Outline

- Graphs, Trees, Search (see Cormen, 5.4–5.5, Chapter 13, Chapters 23–24, 27)
  - Many different interesting algorithms in Bioinformatics use graphs as representations
- Homework: An algorithm for computing the minimum rmsd of two molecules under graph equivalences
What is a graph?

• A directed graph $G$ is a pair $(V, E)$ where $V$ is a finite set and $E$ is a binary relation on $V$

• $V$ is the vertex set

• $E$ is the edge set
  – Set of $(u, v)$ where $u, v$ are in $V$

• Graphs can be directed or undirected

• Degree: number of edges connected to a vertex

• Cycle: path of length greater than 2 which starts and ends on the same vertex
Types of graphs

(a) Directed
(b) Undirected
Types of graphs

\(G\)

\(G'\)

(a)

(b)

Isomorphic

1,2,3,4,5,6 = u,v,w,x,y,z

Non-isomorphic
What is a tree?

- A tree is an **undirected graph** that is **connected**
- A **rooted tree** is one that has a specified special vertex called the **root**
- Trees can be **ordered** or not
Graphs in bioinformatics:
Can represent many things

- Molecules
  - Proteins and DNA: connected, acyclic, directed graphs
  - Organic molecules: connected, possibly cyclic, undirected graphs

Blackboard Examples
Graphs in bioinformatics: Metabolic Pathways (EcoCyc)

*E. coli K-12 Pathway: fatty acid biosynthesis -- initial steps*
Graphs in bioinformatics: Phylogenetic trees
Representing graphs

- Adjacency list
- Adjacency matrix

Undirected graph: Matrix is symmetric
Representing graphs

- Adjacency list
- Adjacency matrix

Directed graph: Matrix is asymmetric
Traversing graphs

• Two basic strategies
  – Breadth first
    • We traverse all of the connected vertices of our current vertex
    • Stop when we run out of vertices
  – Depth first
    • We traverse the first untraversed vertex connected to our current vertex and recursively down
    • Stop when we run out of vertices
Breadth first search

• We start with all vertices initialized:
  – White
  – Depth infinite
  – Predecessor NULL

• We queue up vertices before traversing them

---

for each vertex $u$

  initialize values
  
  $d \leftarrow \infty$
  
  $\text{color} \leftarrow \text{white}$
  
  $\text{pred} \leftarrow \text{null}$

mark start vertex $s$ $(0, \text{white}, \text{null})$

enqueue $(q,s)$

while $q$ is non-null

  $u \leftarrow \text{head}(q)$

  for each connected $v$

    if white

      mark gray

      $d(v) = d(u) + 1$

      $\text{pred}(v) = u$

      enqueue$(q,v)$

  dequeue$(q)$

  $\text{color}(u) \leftarrow \text{black}$
Breadth first search

Traversal order: s w r t x v u y
Depth first search

- We start with all vertices initialized:
  - White
  - Time 0
  - Predecessor NULL
- We recursively traverse downward, processing the vertices as we go

```plaintext
dfs(g)
time ← 0
for each vertex u
  initialize values
    d ← 0
    color ← white
    pred ← null
for each vertex u
  if (color(u) is white) dfs-visit(u)

dfs-visit(u)
color(u) ← gray
time ← time+1
d(u) ← time
for each v adjacent to u
  if white
    pred(v) = u
dfs-visit(v)
color(u) ← black
time ← time+1
finish(u) ← time
```
Depth first search

Traversal order: u v y x (back up) w z (back up)
BFS and DFS form the basis of other algorithms

- Finding a cycle:
  - Do a depth first search
  - If, as we are traversing, we encounter a vertex that we have already marked gray, we have a cycle

- Are two atoms (A,B) part of a ring system?
  - Break their bond (edge)
  - Perform DFS from atom A
  - If we encounter atom B, they are part of a ring system
Other operations on graphs

• Minimum spanning trees
  – Can be applied to clustering
  – Interesting applications in many fields
    (electronic circuit design, molecular diversity)

• Flow
  – Obvious applications in metabolic network analysis
Minimum spanning trees

• You have a weighted, connected, undirected graph $G$

• You must find the tree $T$ such that
  – $T$ is a subgraph of $G$
  – $T$ spans all vertices of $G$
  – The total edge weight of $T$ is a minimum
Minimum spanning tree
We will use a **greedy algorithm**

• We will grow a tree while maintaining the invariant that the tree *must be* a minimum spanning tree
• We start with any vertex, since all vertices must eventually be part of the tree
• We add vertices cleverly (using *safe* edges), to make sure we end up with a tree and that the tree is minimal
• The proof is by induction (see pages 500–502 in Algorithms)
Prim’s algorithm: Pictorially

The key to an efficient implementation is a clever method for computing the next guy to add
Maximum Flow

• A flow network is a directed graph with capacities on the edges
• We define a source and a sink
• A flow is subject to constraints
  – Capacity
  – Conservation
  – Symmetry
Maximum Flow

