# 02-512 Assignment 01

Karan Sikka
ksikka@cmu.edu

September 18, 2014

## 1

**(a)** Cast problem as a bipartite graph where set A is the binding sites and B is the chemical groups. Create edges such that edge between binding site and chemical group is the docking energy. Now the problem is the Minimum Weight B-Perfect Matching (all vertices in B should be in the matching)

1b. Create a graph where the vertices are the drugs and an edge connects drugs which may react with each other. Now solve the Minimum Vertex Coloring problem.

1c.This is wrong - Longest common substring to create alignment. If aligns well, combine the two strings into one. Repeat until all strings are one string.

1d. Use a Markov Chain Monte Carlo algorithm.

1e. Longest common substring

## 2

**(a)** Assume the input is a complete graph where the vertices are the labeled pixels and each edge has a weight equal to the euclidean distance between its vertices. The output is a tree (undirected graph without cycles) covering all the vertices. Also known as a spanning tree. The objective function is the sum of the edge weights of the tree, which we seek to minimize.

**(b)** It's a minimum spanning tree problem. You can use Prim's algorithm.

2c. 2d. 2e.

## 3

**(a)** Image attached Adjacency matrix:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | X | 0 | 0 | -4 |
| 2 | -5 | X | -2 | 0 |
| 3 | 0 | -5 | X | 0 |
| 4 | -4 | 0 | 0 | X |

**(b)** The shortest path covering all vertices is <3, 2, 1, 4 >. Performing the alignment, we obtain the shortest common superstring:

GTACGTTGTAATGTGCGCTAATG

**(c)**

First we split our sequences into 4-mers and 3-mers

```
1.
AATG, ATGT, TGTG, GTGC, TGCG, GCGC, CGCT
AAT, ATG, TGT, GTG, TGC, GCG, CGC, GCT

2.
CGTT, GTTG, TTGT, TGTA, GTAA, TAAT, AATG, ATGT
CGT, GTT, TTG, TGT, GTA, TAA, AAT, ATG, TGT

3.
GTAC, TACG, ACGT, CGTT, GTTG
GTA, TAC, ACG, CGT, GTT, TTG

4.
CGCT, GCTA, CTAA, TAAT, AATG
CGC, GCT, CTA, TAA, AAT, ATG
```

Now we construct a graph where nodes are k-1 mers and edges represent the presence of k mers

```
Nodes:
AAT, ACG, ATG, CGC, CGT, CTA, GCG, GCT, GTA, GTG, GTT, TAA, TAC, TGC, TGT, TTG

Edges:
AAT -> ATG (AATG)
ACG -> CGT (ACGT)
ATG -> TGT (ATGT)
CGC -> GCT (CGCT)
CGT -> GTT (CGTT)
CTA -> TAA (CTAA)
GCG -> CGC (GCGC)
GCT -> CTA (GCTA)
GTA -> TAA (GTAA)
GTA -> TAC (GTAC)
GTG -> TGC (GTGC)
GTT -> TTG (GTTG)
TAA -> AAT (TAAT)
TAC -> ACG (TACG)
TGC -> GCG (TGCG)
TGT -> GTA (TGTA)
TGT -> GTG (TGTG)
TTG -> TGT (TTGT)
```

No unique solution because a k-mer is repeated in more than one sequence, so there will be multiple possible ways to put the sequence back together. Repeated k-mers: ATGT, CGTT, GTTG, CGCT, TAAT, AATG

**(d)**

```
First we split our sequences into 5-mers and 4-mers
1.
AATGT, ATGTG, TGTGC, GTGCG, TGCGC, GCGCT
AATG, ATGT, TGTG, GTGC, TGCG, GCGC, CGCT

2.
CGTTG, GTTGT, TTGTA, TGTAA, GTAAT, TAATG, AATGT
CGTT, GTTG, TTGT, TGTA, GTAA, TAAT, AATG, ATGT

3.
GTACG, TACGT, ACGTT, CGTTG
GTAC, TACG, ACGT, CGTT, GTTG

4.
CGCTA, GCTAA, CTAAT, TAATG
CGCT, GCTA, CTAA, TAAT, AATG

Nodes:
AATG, ACGT, ATGT, CGCT, CGTT, CTAA, GCGC, GCTA, GTAA, GTAC, GTGC, GTTG, TAAT, TACG, TGCG, TG

Edges:
AATG -> ATGT (AATGT)
ACGT -> CGTT (ACGTT)
ATGT -> TGTG (ATGTG)
CGCT -> GCTA (CGCTA)
CGTT -> GTTG (CGTTG)
CTAA -> TAAT (CTAAT)
GCGC -> CGCT (GCGCT)
GCTA -> CTAA (GCTAA)
GTAA -> TAAT (GTAAT)
GTAC -> TACG (GTACG)
GTGC -> TGCG (GTGCG)
GTTG -> TTGT (GTTGT)
TAAT -> AATG (TAATG)
TACG -> ACGT (TACGT)
TGCG -> GCGC (TGCGC)
TGTA -> GTAA (TGTAA)
TGTG -> GTGC (TGTGC)
TTGT -> TGTA (TTGTA)
```

No unique solution because a k-mer is repeated in more than one sequence, so there will be multiple possible ways to put the sequence back together. AATGT, CGTTG, TAATG

### (e)

```
GTAC -> TACG (GTACG)
TACG -> ACGT (TACGT)
ACGT -> CGTT (ACGTT)
CGTT -> GTTG (CGTTG)
```

```
GTTG -> TTGT (GTTGT)
TTGT -> TGTA (TTGTA)
TGTA -> GTAA (TGTAA)
GTAA -> TAAT (GTAAT)
TAAT -> AATG (TAATG)
AATG -> ATGT (AATGT)
ATGT -> TGTG (ATGTG)
TGTG -> GTGC (TGTGC)
GTGC -> TGCG (GTGCG)
TGCG -> GCGC (TGCGC)
GCGC -> CGCT (GCGCT)
CGCT -> GCTA (CGCTA)
GCTA -> CTAA (GCTAA)
CTAA -> TAAT (CTAAT)
```

Shortest string consistent with graph: GTACGTTGTAATGTGCGCTAAT

No it's not the same as the TSP solution. It's missing one character at the end. This disparity is because we lost some information when breaking the sequence apart into 5-mers, increasing the ambiguity when reassembling the pieces. The result of this method was a shorter string, which agreed with the 5-mer data but not completely with the original 4 longer sequences.

## 4

**(a)** Input is a set of drugs, and a set of pairs of drugs which are toxic when taken together. Output is a list of drugs which are not toxic when taken together. Objective function is the length of the output list, which we will maximize.

**(b)** This is similar to the Maximum Independent Set problem, if the drugs are vertices and there's an edge between two drugs which interact toxically.

**(c)**

```
function getMaxIndSet(drugs, interact)
    max_set = {}
    if size(drugs) == 0:
        return {}
    For every drug d1 in drugs:
        ind_set_with_d1    = {d1} + getMaxIndSet(drugs - {d2 for d2 in drugs if (d1, d2) int
        ind_set_without_d1 = getMaxIndSet(drugs - {d1})

        if size(ind_set_with_d1) > size(max_set):
            max_set = ind_set_with_d1

        if size(ind_set_without_d1) > size(max_set):
            max_set = ind_set_without_d1

    return max_set
```