UM-SJTU JOINT INSTITUTE
VE477 Introduction to Algorithms

Homework 3

Li Yong 517370910222

October 19, 2020

Ex.1 Hamiltonian path

1. Depth-first search is an algorithm for traversing or searching tree or graph data structure. The algorithm starts at the root node (or selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

   Time complexity: $\mathcal{O}(|V| + |E|)$

Algorithm 1 Depth-first search

Input: $G(E, V)$
Output: DFS path

```
Let S be a stack
S.push(v)
while S is not empty do
    v ← S.pop()
    if v is not explored then
        Set v is explored
        for all adjacent nodes u of v do
            S.push(u)
        end for
    end if
end while
```

2. Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge (u, v), vertex u comes before v in the ordering. Topological sorting can be only applied on DAG.

   Time complexity: $\mathcal{O}(|V| + |E|)$

Algorithm 2 Topological sorting

Input: $DAG(E, V)$
Output: A linked list of ordered vertexes.

```
Call DFS to compute finishing times v.f for each vertex v in DAG
As each vertex is finished, insert it into the front of a linked list
return The link list
```

Algorithm 3 Hamiltonian path

3. **Input:** $DAG(E, V)$
Output: If $DAG(E, V)$ contains a Hamiltonian path

```
Call Topological sorting to get all results
As each result is got, check if the first and the last node in the linked list are adjacent
if there exists one result satisfies then
    return True
end if
return False
```

4. The number of results of topological sorting must be less than $|V| + |E|$, hence time complexity is $\mathcal{O}((|V| + |E|)^2)$.

5. $\mathcal{NP}$-complete

Ex.2 Critical thinking

(a) No. Plot the function, the figure skews exponentially.

(b)

$$\frac{\log^* \log n}{\log \log^* n} = \frac{\log^* n - 1}{\log \log^* n}$$

Let $t = \log^* n$, then

$$\frac{\log^* \log n}{\log \log^* n} = \frac{t - 1}{\log t}$$

Hence, $\log^* \log n$ is asymptotically larger than $\log \log^* n$.

(c)

Algorithm 4 The lighter ball

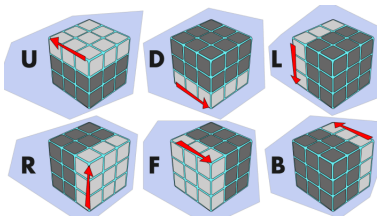
Input: 8 balls, one of which is lighter than others
Output: The lighter ball

```
Compare 6 balls by balance, each side of 3
if the balance is balanced then
    Compare the left 2 balls
    return the lighter ball
else
    Compare 2 of the 3 balls which are lighter
    if the balance is balanced then
        return the left 1 ball
    else
        return the lighter ball
    end if
end if
```

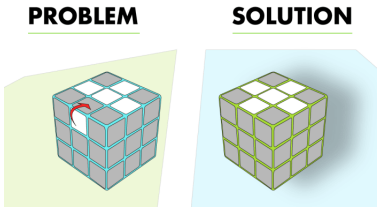
Ex.3 Rubik's Cube

A Rubik's cube is a mechanical puzzle in the shape of a 3×3 cube which is comprised of 26 smaller cubes, with a rotating mechanism in the middle. Each piece of surface of smaller cubes is covered by one of six colors. When the puzzle is solved, each surface of the Rubik's cube, which consists of 9 pieces of surface of smaller cube, is covered by one color. It requires a series of movement sequences, or algorithms, in order to be solved.

