

computing science and biology (2)

mazes, tours, and genomes

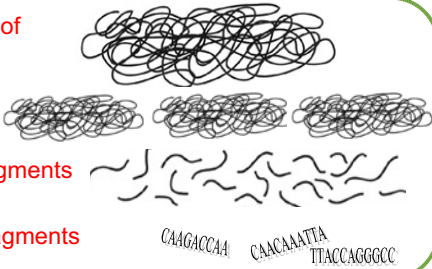
genome sequencing

goals:

- sequence the human genome
- sequence genomes of other organisms
- identify genes, other features of the genome
- understand the function of genes, and mechanisms of the cell

how was the genome sequenced?

- grab a piece of the genome
- make copies
- chop into fragments
- sequence fragments



- record, assemble fragments on a computer, to produce sequence:

CAAGACCAA TTACCGGGCC
CAACAAATTA
CAAGACCAACAAATTACCGGGCC

fragment assembly example

how might you assemble the following fragments?

CTGTAAAT
AATCTG
CCGAAT
ATCTGTA
AATCCG

example (continued)

- one possible arrangement:

CTGTAAAT
AATCCG
CCGAAT
ATCTGTA
AATCTG

- this yields the strand
ATCTGTAAATCCGAATCTG

example (continued)

- another possible arrangement:

CTGTAAAT
AATCTG
AATCCG
ATCTGTA
CCGAAT

- this yields the strand
ATCTGTAAATCTGAATCCGAAT
- this assembled strand is longer, less plausible

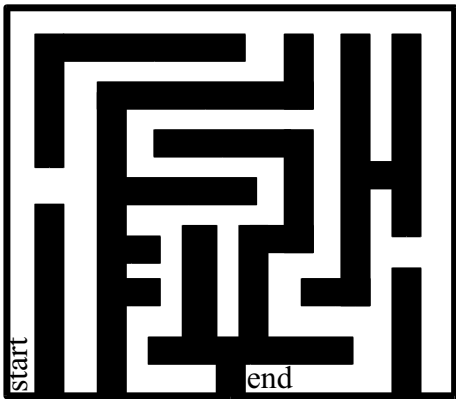
ideas for a fragment assembly algorithm?

- suppose you were given thousands of fragments...what procedure would you use to assemble them?
- one possible guiding principle: find an assembly that results in the *shortest* assembled strand

the fragment assembly problem

given many short DNA fragments, find the shortest DNA strand that includes all of the fragments as substrands

find your way through a maze!



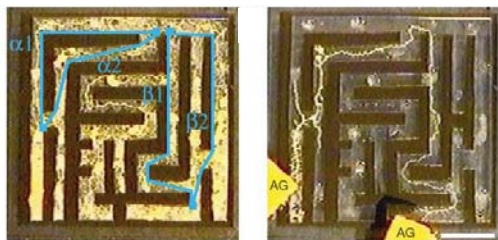
maze solving by the slime mold

- this single-celled organism will put out pseudopodia (tube-like structures) to connect two food sources



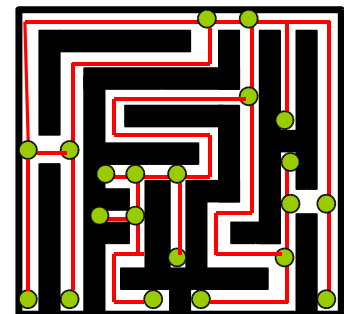
- “This remarkable process of cellular computation implies that cellular materials can show a primitive intelligence”

T. Nakagaki, H. Yamada, A. Toth, Nature, Sep 2000



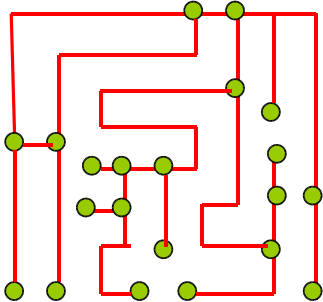
maze search and network algorithms

- searching a maze is reminiscent of crawling the web
- both tasks can be modeled as *network traverse problems*



