COMP3121-Ass4-Q2

z5302513, Kihwan Baek

August 2021

1. Solution

Firstly, we check if there are some shops which are not connected to any road. If at least one shop is not connected to a route, we can say it is impossible to supply all shops.

If all the routes to all the stores exist, we use merge sort for all the times it takes to go from a warehouse to a shop $(d_1, d_2, ...d_m)$ in ascending order. Then, we can get an array A which include all the times in order and this sort takes O($mlogm$) time.

After that, to use binary search, we set the index of the first time in A as low which is '1' and the index of the last time as $high$ which is m from $d_1, d_2, ...d_m$. And then, we set the average value between low and $high$ as mid and make new array B which includes all roads from the first road to 'mid' th road $(A[0], A[1], ...A[mid_1])$.

Then, we only consider the roads in B and use Edmonds-Karp algorithm to see if they are enough to obtain a matching of warehouses with shops which is of size n and it takes O(nm^2) time.

If there doesn't exist such a matching using roads in B and exists such a matching using roads in $B + A[mid]$, we can say we get the algorithm minimising the time until all shops are supplied. If all shops can be supplied by using B, we update $high$ as $mid - 1$. If we cannot supply goods for all shops using B and $B + A[mid]$, we update low as $mid + 1$. Then, we go back to set the new $'B'$ as including all roads from the first road to $'mid'$ th road and repeat this process again until $low > high$. If we don't get the algorithm in this process, we can say the algorithm minimising the time until all shops are supplied doesn't exist.

2. Time Complexity

The total time complexity includes sorting the times, binary search, Edmonds-Karp algorithm and some condition checks. Therefore, it takes O($mlogm$) + O($logm\times$ (nm^2 + some condition checks)) = O($logm * m^2 * n$) time. At this

point, we know that m cannot be greater than n^2, so we can say this algorithm is polynomial time algorithm (O($logm * m^2 * n$)).