Exercise 8.2 The following problem (and more elaborate extensions) appear in reinforcement learning. Let $G = (V, E)$ be a directed graph and let the edges be annotated by positive edge weights $r : E \to \mathbb{R}_{>0}$. We think of the vertices as states of some device under our control, and the edge weights $r(e)$ as rewards obtained by traversing the edge e as follows. Let $s \in V$ be a fixed starting vertex / state.

Exercise 8.2.1. Given an integer $k \leq n$, the goal is to compute a walk of length at most k maximizing the sum of rewards along that walk. (If you repeat an edge, you get the same reward each time you repeat the edge.) Design and analyze an algorithm for this problem.

Solution. Our approach is to constrain Bellman-Ford to compute minimum walks with no more than k edges.

```
K-Bellman-Ford($G = (V, E), s \in V, k$):
/* Computes the minimum-length $d(s, v)$ of a $k$-walk from $s$ to $v$ for all $v \in V$. */
1. Set $d(s, v) = +\infty$ for all $v \in V$.
2. Set $d(s, s) = 0$.
3. Do the following $k$ times
    A. For all edges $e = (u, v) \in E$
        1. $d(s, v) \leftarrow \min(d(s, v),\ d(s, u) + \ell(u, v))$
```

Given $G = (V, E)$ with positive edge weights, we negate the edge weights to obtain a graph G^- with only negatively weighted edges. Running K-Bellman-Ford(G^-, s, k) assigns $d(s, t) :=$ SW(s, t, k) for all $t \in V$. Hence, the distance of the maximum k-walk in G can be directly computed by negating the minimum value of SW(s, t, k) in G^-.

Runtime.

$$T(n) = \sum_{i=1}^{k} \sum_{e \in E} 1 \in O(mk)$$

■

Correctness. The correctness of this algorithm is inherited from the correctness proof of Bellman-Ford. If we want to prove this explicitly, it suffices to show that in k iterations, K-Bellman-Ford produces the shortest walks using no more than k edges; in particular, $d(s, t) =$ SW(s, t, k) for all $t \in V$.

We approach this proof by induction on k, the number of iterations.

In the base case, $k = 0$, we have $d(s, t) = +\infty =$ SW($s, t, 0$) for all $t \neq s$, and $d(s, s) = 0 =$ SW($s, s, 0$), as desired.

Now suppose the claim holds for k. Then on the $(k + 1)$-th iteration, for each $t \in V$, we have one of the following cases:

i. SW($s, t, k + 1$) $=$ SW(s, t, k). Then by the induction hypothesis, the algorithm assigns

$$d(s, t) = \text{SW}(s, t, k) = \text{SW}(s, t, k + 1)$$

as desired.

ii. $\underline{\texttt{SW}(s,t,k+1)} \neq \underline{\texttt{SW}(s,t,k)}$. For such t, there must exist some $v \in V$ such that $d(s,v) = \underline{\texttt{SW}(s,v,k)}$ is finite. When $d(s,v) + \ell(v,t)$ is minimal, we get the shortest possible $(k+1)$-walk from s to t through v. Correspondingly, the algorithm assigns

$$\begin{aligned} d(s,t) &= \min_{v \in V:\ (v,t) \in E} d(s,v) + \ell(v,t) \\ &= \min_{v \in V:\ (v,t) \in E} \underline{\texttt{SW}(s,v,k)} + \ell(v,t) \\ &= \underline{\texttt{SW}(s,t,k+1)} \end{aligned}$$

By induction, the claim holds for all k. ■

□

Exercise 8.2.2. Here we also incorporate a *discount rate*. Let $\alpha \in (0,1)$ be given. Given a walk with edges $e_1, \dots, e_k$, the *discounted total reward* of the walk is given by

$$r(e_1) + \alpha r(e_2) + \cdots + \alpha^{k-1} r(e_k).$$

The idea is that if we think of each edge traversal as also taking a unit of time, then the rewards attained far off in the future are perceived to be worth less than the rewards attained now and in the short term. Given an integer $k \leq n$, the goal is to compute a walk of length at most k maximizing the discounted total reward of the walk. Design and analyze an algorithm for this problem.

Solution. Our approach is to use a modified $\underline{\texttt{SW}(s,t,k)}$ from the text to get the maximum value of a k-walk starting from s.

```
max-discount-walk(s, t, k):
/* Given a nonnegative integer $k \in \mathbb{Z}_{\geq 0}$ and two vertices $s, t \in V$, returns the maximum total
   reward for a path of length $k$ from $s$ to a vertex $t$. */
1. If $k = 0$:
    A. If $s = t$, return 0
    B. Else, return $-\infty$
2. Return $\max \begin{cases} \texttt{max-discount-walk}(s,t,k-1) \\ \max_{v \in V \,:\, e:=(v,t) \in E} \left\{ \texttt{max-discount-walk}(s,v,k-1) + \alpha^{k-1} \cdot r(e) \right\} \end{cases}$
```

We assume there exists some globally defined directed graph $G = (V, E)$ with positive edge weights $r : E \to \mathbb{R}_{>0}$. To use this algorithm, we run $\underline{\texttt{max-discount-walk}(s,t,k)}$ for all $t \in V$ and $i \leq k$, using DP to cache the solutions to the sub-problems. That is,

$$\max_{\substack{v \in V \\ 0 \leq i \leq k}} \left\{ \underline{\texttt{max-discount-walk}(s,t,i)} \right\} \tag{1}$$

Runtime. Since the meaningful structure of our algorithm is exactly $\underline{\texttt{SW}(s,t,k)}$ from page 145 of the text, by Lemma 8.2 we can compute $\underline{\texttt{max-discount-walk}(s,t,k)}$ for all $t \in V$ and $i \leq k$ in $O(k(m+n))$ time. ■

Correctness. We prove correctness by strong induction. The base case when $k = 0$ is trivial.

Now, assume we have shown that the desired property holds for all values up to k. We wish to show that it must then also hold for k. By assumption, a $k-1$-walk gives a discounted total reward of

$$r(e_1) + \alpha r(e_2) + \cdots + \alpha^{k-2} r(e_{k-1}).$$

Upon running $\underline{\texttt{max-discount-walk}(s,t,k)}$, our algorithm presents two cases: either it adds a new edge, or it does nothing.

Case 1. If $\underline{\texttt{MDW}(s,t,k)} = \underline{\texttt{MDW}(s,t,k-1)}$, then correctness follows directly from the inductive hypothesis.

Case 2. If $\underline{\texttt{MDW}(s,t,k)} \neq \underline{\texttt{MDW}(s,t,k-1)}$, then $\underline{\texttt{MDW}(s,t,k)} = \underline{\texttt{MDW}(s,t,k-1)} + \alpha^{k-1} r(e_k)$ for some k^{th} edge e_k. By the inductive hypothesis, we have that

$$\begin{aligned} \underline{\texttt{MDW}(s,t,k)} &= \underline{\texttt{MDW}(s,t,k-1)} + \alpha^{k-1} r(e_k) \\ &= r(e_1) + \alpha r(e_2) + \cdots + \alpha^{k-2} r(e_{k-1}) + \alpha^{k-1} r(e_k). \end{aligned}$$

Thus this algorithm is correct for all $k \in \mathbb{N}_0$. ■

□