Assignment 3

Xiaoting Li (xil139)
Ziyu Zhang (ziz41)
Deniz Unal (des2014)

4. Problem 26-5 from the CLRS text.

Answer:

(a) The capacity of a cut from G is less or equal to the sum of capacity of all edges, and for each edge the max capacity is less or equal to C, so the cut is less or equal to all the edges with C capacity, which is C | E |.

(b) Since the capacity of an augmenting path is the min capacity of all the edges in the path, so the path should have all the edges with at least K capacity. We could modify the graph G by removing all the edges that has less capacity than K (which is $O(E)$), and do a depth first search to find the first path (which is $O(V+E)$) to find the path. And the total time complexity would be $O(V+E)$, which is $O(E)$.

(c) According to the Theorem 26.6, f is a max flow in G is equivalent to the residual network G contains no augmenting paths. In line 4, when the algorithm terminates, there is no augmenting path (K must be less than 1, and since capacity is integer, it means $k = 0$).

(d) At line 5, there is no edge that has larger capacity than k, same reason as (a), the min cut of G will be less than $K|E|$. And after line 7, $K = K/2$, when roll up to line 4, the min cut of G will be less than $2K|E|$.

(e) We know that before we enter the while loop of lines 5-6 (when line 4 is executed), there is a minimum cut with capacity $2K|E|$ at most. Each time the while loop of lines 5-6 is executed, we increase the flow in G by at least K as the augmenting path we find in each iteration has capacity at least K. The maximum amount of flow that we can get is $2K|E|$ as the value of any flow f in a flow network G is bounded from above by the capacity of any cut of G (Corollary 26.5 from textbook). Therefore, this while loop can have $2K|E|/K = 2|E|$ iterations at most. This gives us the upper bound $O(E)$

(f) Since $K = 2^{lgC}$ in the beginning we can say K is bounded above by C. The outer while loop will be executed $O(lgC)$ times as K is halved in each iteration. We already know from section (e) the inner while loop will be executed

$O(E)$ times and we know from section (b) that finding an augmenting path p with capacity at least K will take $O(E)$ time. So the algorithm will run in $O(E^2 lgC)$ time in total.

5. Problem 26-6 from the CLRS text.
Answer:
(a) If M is a matching and P is an augmenting path with respect to M, it means that there are k edges in M and there are $k+1$ edges not in M. Since we are calculating the symmetric difference between M and P, it is easy for us to learn that there are $2k$ edges in $M \cap P$. Therefore, we can get

$$|M \oplus P| = |M \cup P| - |M \cap P| = |M| + 2k + 1 - 2k = |M| + 1$$

And we can get

$$M \oplus (P_1 \cup P_2 \cup ... \cup P_k) = |M \cup P_1 \cup P2 \cup ... \cup P_k| - |M \cap P1 \cap P_2 \cap ... \cap P_k| =$$

$$|M| \cup |P_1| \cup |P_2| \cup ... \cup |P_k| - |M \cap P1 \cap P_2 \cap ... \cap P_k| =$$

$$|M| + (2k_1 + 1) + (2k_2 + 1) + ... + (2k_k + 1) - (2k_1 + 2k_2 + ... + 2k_k) =$$

$$|M| + k$$

(b) Since $G^{'} = (V, M \oplus M^*)$, at least 2 of the edges come from the same matching if each vertex has more than 3 degrees. However, this contradicts with the definition of matching. No two edges in a matching have a common vertex. Therefore, every vertex in $G^{'}$ has at most 2 degrees. Since every vertex in $G^{'}$ has at most 2 degrees, it means there is no path in the graph that has repeated vertices. Therefore, $G^{'}$ is a disjoint union of simple paths and cycles. Since two edges with the same vertex cannot appear in the same matching and $G^{'} = (V, M \oplus M^*)$, so edges in each such path or cycle belong alternately to M or M^*. Since $|M| < |M^*|$ and we've already known that edges in $G^{'}$ are alternated between M and M^*, it means a path that contains one more edge of M^* than M is an augmenting path for M. Therefore, $M \oplus M^*$ contains at least $|M^*| - |M|$ vertex-disjoint augmenting paths with respect to M.

