CSCE 411-200, Fall 2016
Homework 5 Solutions

Problem 1: Exercise 24.3-8 (p. 664): Let $G = (V, E)$ be a weighted, directed graph with nonnegative weight function $w : E \to \{0, 1, \dots, W\}$ for some nonnegative integer W. Modify Dijkstra's algorithm to compute the shortest paths from a given source vertex s in $O(WV + E)$ time.
Solution: Recall that the running time for Dijkstra's algorithm using a generic priority queue implementation is $O(V(T_{ins} + T_{ex}) + E \cdot T_{dec})$, where T_{ins} is the time to insert into the priority queue, T_{ex} is the time to extract the minimum value from the priority queue, and T_{dec} is the time to decrease the key value of an element in the priority queue. Let's look for a priority queue implementation with constant time per insert, constant time per decrease-key, and $O(WV)$ time total for *all* extract-min's.

Since all weights are nonnegative, the algorithm will never be considering paths with cycles in them. Since each edge has weight at most W, the maximum weight of any acyclic path is $W(|V| - 1)$, so the maximum value of an finite d-value in the algorithm is $W(|V| - 1)$. Let's implement the priority queue as an array A of doubly linked lists. The size of the array A is $W(|V| - 1) + 2$, i.e., one entry for each possible d-value (from 0 to $W(|V| - 1)$ inclusive and also infinity). The linked list associated with $A[i]$ holds all vertices whose d-value is i, $i = 0, \dots, W(|V| - 1)$, while the linked list associated with $A[W(|V| - 1) + 1]$ holds all vertices with infinite d-value.

To insert a vertex v with d-value i into the priority queue: insert v at the head of the linked list associated with $A[i]$. This takes constant time.

To decrease the d-value of vertex v from i to j: remove v from the linked list at $A[i]$ and add v to the head of the linked list at $A[j]$. This takes constant time.

To extract the vertex with the minimum d-value: First cut idea is to search through A starting at index 0 until finding a nonempty linked list, and then remove and return the vertex at the head of that list. However, if every extract-min operation starts at index 0, the total time for all the exract-min operations will be $O(WV^2)$, which is too big. The key is to notice that the minimum d-value in the whole graph never decreases, since there are no negative edge weights. Thus once the search in extract-min has gotten to index i, there is never any need to search smaller indices. So keep a pointer to the current location in A and start each extract-min search at that location. Total time now is $O(WV)$.

Problem 2: Exercise 5.2-5 (p. 122): Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an **inversion** of A. Suppose the elements of A form a uniform random permutation of 1 through n. Use indicator random variables to compute the expected number of inversions.
Solution: The sample space is all permutations of 1 through n. Let X be the random variable that is the number of inversions. We want to compute $E[X]$. For each $i = 1, \dots, n-1$ and each $j = i+1, \dots n$, define the indicator random variable X_{ij} as equal to 1 if (i, j) is an inversion and equal to 0 otherwise. Note that $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$.

$$\begin{aligned}
E[X] &= E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right] \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \text{ by linearity of expectation} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[X_{ij} = 1] \text{ by property of indicator random variables}
\end{aligned}$$

Note that in half the permutations, i comes before j, and in the other half, j comes before i. Since each permutation is equally likely, the probability that (i, j) is an inversion is $1/2$. That is, $\Pr[X_{ij} = 1] = 1/2$. Thus:

$$\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} \\
&= \frac{1}{2} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 \\
&= \frac{1}{2} \sum_{i=1}^{n-1} (n-i) \\
&= \frac{1}{2} \left(\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i \right) \\
&= \frac{1}{2} \left((n-1)n - \frac{(n-1)n}{2} \right) \\
&= \frac{n(n-1)}{4}.
\end{aligned}$$

