Problem 1. Let $P$ be a set of $n$ integer pairs, each of which has the form $(id, key)$. It is guaranteed that no two pairs have the same id (but there may be pairs having the same key). Describe a structure of $O(n)$ space to support each of the following operations in $O(\log n)$ time:

- Insert($i, k$): add a pair $(i, k)$ to $P$ if $P$ does not already have a pair with id $i$;
- DecreaseKey($i, k$): if $P$ does not have any pair with id $i$, this operation has no effects. Otherwise, suppose that the pair is $(i, k')$; the operation replaces the key $k'$ of the pair with $k$ if $k < k'$;
- DeleteMin: Remove from $P$ the pair with the smallest key.

Problem 2. Describe how to implement Dijkstra’s algorithm on a graph $G = (V, E)$ in $O((|V| + |E|) \cdot \log |V|)$ time.

Problem 3. In the lecture we proved the correctness of Dijkstra’s algorithm. Point out the place in the proof that requires the assumption that all the weights are non-negative.

Problem 4 (SSSP with Unit Weights). Let us simplify the SSSP problem by requiring that all the edges in the input directed graph $G = (V, E)$ take the same weight, which we assume to be 1. Give an algorithm that solves the SSSP problem in $O(|V| + |E|)$ time.

(Remark: you can of course still use Dijkstra’s algorithm, but as shown earlier, its complexity is $O((|V| + |E|) \log |V|)$. You mission here is to improve the time complexity to $O(|V| + |E|)$)

Problem 5*. In the lecture, we proved the correctness of Dijkstra’s algorithm in the scenario where all the edges have positive weights. Prove: the algorithm is still correct if we allow edges to take non-negative weights (i.e., zero weights are allowed).