IN4301 Advanced Algorithm - Homework Assignment 4

Lu Dai (4506677) L.Dai-1@student.tudelft.nl

Let A denote the nodes in X, B denotes the nodes in $\overline{X}$.

1.
Let v denotes the vertex in V_1. Maximum total weight :

$$W_1(A,B) = \begin{cases} \sum_{u,v \in E} w(u,v) & \text{include v in A} \\ 0 & \text{do not include v in A} \end{cases}$$

Consider only V_1, there are only two situations, put v in A or not. when v is put in A, then its maximum total weight is the weight sum of all eadges connected to v. If we put v in B, then A is empty set at this moment with the total weight 0. Thus, the maximun total weight over all partitions is $\sum_{u,v \in E} w(u,v)$.

2.
$V_{i+1} = Vi \cup \{v\}$:

$$OPT_{i+1}(A,B) = \begin{cases} OPT_i(A \backslash \{v\}, B \cup \{v\}) + \sum_{(u,v) \in E, u \notin A} w(u,v) - \sum_{(u,v) \in E, u \in A} w(u,v) & \text{include v in A} \\ OPT_i(A,B) & \text{do not include v in A} \end{cases}$$

As V_{i+1} has one more vertex than V_i, we only need to consider whether to put v in A or put v in B. If we put v in A, then we need to delete weight of edges between (u,v) where u already in A. In addition, we need to add weight of edges (u,v) where u in B. If we put v in B, then the result remains the same as node i, then we need to do nothing.

3.
$V_{i+1} = Vi \backslash \{v\}$:

$$OPT_{i+1}(A,B) = max\{OPT_i(A,B), OPT_i(A \cup \{v\}, B \backslash \{v\})\}$$

In this case, V_{i+1} do not introduce new vertex, thus we only need discuss the already-existed partitions. As V_i do not contain vertex v, the optimal results can be generated by comparing maximum total weight between the exsiting partitions include v and exclude v.

4.
Let $f_t(A,B)$ denotes $OPT(A,B)$. The psuedocode of algorithms is:

```
function MAXWEIGHT
    Root T at node r
    for each node t of T in post-order do
        if t is leaf node then
            for each partition (A,B) of V_t do
                f_t(A, B) = W_t(A, B)
            end for
        else
            if |V_t| > |V_{t-1}| then
                for each partition (A,B) of V_t do
                    f_t(A, B) = W_t(A, B)
                end for
            else
                Let v deneote the vertex in node V_{t-1} but not in node V_t
                for each partition (A,B) of V_t do
                    f_t(A, B) = max{f_{t-1}(A, B), f_{t-1}(A ∪ {v}, B\{v})}
                end for
            end if
        end if
    end for
     return max{f_r(A, B) : each partition (A, B) of V_r}
end function
```

5.

As the width of the tree is k, the biggest node contains k+1 vertices. Thus, there are at most 2^{k+1} different partitions in each node. Also, there are r different tree node.

Therefore, the space complexity is $O(r \cdot 2^{k+1})$.

6.

In a path decomposition, we have r nodes, where there are $\frac{r}{2}$ introduce nodes and $\frac{r}{2}$ forget nodes.

For each introduce node, we need calculate $W_i(A, B)$, which requires $O(n)$ time for checking edges and getting weights. This needs to be done for at most 2^{k+1} different partitions for each node. Therefore, $O(\frac{r}{2}\, n2^{k+1})$ time is need for all introduce nodes in the worst case.

For each forget node, we need to do at most 2^k compararations of partitions pairs with v and without v. Thus, the worst case for dealing with all forget nodes will need time $O(\frac{r}{2}\, 2^k)$.

In conclusion, the time complexity is $O(\frac{r}{2}\, n2^{k+1} + \frac{r}{2}\, 2^k)$, which is simplified as $O(n \cdot r \cdot 2^k)$.