Problem type 1:

You are given a priority queue data structure that performs the operations listed below in the corresponding runtimes. Djikstra's algorithm is listed to the left. What is the asymptotic bound of Djikstra's algorithm if you were to use the implementation of the priority queue data structure below:

```
Initialize for $v \in V$, $\text{dist}(s,v) = d'(s,v) = \infty$
Initialize $X = \emptyset$, $d'(s,s) = 0$
**For** $i = 1$ to $|V|$
    Find $v$ such that $d'(s,v) = \min_{u \in V-X} d'(s,u)$
    $\text{dist}(s,v) = d'(s,v)$
    $X = X \cup \{v\}$
    **For** $u \in V - X$:
        $d'(s,u) = min\{d'(s,u), \text{dist}(s,v) + \ell(v,u)\}$
```

(See variants below)

a. **BYF**

Dumb PQ:

- **makePQ**: $O(1)$
- **findMin**: $O(n)$
- **extractMin**: $O(n)$
- **insert**$(v, k(v))$: $O(1)$
- **delete**(v): $O(1)$
- **decreaseKey**$(v, k'(v))$: $O(1)$
- **meld**: $O(1)$

Solution: This is the same as not having priority queues and simply doing the min operation by brute force. As discussed in lecture, this takes: $O(n^2 + m)$ ■

b. **BYE**

Heap PQ:

- **makePQ**: $O(n)$
- **findMin**: $O(1)$
- **extractMin**: $O(\log(n))$
- **insert**$(v, k(v))$: $O(\log(n))$
- **delete**(v): $O(\log(n))$
- **decreaseKey**$(v, k'(v))$: $O(\log(n))$
- **meld**: $O(\log(n))$

Solution: As discussed in lecture, this takes: $O((n+m)\log(n))$ ■

c. **BYA**

Fibonacci Heap PQ:

- **makePQ**: $O(n)$
- **findMin**: $O(1)$
- **extractMin**: $O(\log(n))$
- **insert**$(v, k(v))$: $O(1)$
- **delete**(v): $O(\log(n))$
- **decreaseKey**$(v, k'(v))$: $O(1)$
- **meld**: $O(1)$

Solution: As discussed in lecture, this takes: $O(n\log(n)+m)$ ■

d. **BYC**

Ideal PQ:

- **makePQ**: $O(n)$
- **findMin**: $O(1)$
- **extractMin**: $O(1)$
- **insert**$(v, k(v))$: $O(1)$
- **delete**(v): $O(1)$
- **decreaseKey**$(v, k'(v))$: $O(1)$
- **meld**: $O(1)$

Solution: In this case, extracting the min takes constant time as does decreasing the vertex distance value. So every edge and vertex is considered once with constant time. This takes: $O(n+m)$ ■

Problem type 2:

Answer the problem:

(See variants below)

P.S. You may **not** use lab solutions as a black box.

a. **BYH**

I have a graph where all the edge lengths are of length 1 or 2. Provide a algorithm (as fast as possible), that finds the shortest path between two vertices s and t.

Solution:
Optimal solution: Make a new graph G where every edge of length 2 is broken into two edges of length one and a node in the middle. Then we can just use $BFS(G', s)$
Suboptimal solution (1 point): Djikstra's/FW/BF/etc. ■

b. **BYG**

Suppose you are given a weighted directed graph $G = (V, E)$ in which edges that leave the source vertex s may have negative edge weights, but all other edge weights are none negative and there are no negative-weigth cycles. Does Djikstra's algorithm still work? Explain why or why not?

Solution:
Yup it works in this case. Djikstra's algorithm works on the assumption that the path weight keeps increasing whihc in this case it does. So everything works out. ■

c. **BYB**

You are given a directed acyclic graph. Describe a algorithm that finds the **shortest** path from a vertex s to a vertex v.

Solution: (Paraphrased from the lab16 solutions)

- Topologically sort the graph
- Set the distance of all vertices to ∞
- Starting from s and evaluating the vertices in topological order:
- for each vertex v adjacent to u: $d(v) = min(d(v), d(u) + ell(u, v))$

■

d. **BYD**

You are given a directed acyclic graph. Describe a algorithm that finds the **longest** path from a vertex s to a vertex v.

Solution: (Paraphrased from the lab16 solutions)

- Topologically sort the graph
- Set the distance of all vertices to $-\infty$
- Starting from s and evaluating the vertices in topological order:
- for each vertex v adjacent to u: $d(v) = max(d(v), d(u) + \ell(u, v))$

■