COMP3121-Ass2-Q2

z5302513, Kihwan Baek

July 2021

Firstly, we use merge sort for workers, entry-level job openings and senior job openings in increasing order of skill level in time $O(N \log N)$, $O(P \log P)$ and $O(Q \log Q)$ respectively. After that, we number each of workers from x_1 to x_n. And then, we search for a entry-level job opening in which the first worker (who has the lowest skill level) can participate in from the beginning (which means $x_i <= p_k$ where $1 <= i <= N$ and $1 <= k <= P$). If we find the first job opening for the first worker, put the first worker into the job opening and keep finding the pair (worker, job opening) under the condition ($x_i <= p_k$) until there is no more entry-level job opening in which the current worker cannot participate in (because his skill level is higher than the level of the job opening). This searching takes as long as O(P).

If there are still other workers who couldn't participated in a job opening, check if the skill level of the first worker from the rest is higher or equal to the first senior job opening's requirement. If not, search for a senior job opening in which the worker's skill level is higher or equal to the requirement from the beginning. If we find, we keep going until there is no more senior job opening in which the current worker cannot participate in (because his skill level is less than the level of the job opening). This searching takes as long as O(Q). Finally, all this process is based on greedy algorithm because we keep tracking the best solution at each moment for solving the problem and the total time complexity is $O(N \log N + P \log P + Q \log Q + P + Q) = O(N \log N + P \log P + Q \log Q)$.