UM–SJTU Joint Institute
VE477 Intro to Algorithms

Homework 3

Wang, Tianze
515370910202

Question 1 Hamilton Path

A Hamilton Path is a path that visit each vertex in a graph exactly once.

(1)

Not done yet.

(2)

Not done yet.

(3)

Algorithm 1: Hamilton Algorithm

Input : An undirected graph G
Output : The Hamilton Path in G

```
Function Hamilton(G):
    L ← [];
    S ← nodes with no coming edges;
    if S size >1 then
        return No result
    end if
    if S size == 0 then
        Reverse the direction of all edges;
        S′ ← nodes with no coming edges;
        if S′ size ==0 then
            return Exist Hamilton Path
        else
            while S ≠ ∅ do
                remove n from S;
                Append n to tail of L;
                for node m with an edge e from n to m do
                    remove e from graph.
                end for
                Update S;
            end while
            if Graph has other edges then
                return No result
            else
                return L
            end if
        end if
    end if
    while S ≠ ∅ do
        remove n from S;
        Append n to tail of L;
        for node m with an edge e from n to m do
            remove e from graph.
        end for
        Update S;
    end while
    if Graph has other edges then
        return No result
    else
        return L
    end if
end
```

(4) Complexity

The complexity is a DFS with stack, and the algorithm will visit every vertex and every edge in the graph for only once, so the overall complexity is

$$\mathcal{O}(V + E)$$

(5)

Linear Complexity.

Question 2 Critical Thinking

(1)

The answer is no. The reason is that, to be bounded by polynomial, it should have time complexity of $\mathcal{O}(n^k)$. However, in this case, for

$$\lceil \log n \rceil !$$

We are not able to find a k satisfying the condition since the existing factorial calculation.

(2)

Yes.
The definition of $\log^*$ is

$$x = \begin{cases} 0 & x \leq 1 \\ 1 + \log^* \log_2 x & x > 1 \end{cases}$$

And the definition of $f(x)$ to be asymptotically larger than $g(x)$ is

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \infty$$

In this problem,

$$\log^* \log n = \log^* (\frac{\log_2 n}{\log_2 e})$$

which means,

$$\log^* n - 2 < \log^* \log n < \log * n - 1$$

And we can conclude

$$\log^* \log n = O(\log^* n)$$

The result is then

$$\lim_{x \to \infty} \frac{\log^* n - 1}{\log \log^* n} = \lim_{a \to \infty} \frac{a}{\log a} = \infty$$

□

(3)

We will only need 3 weigh.

Input : 8 balls with one lighter and 7 same weight
Output : The light weight ball

Function get_ball(8 balls):
    Divide 8 balls into two 4-ball groups, A_1, A_2;
    Weigh A_1, A_2;
    Divide lighter-weight 4 balls into two 2-ball groups, B_1, B_2;
    Weigh B_1, B_2;
    Weigh inside the lighter-weight group from B_1 and B_2. **return** *Lighter weight ball from previous weigh*
end

Question 3 Rubik's Cube

The Rubik's cube is a cube with 6 sides in different color, and each side is cut into 9 identical square pieces. There is an internal pivot such that enable each side to rotate, when the side is rotating, 3 adjacent square pieces that adjacent to the rotating side in all 4 adjacent sides will move with the side. To solve a Rubik's cube, all sides should be restored to same color.

Solving Strategy 1: CFOP A method called CFOP, which is a short spelling for Cross, First-2-layers, Orienting-the-last-layer, Permutation-last-layer. First build a cross, on a single layer, the cross should be also valid for the adjacent for side. Second, finish the bottom two layer according to formula. Third, finish upper slide, without considering the permutation of the layer.(namely the color are correct however the position may be wrong) Fourth, finish the permutation, using formula.

Solving Strategy 2: Heise Method A method is defined in 4 steps. First, build 4 successive square-shaped blocks.

Second, match the sequence and orient edges.

Third, solve the remaining edges and any two corners. Fourth, solve all the corners.
https://www.ryanheise.com/cube/heise_method.html