CS325: Analysis of Algorithms, Winter 2024

Practice Assignment 4 – Solutions

Problem 1. An undirected graph $G = (V, E)$ is bipartite if the vertices V can be partitioned into two subsets L and R, such that every edge has exactly one endpoint in L and one endpoint in R.

(a) Prove that every tree is a bipartite graph.

(b) Prove that a graph G is bipartite if and only if every cycle in G has an even number of edges.

(c) Describe and analyze an efficient algorithm that determines whether a given undirected graph is bipartite.

Try to work on this problem as you read about graph search algorithms.

Solution.

(a) This is implied by part (b), (Why?).

(b) ($\Rightarrow$) Let $G = (V, E)$ be a bipartite graph. Therefore, V can be partitioned to non-empty X and Y, such that every edge in E has exactly one endpoint in X and one endpoint in Y. Now, let $C = (v_1, v_2, \ldots, v_n = v_1)$ be any cycle in G. Suppose $v_1 \in X$, the other case is similar. It follows that for all even $1 \leq i \leq n$, $v_i \in Y$, and for all odd $1 \leq i \leq n$, $v_i \in X$. In particular, $v_n = v_1 \in X$, hence n is odd. Therefore, C has an even number of edges.

($\Leftarrow$) Suppose $G = (V, E)$ does not have any cycle with an odd number of edges. Let $s \in V$ be an arbitrary vertex of G, and let T be the BFS tree rooted at s. Further, for each pair of vertices $u, v \in V$, let $\ell(u, v)$ denote the length of the u to v path in T. Now, let X be the set of vertices $v \in V$ for which $\ell(s, v)$ is even; in particular $s \in X$. Also, let Y be the set of vertices $v \in V$ for which $\ell(s, v)$ is odd. We claim that all the edges of G have exactly one endpoint in X and one endpoint in Y (thus, G is bipartite). We use proof by contradiction to show this claim holds. Suppose, to derive a contradiction, there exists and edge $uv \in E$ with $u, v \in X$; the other case that $u, v \in Y$ is similar. There are two cases to consider.

(1) The vertices u and v have ancestor/descendent relationship in T. Since both $u, v \in X$, $\ell(u, v)$ is even. Since $uv \in E$, the cycle obtained from the u to v path in T plus the edge uv has odd length, which is a contradiction.

(2) The vertices u and v do not have ancestor/descendent relationship T. In this case let w be the lowest common ancestor of u and v in T. Note,

$$\ell(u, v) = \ell(u, w) + \ell(w, v),$$

which has the same parity as

$$\ell(u, w) + \ell(w, v) + 2\ell(w, s) = (\ell(u, w) + \ell(w, s)) + (\ell(w, v) + \ell(w, s)) = \ell(s, u) + \ell(s, v)$$

which is even because $\ell(s, u)$ and $\ell(s, v)$ have the same parity (as $x, y \in X$). We conclude that the cycle obtained by the path between u and v in G and the uv edge has odd length, which is a contradiction,

(c) The algorithm follows from the proof of part (b). Let X and Y be as explained in part (b). They can be computed by running BFS from s. Then, check if there exists any edge with both endpoints in X or both endpoints in Y. If there is such an edge, output that the graph is not bipartite. Otherwise, output that the graph is bipartite. X and Y can be computed in $O(V + E)$, the running time of BFS. The check can be done in $O(E)$ time, with on pass on the set of edges. Therefore, the total running time is $O(V + E)$.

Problem 2. Describe and analyze an algorithm to compute an optimal ternary prefix-free code for a given array of frequencies $f[1 \ldots n]$. Don't forget to prove that your algorithm is correct for all n. This is a good exercise to ensure that you understand Huffman codes. What you get here should be very similar to the Huffman code; try to modify each step/proof of Huffman codes to work for ternary codes instead of binary codes.

Solution. The algorithm is very similar to the binary Huffman code. Let's assume that the number of symbols is odd. If not we add a new symbol with frequency zero to make it odd (Why is this alright?). At each step, we find the three letters x, y and z with the least frequencies. We replace them with a new letter $\overline{xyz}$, with frequency $f[\overline{xyz}] = f[x]+f[y]+f[z]$. Then, we recursively compute the set of ternary codes for the new set of letters. After the recursive call, the code for every letter is known except for x, y and z. Let the code for $\overline{xyz}$ calculated recursively be c. We set the code of x, y and z to $0 \circ c$, $1 \circ c$ and $2 \circ c$ in order, where $\circ$ is concatenation.

The proof of correctness is very similar to the binary case. First, we show that there exists an optimal tree with the three lease frequent letters being siblings and deepest vertices in the tree. Next, we show that the optimal solution in the recursive call implies the optimal solution of the main problem.

The asymptotic running time is the same as the running time of the binary Huffman code.

Problem 3. For each of the following statements, respond *Ture*, *False*, or *Unknown*.

(a) If a problem is decidable then it is in P. **F**

(b) For any decision problem there exists an algorithm with exponential running time. **F**

(c) $P = NP$. **U**

(d) All NP-complete problems can be solved in polynomial time. **U**

(e) If there is a reduction from a problem A to Circuit SAT then A is NP-hard. **F**

(f) If problem A can be solved in polynomial time then A is in NP. **T**

(g) If problem A is in NP then it is NP-complete. **F**

(h) If problem A is in NP then there is no polynomial time algorithm for solving A. **F**