(a)

(b)
Graphs, trees, and molecules

• Many interesting scientific problems in computational chemistry can be addressed using graph and tree algorithms
  – Molecular diversity
    • We want to pick a small number of molecules from a large collection, where the small set is “diverse”
  – Evaluating the quality of molecular docking
    • We need to compute how good a molecular docking is, but internal symmetries in small molecules makes this nontrivial
Biologically Relevant Chemical Diversity

- Diversity increases leverage
  - Smaller number of compounds synthesized
  - Greater number of hits
  - Broader SAR

- Diversity measure must be sensitive to molecular properties that relate to specific binding events
  - Maximize the likelihood of each molecule probing a different protein binding pocket

- Critical features of distance measure between molecules
  - Small distance -> high probability of binding to the same pocket
  - Large distance -> low probability of binding to the same pocket
  - Must be very fast to compute
We can compute similarities quickly using vectorial approximations.

Molecular hashkeys measure surface properties of molecules by seeing “who they look like.”

Basis molecules

<table>
<thead>
<tr>
<th>Key1</th>
<th>Key2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.54</td>
<td>0.65</td>
</tr>
<tr>
<td>0.52</td>
<td>0.78</td>
</tr>
<tr>
<td>0.28</td>
<td>0.48</td>
</tr>
<tr>
<td>0.30</td>
<td>0.47</td>
</tr>
<tr>
<td>0.57</td>
<td>0.50</td>
</tr>
</tbody>
</table>

Molecules with similar keys
Diversity Analysis of Antibacterials

- 450 antibacterials in the CMC
- Small number of protein targets and chemical classes
  - Cephalosporins: 74
  - “Mycins”: 74
  - Penicillins: 62
  - Sulfa drugs: 49
  - Quinolones: 27
  - Nitrofurantoin + analogs: 23
  - Tetracyclines: 21
  - Miscellaneous (dermatologicals etc...)
- We can automatically select a small diverse subset that hits all classes
Diverse set of 15 covers all classes

Chosen by maximizing diversity of 450 molecular hashkeys
Docking accuracy

- We have a ligand of a protein and dock it into the protein
- We have determined the crystal structure of the protein ligand complex
- We can define the accuracy of the docking as the rmsd of the heavy atoms (non-hydrogens)
- \( \text{Rmsd} = \sqrt{\text{sum of squared deviations}} \)
A molecule with symmetries may be correctly docked but have high nominal rmsd.
rmsd = 1.5 but should be 0.5
Sybyl mol2 file format

@<TRIPOS>MOLECULE
ran-00-ligand
 18 18 0 0 0
SMALL
NO_CHARGES

@<TRIPOS>ATOM
 1 C    -1.221  -4.911  -4.953  C
 2 C    -1.784  -4.887  -6.202  C
 3 C    -2.990  -5.559  -6.437  C
 4 C    -3.629  -6.253  -5.397  C
 5 C    -3.047  -6.261  -4.127  C
 6 C    -1.846  -5.587  -3.920  C
 7 C     0.084  -4.195  -4.703  C
 8 N     0.366  -3.685  -3.541  N
 9 N     1.022  -4.226  -5.603  N
10 H    -1.297  -4.349  -7.006  H
11 H    -3.432  -5.543  -7.426  H
12 H    -4.561  -6.776  -5.577  H
13 H    -3.526  -6.788  -3.309  H
14 H    -1.394  -5.591  -2.935  H
15 H     1.263  -3.201  -3.395  H
16 H    -0.309  -3.764  -2.767  H
17 H     0.831  -4.624  -6.533  H
18 H     1.957  -3.852  -5.385  H

@<TRIPOS>BOND
  1    1    7   1
  2    1    6   ar
  3    1    2   ar
  4    2    3   ar
  5    3    4   ar
  6    4    5   ar
  7    5    6   ar
  8    7    9   2
  9    7    8   1
 10   2   10   1
 11   3   11   1
 12   4   12   1
 13   5   13   1
 14   6   14   1
 15   8   15   1
 16   8   16   1
 17   9   17   1
 18   9   18   1
Homework 2: Due April 13th

- Write a program that will compute the minimum rmsd between two molecules over all isomorphic projections
- Input: a list of pathnames to pairs of molecule files
- Output
  - Actual rmsd (atom number equivalence)
  - Min rmsd under isomorphism
  - Correspondence of atoms that gave rise to the min rmsd
- You should not check bond order equivalence, since it will cause trouble
- Instead, check atom equivalence
  - Atoms A and B are the same element
  - They have the same number of substituents
  - Their substituents are the same elements
- Only worry about the following elements: C N O S P F Cl Br I

What to turn in
- A listing of your program
- The output of your program on Pathlist (sensibly formatted)
- Brief discussion of the complexity of your algorithm
- Email 1 file to ajain@cc.ucsf.edu