HEURISTICS GUILD

Correction to Application: Hamiltonian Cycle

Karthik Kashyap

EE23B030™

All of the codes referenced in this application can be found on my GitHub.

§2 Heuristics

4. *(Welcome to Heuristics)* Solve the problem sheet and explain how each question is different and what approaches you could apply for each question.

Clarification in my answer for Problem B:
Thank you for pointing this out. I have also pointed out the same confusion in my original application too, but now I get that it was actually due to the test cases that it passed and not due to the algorithm.

Also thank you for hinting at a DP solution, hence I took reference from GeeksforGeeks's algorithm for finding if a Hamiltonian path exists using `DP` and `Bitmasks`, and then used this to find the Hamiltonian cycle.

Here, we use bitmasks to cleverly identify which nodes of the graph have been traversed in the path. For example, $mask = 0b010011$ indicates that nodes $0, 1, 4$ have been traversed in the path so far.

Combining this with a boolean DP, we denote `dp[i][mask]` to be true if there is a path with the nodes present in the mask which ends at node `i`. We initialise this with `dp[i][1<<i] = true` as every path containing `i` as the only node must end at `i`.

Now, we traverse through all set of masks from 0 to $2^N - 1$ and update `d[j][mask]` to be `true` if there exists a node `k` such that there exists a path ending at `k` with nodes contained in mask except `j`.

After the DP is done executing, we now need to check if there is a path with mask `(1<<N) - 1`, i.e., containing all the nodes.

To find the path, we can maintain a `parent` $2D$ vector which keeps track of parent nodes in subsequent masks.

Code can be found on the next page.

The Time Complexity of the algorithm is now $\mathcal{O}(n^2 \times 2^n)$. For $n \leq 20$, the maximum number of iterations is 4×10^8, which is relatively good enough to run in the given time limit. So hence, my confusion from the application was cleared and this algorithm does run perfectly under the constraints.

Note: The code seems to be erroring out in some test cases when I submit it on HackerRank, but when I manually run it for the same test cases and verify the output, it seems to be a valid Hamiltonian Cycle. I think the program is encountering runtime errors due to excessive memory usage.

```cpp
vector<int> check_cycle(int n, vector<vector<int>> &roads)
{
    vector<vector<bool>> dp(n, vector<bool>(1 << n, false));
    vector<vector<int>> parent(n, vector<int>(1 << n, -1));
    for (int i = 0; i < n; i++) dp[i][1 << i] = true;

    for (int i = 0; i < (1 << n); i++) {
        for (int j = 0; j < n; j++) {
            if (i & 1 << j) {
                for (int k = 0; k < n; k++) {
                    if ((i & 1 << k) && roads[k][j] && k != j && dp[k][i ^ (1 << j)]) {
                        dp[j][i] = true;
                        parent[j][i] = k;
                    }
                }
            }
        }
    }

    vector<int> path;
    int end_node = -1;
    for (int i = 1; i < n; i++) { // Start from i = 1 so that i != 0
        if (dp[i][(1 << n) - 1] && roads[i][0]) {
            end_node = i;
            break;
        }
    }

    if (end_node == -1) return {-1};

    int node = end_node;
    int mask = (1 << n) - 1;
    while (node != -1) {
        path.push_back(node);
        int next = parent[node][mask];
        mask ^= 1 << node;
        node = next;
    }

    reverse(path.begin(), path.end());
    int zero_ind = -1;
    for (int i = 0; i < n; i++) {
        if (path[i] == 0) {
            zero_ind = i;
            break;
        }
    }
    rotate(path.begin(), path.begin() + zero_ind, path.end());
    path.push_back(0);
    return path;
}
```