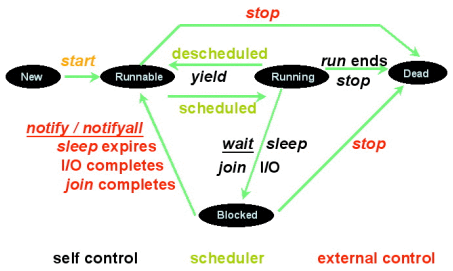
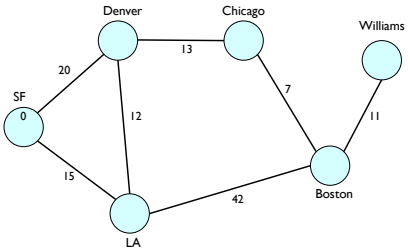
Model of Concurrency in Java

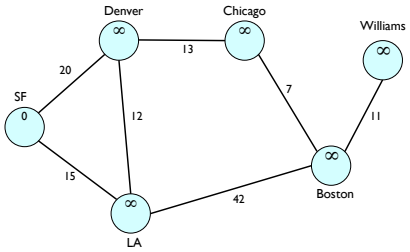


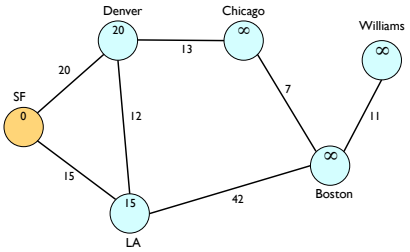
www.traceroute.org

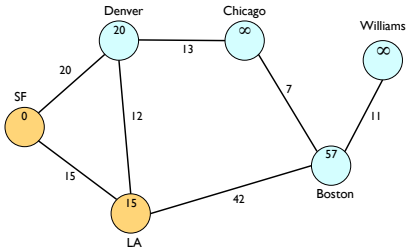
England (Othello Technology) to longhorn.cs.pomona.edu:

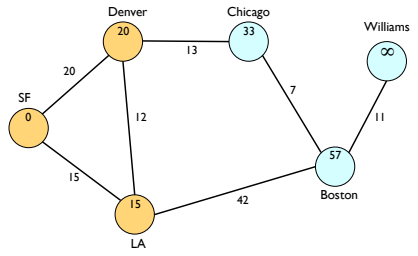
```
traceroute to linus.cs.pomona.edu (12.127.17.83)
 1  80.82.140.227 (80.82.140.227)  0.276 ms  0.243 ms  0.272 ms
 2  transit1.othellotech.net (80.82.140.41)  0.612 ms  0.464 ms  0.615 ms
 3  router1.othellotech.aexiomus.net (80.82.128.221)  0.616 ms  0.875 ms  0.602 ms
 4  peer4.rbl-hex.aexiomus.net (80.82.128.42)  1.241 ms  0.915 ms  0.970 ms
 5  ge-2-0-173.ipcolo2.London1.Level3.net (212.113.10.213)  1.320 ms  1.309 ms  1.117
ms
 6  ae-0-54.bbr2.London1.Level3.net (212.187.131.146)  1.274 ms  1.354 ms  1.976 ms
 7  ae-0-0.bbr1.Dallas1.Level3.net (64.159.1.109)  106.307 ms  106.947 ms  106.643 ms
 8  so-3-0-0.mp1.Tustin1.Level3.net (209.247.8.118)  138.213 ms  138.236 ms  138.054
ms
 9  pos9-0.hsa1.tustin1.level3.net (4.68.114.6)  138.150 ms  138.938 ms
    pos8-0.hsa1.tustin1.level3.net (4.68.114.2)  137.106 ms
10  ln-usc-gsr-level3.ln.net (67.30.130.66)  139.068 ms  139.047 ms  139.381 ms
11  claremont-vlan2016.ln.net (130.152.181.89)  143.198 ms  142.025 ms  142.903 ms
12  Pomona-3.Router.Claremont.Edu (134.173.254.44)  143.474 ms  142.474 ms  142.169 ms
```

