COMP3121
Assignment 3
A17S1N3

Evangelos Kohilas
z5114986

By submitting this document you are confirming that all the answers are your work and are not take from any other sources unless clearly mentioned.

Question 1

We can find the minimal tree by recursing down the tree and balancing a subtree by increasing the lengths on each side such that the path sums on both sides are equal.

When we reach the final lengths, we set the smaller length to be equal to the larger length.

By doing this, as all the base lengths will first become equal, all the other lengths will be balanced to their possible minimum.

```
def recurse(node):
    if node.left is leaf and node.right is leaf:
        new_length = max(node.left_length, node.right_length)
        node.left_length = new_length
        node.right_length = new_length
        return new_length
    else:
        right_sum = recurse(node.right)
        left_sum = recurse(node.left)
        max_sum_path = max(right_sum + node.right_length, left_sum + node.left_length)
        node.left_length = max_sum_path - left_sum
        node.right_length = max_sum_path - right_sum
        return max_sum_path
```

Question 2

The objects should be machined in descending order of polishing time, and then ascending order of machining time.

Proof of optimality:
let A be our greedy solution produced by our algorithm and O be the optimal solution.
let (m, p) be the machining and polishing time of some item and (m', p') be the machining and polishing time of another item.

In the case where $p' < p$ and $m' = m$,
we let the time taken for two items done in order a and b be $a_m + b_m + \max(a_p - b_m, b_p)$

If A selects (m, p) to be machined first, we let $total_A = m + m' + \max(p - m', p')$

In the first case where $p > m$:

$$\begin{aligned} total_A^1 &= m + m' + p - m' \\ &= m + p \end{aligned}$$

Otherwise:

$$\begin{aligned} total_A^2 &= m + m' + p' \\ &= 2m + p' \end{aligned}$$

If we violate our greedy choice and O selects (m', p') to be machined first,
we let $total_O = m' + m + \max(p' - m, p)$

In the first case where $p > m$:

$$\begin{aligned} total_O^1 &= m' + m + p \\ &= 2m + p \end{aligned}$$

Otherwise:

$$\begin{aligned} total_O^2 &= m' + m + p' - m \\ &= m + p' \end{aligned}$$

Then we see that $total_A^1 < total_O^1$ and $total_A^2 < total_O^2$.
Therefore the optimal solution O is worse than the greedy solution A.

In the case where $p = p'$ and $m' < m$,
we let the time taken for two activities done in order a and b be $a_m + \max(a_p, b_m + b_p)$

Using A, we select (m, p) to be machined first, so we get:

$$\begin{aligned} total &= m' + \max(p', m + p) \\ &= m' + \max(p', m + p') \\ &= m' + m + p' \end{aligned}$$

If we violate our greedy choice and O selects (m', p') to be machined first, we get:

$$\begin{aligned} total' &= m + \max(p, m' + p') \\ &= m + \max(p, m' + p) \\ &= m + m' + p \end{aligned}$$

Then we see $total = total'$ showing no difference between O and A.

Since a change in machining time sees no difference, then there will be no difference in every other case.

Therefore, A is optimal as we cannot create a better solution than the greedy one produced by our algorithm.

Question 3

If Alice makes a graph connecting people that know each other, Alice can continuously remove all people of degree less than 5 until either everyone is removed (in which case the constraints cannot hold for any amount of invitees) or until no more people can be removed (in which case the largest number of invitees have been found).

Proof of optimality:
let $S = \{p_1, p_2, p_3, \ldots, p_n\}$
let O be the set of the optimal solution.
let S_i be the subset of S after i iterations where $0 \leq i \leq n$.

We prove by induction that O will always be a subset of S_i.

When $i = 0$, no people have been removed, and so O remains a subset of S_0

Assume $i = k$ such that after k iterations, O is a subset of S_k

let $i = k + 1$:
O will still remain as a subset of S_{k+1} as p_{k+1} will only be removed if p_{k+1} doesn't satisfy the two constraints.

Therefore by induction, since O is still a subset of S, S is optimal since $|S| \geq |O|$, as we are trying to maximise the number of invitees.

Question 4

a) Given the following arrangement,

By using the closest pair method (the proposed algorithm) our total length will be $5 + 1 + 1 = 7$

This is not optimal as the following solution has a shorter length of size $2 + 2 + 1 = 5$

b) Going from left to right, we can produce an optimal solution by matching each dot with its earliest possible match.

Proof of optimality:
Let A be our greedy solution produced by our algorithm and O be the optimal solution.

If we violate our greedy algorithm, then there are two cases that can occur.

If we swap two pairs in A such that a pair finishes in the position of an earlier pair, then the earlier pair will finish in the position of the later pair, but the total length will not increase as both will have changed by the same amount.

If we swap two pairs in A such that a pair finishes in the position of a later pair, then the length of that pair will increase, but the length of the later pair will either decrease by an equal amount such that the total length will not increase, or not decrease, such that the total length has increased.

Therefore A is optimal as O does not offer a better solution than the one produced by A.

Question 5

To complete n tasks with as low total penalty as possible, we pick the highest penalty task and schedule it to be completed as late as possible before incurring a penalty. If the picked task is scheduled to be completed in such a way that would incur a penalty, we ignore it, scheduling it to be completed after all tasks without penalty have been completed. We repeat this until all tasks have been scheduled.

Proof of optimality:
Let A be our greedy solution produced by our algorithm and O be the optimal solution.

We can violate our greedy algorithm by swapping tasks from A to O, such that they are either scheduled earlier, or scheduled later.

In both cases, by doing so we have either:

1. Scheduled the tasks such that there is no change to the total penalty.
2. Scheduled one of the tasks to be past its due date, and thus increasing our penalty.
3. Scheduled a lower cost task to be before its due date, but scheduled a higher cost task after its due date, thus increasing out total penalty.

Therefore A is optimal as all cases cannot create a better solution than the greedy one produced by our algorithm.

Question 6

To calculate the first 10 books that we need to borrow, we iterate through the sequence of books, adding each book to a set and continuing until our set is of length 10. Whenever we come across a book we need that is not in our set, we make another trip to the library, creating a new set and iterating as before.

Proof of optimality:
let B be the sequence of books.
let A be the set of the first 10 unique books from B to keep produced by our algorithm.

Suppose A is not optimal, and assume O be the optimal solution.
If we violate the greedy algorithm, taking one less book, then $|O| = 9$.

If $|O| = 9$ then one of two things have occurred:

Either we have moved a book to the next set such as the next set will contain the 20^{th} book that we will need, as it will be the same as the book removed.

Or we have removed a book such that we require an additional trip to compensate for the 20^{th} book that could not be added to our set.

Therefore, A is optimal as we cannot create a better solution than the greedy one produced by our algorithm.