AM 221: Advanced Optimization Spring 2016

Prof. Yaron Singer *Problem Set 9 — Due Wed, Apr. 6th at 23:59*

Instructions: All your solutions should be prepared in LaTeX and the PDF and .tex should be submitted to canvas. For each question, the best and correct answers will be selected as sample solutions for the entire class to enjoy. If you prefer that we do not use your solutions, please indicate this clearly on the first page of your assignment.

The programming parts can be written in the programming language of your choice and the code should be submitted alongside your solutions.

1. Vertex Cover is NP-complete In this problem we will show that the Vertex Cover problem is NP-complete by reduction from the Clique problem.

Definition. *Consider an undirected graph $G = (V, E)$. A set $C \subseteq V$ is called a* vertex cover *of G iff for all $\{u, v\} \in E$, $u \in C$ or $v \in C$ (or both).*

In the decision version of the Vertex Cover problem, the input is an undirected graph $G = (V, E)$ and an integer k and the goal is to decide whether G has a vertex cover of size k.

Definition. *Given an undirected graph $G = (V, E)$, a set $C \subseteq V$ is called a* clique *iff:*

$$\forall \{u, v\} \in C \times C,\ (u, v) \in E$$

In the Clique problem, the input is a graph $G = (V, E)$ and an integer k and the goal is to decide whether G has a clique of size k. The Clique problem is known to be NP-complete.

a. Explain why Vertex Cover is in NP.

b. Given a graph $G = (V, E)$, we define its *complement* $\bar{G} \overset{\text{def}}{=} (V, 2^V \setminus E)$. In other words, $\bar{G}$ has the same set of vertices, and $\{u, v\}$ is an edge in $\bar{G}$ if and only if it is not an edge in G.

   Let $C \subseteq V$ be a set of vertices. Show that C is a vertex cover of G if and only if $V \setminus C$ is a clique of $\bar{G}$.

c. Show that Clique is polynomial-time reducible to Vertex Cover and conclude that Vertex Cover is NP-complete.

2. A 1/2-approximation Algorithm for Vertex Cover. In Section 3, we saw the Vertex Cover problem and formulated it as an integer program. We then studied its LP relaxation and the dual of this relaxation and saw that it could be interpreted as the relaxation of the Maximum Matching problem. In this problem, we will draw inspiration from the Vertex Cover/Matching duality to construct a 2-approximation algorithm for Vertex Cover.

The optimization version of Vertex Cover problem is to find a vertex cover S of minimum size. Let us denote by $\texttt{OPT}_G$ the optimal value of the Vertex Cover problem for G.

Definition. *A set $M \subset E$ is called a* matching *of G if and only if:*

$$\forall u \in V,\ |\{e \in M \mid u \in e\}| \leq 1$$

In other words, a vertex is incident to at most one edge of M. A matching M is said to be maximal *if there is no matching M' with $M \subset M'$.*

a. Give an algorithm to construct a maximal matching of G. What is the running time of this algorithm?

b. Let M be a matching of G, show that $|M| \leq \texttt{OPT}_G$.

c. Let M be a matching and let us denote by $V(M)$ the set of endpoints of edges in M:

$$V(M) \overset{\text{def}}{=} \{u \mid \exists e \in M,\ u \in e\}$$

Show that when M is a maximal matching of G, $V(M)$ is a vertex cover of G.

d. Let M be a maximal matching of G, show that $|V(M)| \leq 2\texttt{OPT}_G$.

3. DNA sequence alignment: algorithms. In this problem we study the problem of DNA sequence alignment. The input to this problem is a pair of DNA sequences:

```
1: TTGATCAATGG
2: ATCATACAAGGA
```

and the goal is to understand whether or not they have the same function. For this we find the best way to align the sequences to each other. If after alignment, the sequences look "similar enough", we will conclude that they have the same function.

More precisely, the sequences are aligned by introducing gaps. For the above two sequences, a possible way to introduce gaps is as follows (gaps are represented by hyphens):

```
1: TTGAT-CAATGG-
2: ATCATACAA-GGA
```

After introducing the gaps, the sequences are compared by counting the number of mismatches, *i.e.* the number of locations where the nucleotides differ. For the above example, there are two mismatches: one at location 0, and one at location 2 (using the usual array indexing convention in programming).

Let g denote the number of gaps and m denote the number of mismatches in a given alignment, the overall cost c of the alignment is then defined by:

$$c \overset{\text{def}}{=} 2 \cdot g + m$$

The above alignment has cost $2 \cdot 3 + 2 = 8$. Another possible alignment is:

```
1: TTGAT-CAATGG
2: ATCATACAAGGA
```

which has cost $2 \cdot 1 + 4 = 6$, which is minimal among all possible alignments of these two sequences.

Let us denote by s (resp. t) the first (resp. second) sequence. We define $c(s,t)$ to be the cost of the alignment of minimal cost among all possible alignments. Finally, we denote by n (resp. m) the length of s (resp. t).

a. For a sequence s, $s[i:j]$ will denote the substring of s ranging from index i inclusive to j exclusive. Give a formula to compute $c(s[0:i], t[0:j])$ as a function of $c(s[0:i], t[0:j-1])$, $c(s[0:i-1], t[0:j])$ and $c(s[0:i-1], t[0:j-1])$.

b. Using part a. design and describe an algorithm to compute $c(s,t)$ [**hint:** think dynamically!]. What is the running time complexity of this algorithm?

4. DNA sequence alignment: implementation. You will now implement the algorithm you designed in Problem 3. The dataset is available at `http://thibaut.horel.org/dna.txt`. Each line of the datafile is the list of the first 10,000 base pairs of the genome of the well-known *Escherichia coli* bacteria. There are only two lines corresponding to two different species of the bacteria.

a. Write a script which reads the datafile and outputs the cost of the optimal alignment of the two DNA sequences. Submit the code you wrote.

b. Two DNA sequences are considered to have the same function is the cost of the optimal alignment divided by the length of the sequence is smaller than 5%. Do the two DNA sequences in the datafile play the same function?