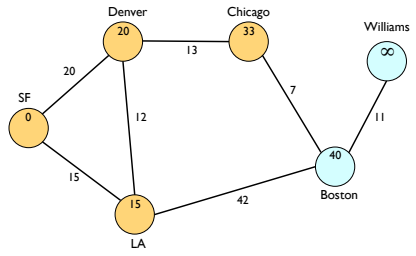


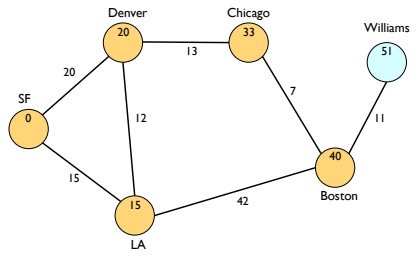


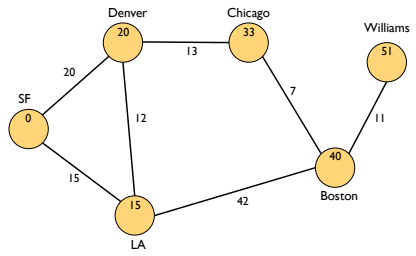












Dijkstra's Algorithm

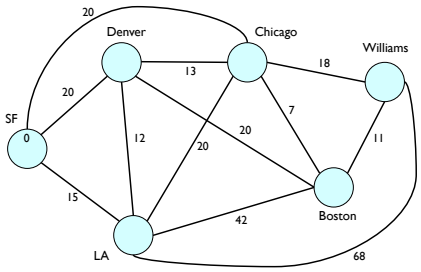
```
for each node n in G {
   dist[n] = INFINITY
}

dist[source] = 0

while (dist[dest] == INFINITY) {
  next = indexOfSmallestUnfixedDistance(dist);
  next.setFixed();
  for each n connected to next {
    if (G(next, n) + dist[next] < dist[n]){
      dist[n] = G(next, n) + dist[next];
    }
  }
}
```

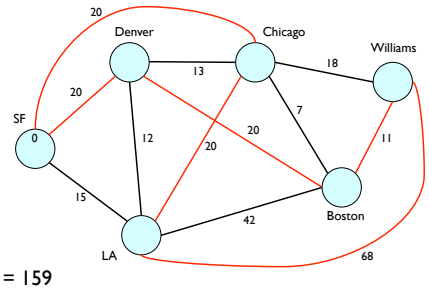
Running Time

- Size of Graph:
  - V is number of vertices
  - E is number of edges
- $T(V,E) \sim E * \log_2(V)$

Travelling Salesmen

- What is the shortest path in the graph that visits each node exactly once?





Distance = 159

Best Known Algorithm

- Try every possible path and see which one is shortest.
- Number of possible paths:
- Running Time: $T(V) \sim 2^V$

But, Wait There's More

- Another Problem:
  Can we determine whether a given Java program will ever halt when started with no input?
- Called "Halting problem"

- Turing proved the problem can never be solved.
- Suppose someone claims they have a solution:
  - a method that takes a string with the name of the file as a parameter, and then returns true or false depending on whether or not the file contains a program that halts.

```java
/* Charlatan class contains method halts, which
   takes a filename as input returning true iff
   the program in the file is legal and halts
   on empty input. */

public class Debunker {

   public void what(String fileName) {
      Charlatan charlatan = new Charlatan();
      if (charlatan.halts(fileName)) {
         while (true){}   // run forever!
      }
   }     // else halt

   public static void main(String[] args) {
      Debunker debunk = new Debunker();
      debunk.what("Debunker.java");}
   }
}
```

- If `charlatan.halts("Debunker.java")` returns true then method `what` enters while loop
  - runs forever -> doesn't halt!
- If `charlatan.halts("Debunker.java")` is false then procedure completes
  - halts!
- What do we know about charlatan.halts?

The end ...