Problem 3: Exercise 7.4-4 (p. 184): Show that RANDOMIZED-QUICKSORT's expected running time is $\Omega(n \lg n)$.
Solution: Inspired by page 184 (and assuming n is even), we have:

$$\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&> \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{1}{k+1} && \text{change of variable for inner summation} \\
&> \sum_{i=1}^{n/2} \sum_{k=1}^{n-i} \frac{1}{k+1} && \text{consider fewer terms in outer summation} \\
&> \sum_{i=1}^{n/2} \sum_{k=1}^{n/2} \frac{1}{k+1} && \text{since } i \le n/2,\ n-i \ge n/2 \\
&> \frac{n}{2} \sum_{k=1}^{n/2} \frac{1}{k+1} && \text{since no more reliance on } i \\
&= \frac{n}{2} \sum_{h=2}^{n/2+1} \frac{1}{h} && \text{change of variable for summation} \\
&> \frac{n}{2} \sum_{h=2}^{n/2} \frac{1}{h}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{n}{2}\left(\sum_{h=1}^{n/2}\frac{1}{h} - 1\right)\\
&= \frac{n}{2}\left(\ln\frac{n}{2} + O(1) - 1\right) \text{ by (A.7)}\\
&= \frac{n}{2}(\ln n - \ln 2 + O(1) - 1)\\
&= \Omega(n \lg n).
\end{aligned}
$$

Problem 4: Exercise 26.2-9 (p. 731): Suppose that both f and f' are flows in a network G and we compute flow $f \uparrow f'$. Does the augmented flow satisfy the flow conservation property? Does it satisfy the capacity constraint? For each of the two properties, either prove that every such augmented flow satisfies the property or give a counter-example.

Solution: First note that for all u and v, $(f \uparrow f')(u,v) = f(u,v) + f'(u,v)$. Why? Suppose $(u,v) \notin E$. Since both f and f' (separately) satisfy the capacity constraint for G, it follows that $f(u,v)$ and $f'(u,v)$ are both 0. Also, the definition of $(f \uparrow f')$ states that $(f \uparrow f')(u,v)$ is 0 in this case.

Now suppose that $(u,v) \in E$. Since G is a flow network, $(v,u) \notin E$. Thus $f'(v,u) = 0$ since f' satisfies the capacity constraint. The definition of $(f \uparrow f')(u,v)$ is $f(u,v) + f'(u,v) - f'(v,u)$, which equals $f(u,v) + f'(u,v)$.

Now we can show that flow conservation is satisfied by $(f \uparrow f')$. Let u be an vertex other than s and t.

$$
\begin{aligned}
\sum_{v\in V}(f \uparrow f')(u,v) &= \sum_{v\in V} f(u,v) + \sum_{v\in V} f'(u,v) \text{ by note above}\\
&= \sum_{v\in V} f(v,u) + \sum_{v\in V} f'(v,u) \text{ by flow conservation of } f \text{ and } f' \text{ separately}\\
&= \sum_{v\in V}(f \uparrow f')(v,u) \text{ by note above}
\end{aligned}
$$

Note: This is different than what is done in Ford-Fulkerson. In that case, we compute $(f \uparrow f_p)$, where f_p is a flow in the residual network G_f, but not necessarily a flow in G. In this exercise, both f and f' are flows in G.

Now let's consider the capacity constraint condition. This is not necessarily satisfied. A simple example is the flow network that consists of just s and t with one edge from s to t. Let 1 be the capacity of (s,t), let $f(s,t) = 1$ and $f'(s,t) = 1$. Then $(f \uparrow f')(s,t) = 2$, which is larger than the capacity.

Problem 5: Exercise 26.3-2 (p. 735): Prove the integrality theorem (26.10), saying that if the capacity function c takes on only integral values, then the maximum flow f produced by the Ford-Fulkerson method has the property that $|f|$ is an integer, and moreover, $f(u,v)$ is an integer for all vertices u and v.

Solution: Show by induction on the iterations of the while loop that $f(u,v)$ is an integer for all vertices u and v. Initially, $f(u,v) = 0$ for all u and v. Suppose $f(u,v)$ is an integer for all u and v before an iteration of the while loop. Since all capacities are integers, all residual capacities are integers. Thus the minimum residual capacity on the augmenting path chosen is an integer. The flow for each edge in E is updated at the end of the iteration by starting with the previous value (an integer by the inductive hypothesis) and then adding and subtracting values that are either 0 or are the minimum residual capacity on the augmenting path, which we argued just above is an integer. Thus the result is an integer.

Now that we now that $f(u,v)$ is an integer for all u and v, it follows that $|f|$ is an integer, since $|f|$ is calculated by adding and subtracting various flow values.