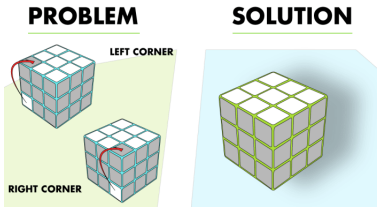
- **Algorithm 1[1]**
  Movement notation:



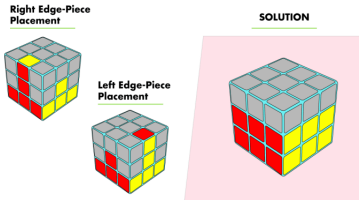
1. Getting the White Cross
   F'UL'U'



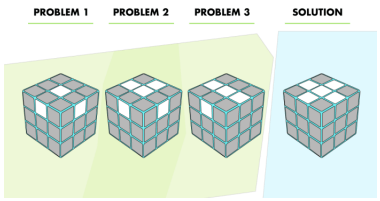
2. First Layer: Left and Right Corners
   (a) Left corner: DLD'L'
   (b) Right corner: D'R'DR



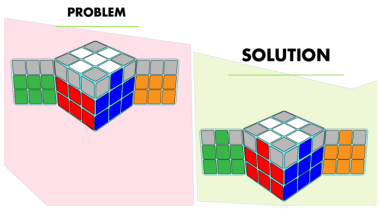
3. Solving Edge-Piece Placement
   (a) Right edge-piece placement: URU'R'U'F'UF
   (b) Left edge-piece placement: U'L'ULUFU'F'



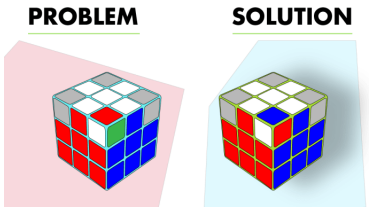
4. Getting the White Cross Without Disrupting the Rest of the Cube
   FRUR'U'F'



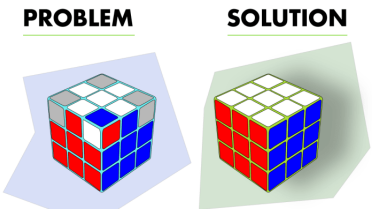
5. Solving Third Layer Edge Pieces
   RUR'URUUR'



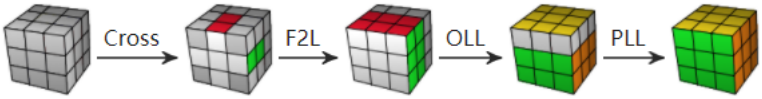
6. Aligning the Third Layer Corner Pieces
   URU'L'UR'U'L



7. Aligning the Corners and Finishing the Cube
   R'D'RD



- **Algorithm 2[2]**
  Fridrich (CFOP) method.



  1. Cross
  2. First two layers (F2L)
  3. Orienting the last layer (OLL)
  4. Permutate the last layer (PLL)

Ex.4 The $\mathcal{NP}$ classes

1. We could simply use DFS to check if a given graph has a simple path, the complexity of which is $\mathcal{O}(|V| + |E|)$. Hence it is in $\mathcal{NP}$.

2. The complexity of the simplest way that we just check from 1 to the given integer n is $\mathcal{O}(n)$. Hence it is in $\mathcal{NP}$.

3. Let us consider the reduction of vertex cover problem to *clique* problem[1], which is $\mathcal{NP}$-complete, a give graph G has a clique of size k if and only if the complement of G, $\bar{G} = (V, \bar{E})$ has a vertex cover of size $|V| - k$.

   Assume that G has a clique $V' \subseteq V$, and $|V'| = k$. Let (u, v) be an arbitrary edge in $\bar{E}$, then $(u, v) \notin E$. So that at least one of u and v does not belong to V'. Equivalently, at least one of u and v belongs to $V \backslash V'$, i.e., (u, v) is covered by $V \backslash V'$. Hence, $V \backslash V'$ size of $|V| - k$ is a vertex cover of $\bar{G}$. Vice versa.

Ex.5 PRIMES is in $\mathcal{P}$

References

[1] https://hobbylark.com/puzzles/Rubik-Cube-Algorithms

[2] https://ruwix.com/the-rubiks-cube/advanced-cfop-fridrich/

[3] https://www.zhihu.com/question/47423427

[1] Clique of $G(V, E)$ is a subset of V, where each couple of vertexs is connected by one of the edges in E.