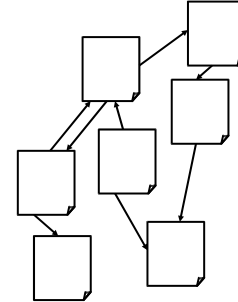
maze search and network algorithms

- searching a maze is reminiscent of crawling the web
- both tasks can be modeled as *network traverse problems*



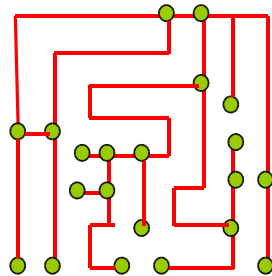
maze search and network algorithms

- searching a maze is reminiscent of crawling the web
- both tasks can be modeled as *network traverse problems*



maze search and network algorithms

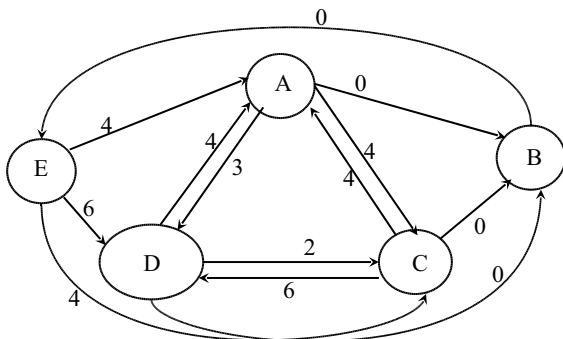
- searching a maze is reminiscent of crawling the web
- both tasks can be modeled as *network traverse problems*
- by systematically traversing all paths, and marking dead-ends as such, the end point of a maze can be found without ever exploring an edge more than once
- algorithms for traversing networks are widely used



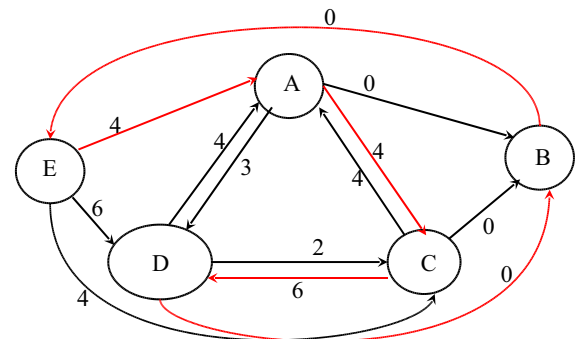
digression: a new network problem

- the *traveling salesman problem (TSP)*: find the cheapest tour in a given network
- edges in the network are labeled with costs
- a *tour* must visit each node *exactly once*, and end at its starting point
- the *cheapest tour* is the tour for which the sum of its costs is smallest

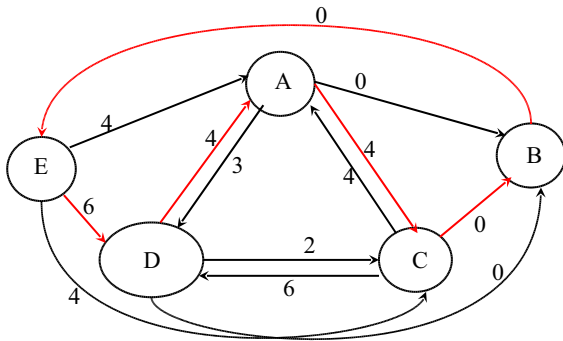
TSP instance



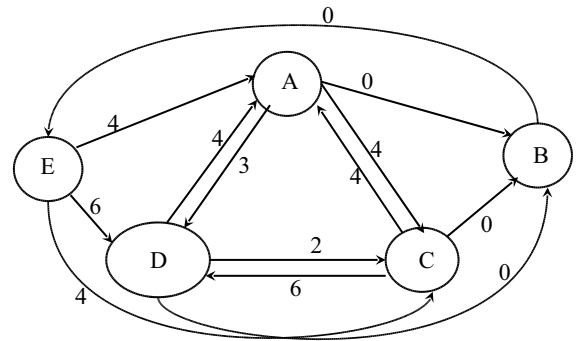
TSP instance: one possible tour, which has cost 14



