1 Comments for Improvements on the Chapter

1.1 Regarding Theorem 8.1.1

When going through the proof, I understood it but I believe it can be slightly modified without becoming more complicated, to be more easily understood by the reader. Specifically, I think that points 3 and 4. of the proof are not very clear. For example, point 3. might never be satisfied, since there might be a VC of size \(< k\). I believe that point 3. should be:

"3. Keep doing this until either the tree is of height \(k\) or there are no edges left in the set \(G - R\), where \(R \subseteq G\) is the set of vertices removed by this path of the algorithm's tree so far."

Similarly, point 4. should be:

"If one of the leaves' graph \(G - R\) contains no edges, then \(R\) is a vertex cover of size \(\leq k\). If not, then there is not."

1.2 Chapter Bugs/Improvements

Since some improvements I suggest can be also considered bugs, I added this section, where I explain them.

1. On page 213, in the proof of Theorem 8.2.1, on step 3, it should be "If there is a vertex \(v\) of degree at least \(L + 1\ldots". The algorithm does not work properly with the exact value.

2. On page 214, Theorem 8.2.1 should be denoted Theorem 8.2.4.
### 2 Improving Figure 9.1

Using Tikz, I improved Figure 9.1, which is also now easy to modify further in case the authors later want to use it in the book with some changed parameters/notation within the Figure.

<table>
<thead>
<tr>
<th>$S_{4i-3,4j-3}^I$ : $(iN - z, jN + z)$</th>
<th>$S_{4i-3,4j-2}^J$ : $(iN + z, jN + z)$</th>
<th>$S_{4i-3,4j-1}^I$ : $(iN - \alpha, jN + z)$</th>
<th>$S_{4i-3,4j}^J$ : $(iN + z, jN + z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{4i-2,4j-3}^I$ : $(iN - z, jN + b)$</td>
<td>$S_{4i-2,4j-2}^J$ : $((i + 1)N, (j + 1)N)$</td>
<td>$S_{4i-2,4j-1}^J$ : $(iN, (j + 1)N)$</td>
<td>$S_{4i-2,4j}^J$ : $(iN + z, (j + 1)N + b)$</td>
</tr>
<tr>
<td>$S_{4i-1,4j-3}^I$ : $(iN - z, jN - b)$</td>
<td>$S_{4i-1,4j-2}^J$ : $((i + 1)N, jN)$</td>
<td>$S_{4i-1,4j-1}^I$ : $(iN, jN)$</td>
<td>$S_{4i-1,4j}^J$ : $(iN + z, (j + 1)N - b)$</td>
</tr>
<tr>
<td>$S_{4i,4j-3}^I$ : $(iN - z, jN - z)$</td>
<td>$S_{4i,4j-2}^J$ : $((i + 1)N + \alpha, jN - z)$</td>
<td>$S_{4i,4j-1}^J$ : $(iN + 1N - \alpha, jN - z)$</td>
<td>$S_{4i,4j}^J$ : $(iN + z, jN - z)$</td>
</tr>
</tbody>
</table>
3 Additional Problems/Results

**Bichromatic Closest Pair (BCP):** Given two sets \( A \) and \( B \), of points in some space, find \( a \in A \) and \( b \in B \) such that \( \|a - b\| \) is as small as possible (assume \( l_1 \) norm).

Suppose \(|A| = |B| = n|\). The result of [2] gives a lower bound on the time complexity of BCP assuming SETH:

**THEOREM**

Assume SETH. Then for all \( \epsilon > 0 \), solving BCP requires \( \Omega(n^{2-\epsilon}) \) time.

**END THEOREM**

**Offline Nearest Neighbor (OffNN):** Given a set of points \( A \) in some space and a set of query points \( B \), for each query point \( b \in B \) find the point \( a \in A \) that is closest to \( b \) and the distance between \( a \) and \( b \).

**LEMMA**

Assume SETH. Then for all \( \epsilon > 0 \), solving OffNN requires \( \Omega(n^{2-\epsilon}) \) time.

**END LEMMA**

Derived directly from Theorem 1.

**Online Nearest Neighbor (OnNN):** Given a set of points \( A \) in some space, preprocess \( A \). Then, for each incoming query point \( b \) from a set of query points \( B \) that is provided online, find the point \( a \in A \) that is closest to \( b \) and the distance between \( a \) and \( b \).

Suppose \(|A| = |B| = n|\). Then [3] gives the following hardness result for OnNN assuming SETH:

**THEOREM**

Assume SETH. Let \( \delta, \epsilon > 0 \). Assume algorithm \( \text{Alg} \) that is allowed \( O(n^\epsilon) \) preprocessing time for input set \( A \). \( \text{Alg} \) requires \( \Omega(n^{1-\delta}) \) time to answer each online NN query \( b \in B \).

**END THEOREM**

**Dominating Set (DOM):** The following result for DOM found in [5] uses a different reduction to the one mentioned in the book:

**THEOREM**

Assuming the ETH, there is some \( \delta > 0 \) such that \( q \)-Dominating Set has no \( O(n^{\delta q}) \)-time algorithms for all sufficiently large \( q \).

**END THEOREM**

The proof uses a very interesting reduction from \( k \)-SAT to \( q \)-DOM.

Also, similar to Theorem 9.4.2 in the book, we can have the following result for DOM with respect to SETH, the proof of which is in [5].

**THEOREM**

Let \( q \geq 3 \) and \( \epsilon > 0 \). There is no \( q \)-Dominating Set algorithm running in time \( O(nq - \epsilon) \) unless SETH fails.

**END THEOREM**

Finally, another interesting exercise for this chapter could be to show that assuming ETH, Subset Sum has no \( 2^{o(n)} \) time algorithm.
References

[1] CS 354 Stanford Lecture Notes:


