 C
B 4 C
C 1 C
D 2 C
E 6 C
F 0 F
A
B
C
D
E
F
2
4
8
1
1
3
1


A 0 A
B 1 B
C 3 D
D 2 D
E 6 D
F 4 D
A 2 A
B 3 A
C 1 C
D 0 D
E 4 E
F 2 C
A 6 D
B 7 D
C 5 D
D 4 D
E 0 E
F 6 D
Ooops...what if the network changes??
A
B
C
D
E
F
4
8
2
2
1
1
1
3
A 1 A
B 0 B
C 3 C
D 3 A
E 7 A
F 4 C
A 3 D
B 3 B
C 0 C
D 1 D
E 5 D
F 1 F
A 4 C
B 4 C
C 1 C
D 2 C
E 6 C
F 0 F
Have enough information to keep updating the table until it stops changing