1. specifications

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?

input: *n* cities and the distance *d*(*i*,*j*) between each pair of cities *i*,*j*

output: ordering of cities

legal solution: all cities included exactly once

optimization goal: shortest total distance

2. size

3. examples

4. targets

5. tactics

6. approaches

ordering

process input – for each city, where in the ordering do we visit it?

produce output – what's the next city to visit?

Produce output seems easier to define, and only requires appending to the partial solution so far.

7. generalize / define subproblems

    a) partial solution

an ordering of the first *k* cities visited

    b) alternatives

each of the remaining unvisited cities

    c) subproblem

Given a list of cities, the distances between each pair of cities, and an ordering of cities visited so far, what is the shortest possible route that visits the remaining cities and returns to the origin city?

input: set *C* of *n*-*k* remaining cities, the distance *d*(*i*,*j*) between each pair of cities *i*,*j*, the ordering *T* of the *k* cities visited so far

output: ordering of remaining cities, total distance

legal solution: all remaining cities included exactly once

optimization goal: shortest total distance (including returning to the start after the last city)

8. base case(s)

no cities remaining – all have been visited

solution is empty list, but total distance is the distance to return to the origin city

9. main case

for each c of the n-k remaining cities

```
    ($T_c$,$d_c$) = tsp(C-{c},$T$ with $c$ appended)
    return (c with $T_c$ appended,$c$+$d_c$) for the solution with max $d_c$
```

10. top level
    a) initial subproblem
    b) setup
    c) wrapup

11. special cases

~~12. algorithm~~

13. termination
    a) making progress
    b) reaching the end

14. correctness
    a) establish the base case(s)
    b) show the main case
    c) final answer

15. implementation
    a) memoization

cities left - need

distances between them – global info

ordering so far – what matters is the current city – index in the original list, the origin city – global info

    b) order of computation
    c) dynamic programming

16. time and space