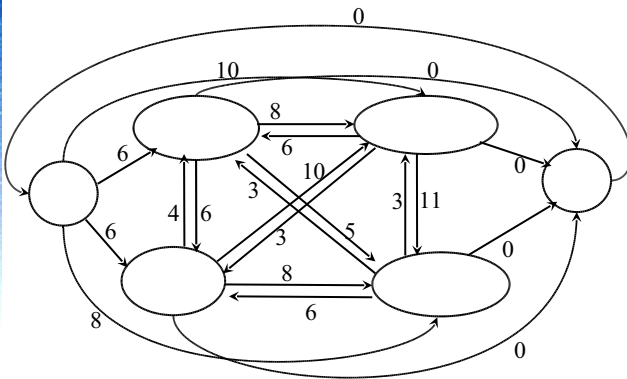
TSP instance: another possible tour, which also has cost 14



TSP instance: which is the cheapest tour?



another TSP instance: which is the cheapest tour?



two network problems: network traversal vs TSP

- graph traverse: visit all nodes reachable from a starting node in a graph
- TSP: find the cheapest tour in a graph whose edges have costs

network traversal vs TSP

- simple, fast network traverse algorithms are widely used on data that is organized as a network
- in contrast, no-one has found an algorithm that will quickly solve the TSP on all networks
- \$1M prize available for anyone who finds a fast algorithm for TSP, or proves that no good algorithm exists!
- still, decades of research have yielded algorithms that work reasonably well on many practical networks

back to fragment assembly

how might you assemble the fragments following fragments?

```
AATCTG
CCGCAA
TGTAATCCG
CTGTAAAT
```

one possible assembly

- the fragments AATCTG, CCGCAA, ATCTGTAAATCCG, CTGTAAAT
- can be arranged as follows:


```
AATCTG   CCGCAA
      TGAAATCCG
      CTGTAAAT
```
- and assembled to produce the sequence


```
AATCTGTAAATCCGCAA
```

 (length 17)

another possible assembly

- an alternative arrangement of the fragments is


```
AATCTG
      CCGCAA
      TGAAATCCG
      CTGTAAAT
```
- which assemble to produce the sequence

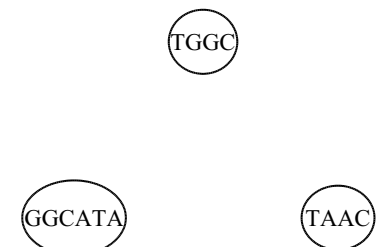

```
CCGCAATCTGTAAATCCG
```

 (length 18!)

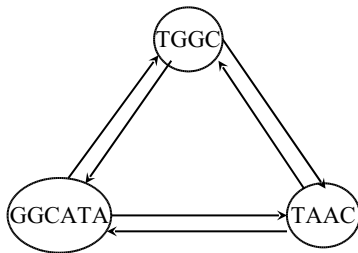
solving the fragment assembly problem

- it turns out that the fragment assembly problem is the TSP in disguise
- let's look at an example, to see the analogy
- three fragments in our example: GGCATA, TGGC, TAAC
- we'll cast the problem of assembling these three fragments as a TSP instance

from fragment assembly to TSP

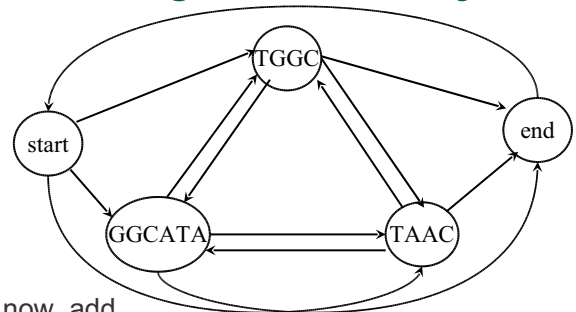
- 
- our TSP graph has one node per fragment