(c) Since P is the shortest augmenting path with respect to $M^{'}$ and $M^{'} = M \oplus (P_1 \cup P_2 \cup ... \cup P_k)$, it means that any edge of P that is not in $M^{'}$ cannot be in M, any edge of P that is in $M^{'}$ must be in M. We also know that P has edges alternate between $M^{'}$ and $E^{'} - M^{'}$ and both of the endpoints of $M^{'}$ are not in M. So we can tell that P is also an augmenting path with respect to M and P must be at least of length l. Since $P_1, P_2, ..., P_k$ is a maximum set of vertex-disjoint augmenting paths of length l and P is vertex-disjoint with $P_1 \cup P_2 \cup ... \cup P_k$, if P also has the length of l, then it is a contradiction. Therefore, P has more than l edges.

(d) Any edge in $M \oplus M^{'}$ is either in M or $M^{'}$. Since $M^{'} = M \oplus (P_1 \cup P_2 \cup ... \cup P_k)$, it means that any edge in $M^{'}$ is either in M or $P_1 \cup P_2 \cup ... \cup P_k$. So the symmetric difference between M and $M^{'}$ is $P_1 \cup P_2 \cup ... \cup P_k$. And $A = M \oplus M^{'} \oplus P$. Therefore, $A = (P_1 \cup P_2 \cup ... \cup P_k) \oplus P$. Based on what we get from (a), we can say $|M^{'} \oplus (P_1 \cup P_2 \cup ... \cup P_k)| - |M| = k + 1$, which means there are at least $k+1$ more vertex-disjoint augmenting path with respect to M. Since the length of such an augmenting path is at least l, so we can conclude that $|A| \geq (k+1)l$. Since $P_1, P_2, ..., P_k$ are vertex-disjoint augmenting path, they must have at least kl distinct edges. P is not vertex-disjoint with $P_1, P_2, ..., P_k$, which means that it shares at least one edge with any of $P_1, P_2, ..., P_k$. But it still needs to have at least l distinct edges to make the equation $|A| \geq (k+1)l$ work. Therefore, P has more than l edges.

(e) We know that the shortest augmenting path with respect to M has l edges. This means that each shortest augmenting path with respect to M contains at least $l + 1$ vertices. Also, from (b) we learn that $M \oplus M^*$ contains at least $|M^*| - |M|$ vertex-disjoint augmenting paths with respect to M. So $|M^*| - |M|$ has at most $|V|/(l + 1)$ augmenting paths with respect to M. Therefore, the size of the maximum matching is at most $|M| + |V|/(l + 1)$.

(f) From (e) we learn that the size of the maximum matching is at most $|M| + |V|/(l + 1)$. Let M^* be the maximum matching and M be the matching after $\sqrt{|V|}$ iterations. So the shortest augmenting path must be at least $\sqrt{|V|}$. Since we have $|M^*| - |M| \leq |V|/(l + 1)$, we can get $|M^*| - |M| \leq |V|/(\sqrt{|V|} + 1) \leq \sqrt{|V|}$. It means after $\sqrt{|V|}$ iterations, the algorithm can grow no more than $\sqrt{|V|}$. So we can conclude that the number of repeat loop iterations in the algorithm is at most $2\sqrt{|V|}$.

(g) Modify the graph by adding source vertex s, sink vertex t, directed edges between s and free vertices in L, and directed edges between free vertices in R and t. Run BFS on the modified graph to get a shortest path between s and t. Pick a free vertex in L, then use BFS to find path alternating between matched and unmatched edges until it hits a free vertex in R. The first time on it hits on a free vertex in R, we know that the length of the shortest augmenting path is k. In this way, we can stop early in the following search if the length is larger than k. Then we perform a greedy algorithm to trace from the ending free vertex back to the starting free vertex to see if the starting free vertex is unmarked. If it is unmarked, we find a path and marked the starting free vertex. This takes $O(E)$ time, which means each iteration takes $O(E)$ time. From (f) we learn that the number of repeat loop iterations in the algorithm is at most $2\sqrt{|V|}$. So we can conclude that the total running time of HOPCROFT-KARP is $O(\sqrt{|V|}E)$.