from fragment assembly to TSP



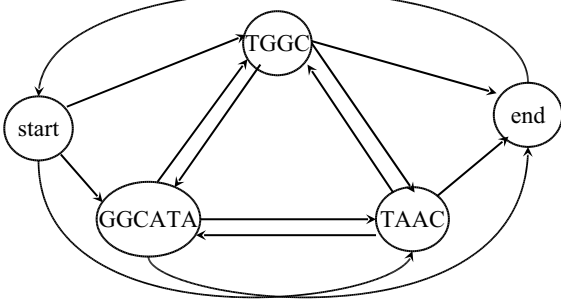
- add a directed edge between each pair of nodes

from fragment assembly to TSP



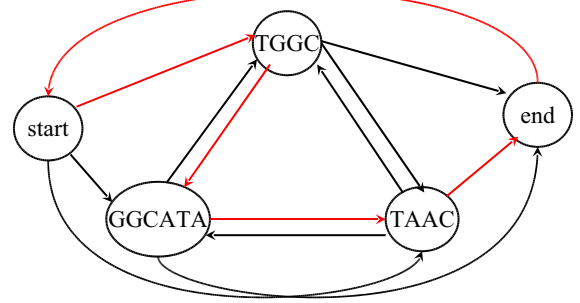
- now, add
 - a *start* node with an edge **to** all fragment nodes, and
 - an *end* node with an edge **from** all fragment nodes
 - an edge from end to start

from fragment assembly to TSP



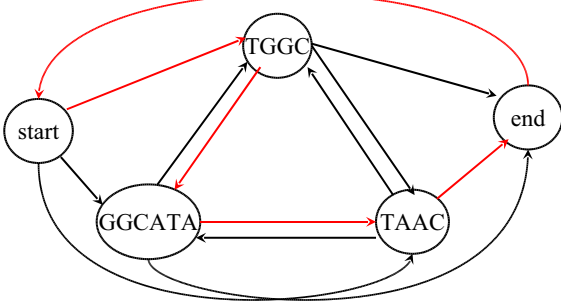
- a tour, starting at *start*, corresponds to an assembly of the fragments

from fragment assembly to TSP



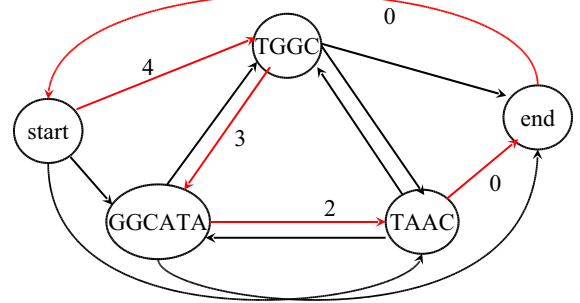
TGGC
GGCATA TGGCATAAC
TAAC

from fragment assembly to TSP



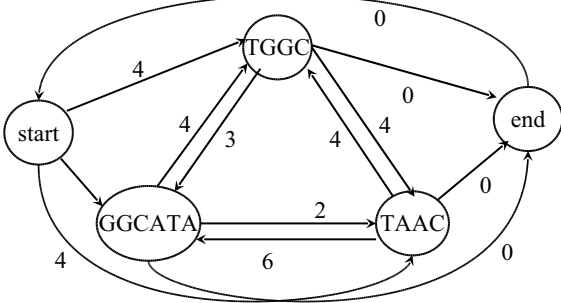
- we want to choose edge costs so that the cost of a tour corresponds to the cost of the corresponding assembled strand

from fragment assembly to TSP



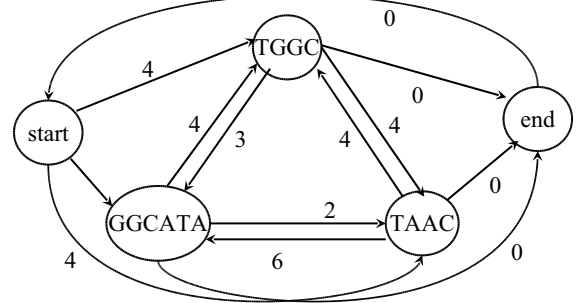
TGGC
GGCATA
TAAC

from fragment assembly to TSP



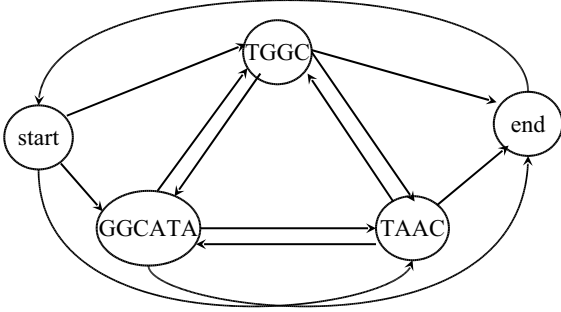
- what about the other costs?

from fragment assembly to TSP



- the cheapest tour in the network corresponds to the shortest assembly of the fragments!

from fragment assembly to TSP



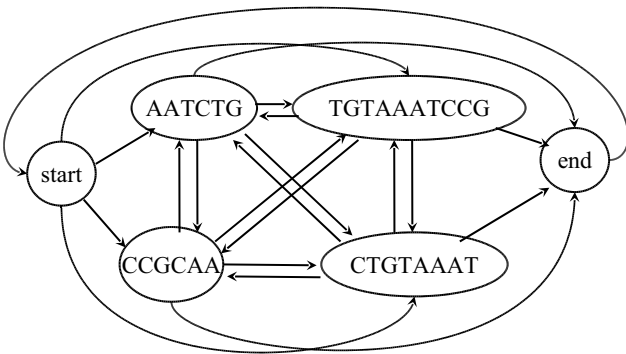
- can you formulate a general rule for assigning costs to edges?

another example

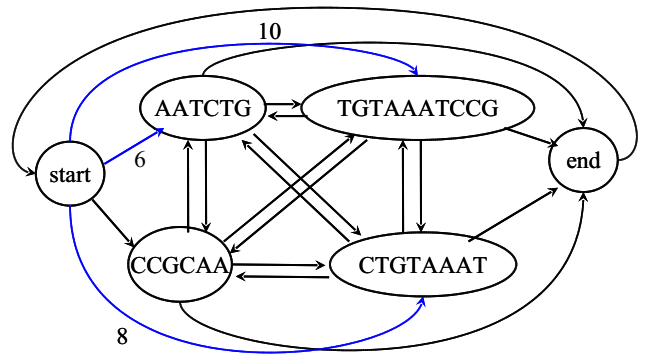
- can you construct the TSP instance corresponding to the following fragment assembly instance?

AATCTG
CCGCAA
TGTAATCCG
CTGTAAAT

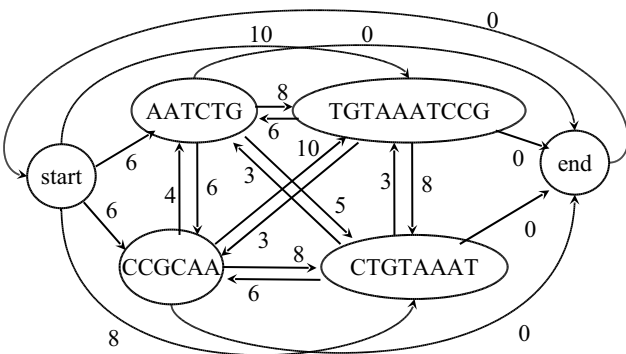
corresponding TSP instance



TSP instance



TSP instance



summary

- fragment assembly computation is an important step in sequencing a genome
- fragment assembly can be solved by computer, using algorithms for a seemingly unrelated network problem - the TSP
- the TSP is an example of the famous so-called 'NP-hard' problems for which no fast algorithms are known



resources

- here is a report on the use of the code for the traveling salesman problem in deriving the mouse genome
 - www.ncbi.nlm.nih.gov/genome/rhmap/
- for more on the \$1M prize for finding a fast solution to the TSP, see
 - www.claymath.org/millennium/P_vs_NP/
 - lecture by Vijaya Ramachandran at www.claymath.org//Popular_Lectures/U_Texas/ explains the problem