VE477 HW 3

Wu Jiayao 517370910257

October 17, 2019

1

1.1

Not done

1.2

Not done

1.3

Algorithm 1: Hamiltonian path problem

Input : A graph $G(V, E)$
Output: Whether there is a Hamiltonian path

```
Function topSorting(V,E):
    visitNum ← 0
    H ← {}
    while E ≠ ∅ do
        v ← a vertex in V that no edge points to it
        visitNum++
        H[visitNum] ← v
        V ← V\v
        for vertex u in V do
            if edge < v, u > exists then
                E ← E\ < v, u >
    return H
H ← topSorting(V,E)
for i in range(0, len(H)) do
    if edge < H[i], H[i + 1] > is not in E then
        return false
return true
```

1.4

Topological sorting traverses all vertices in G. Overall, all edges are traversed for once, since one edge will be removed from E if traversed. Therefore, the complexity is

$$T(n) = \mathcal{O}(V + E)$$

1.5

It is $\mathcal{NP}$ -complete , as this problem is included in Karp's 21 $\mathcal{NP}$-complete problems.

2

2.1

It is not bounded by a polynomial.

2.2

Yes.

$$\log^* n := \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + \log^*(\log n) & \text{if } n > 1 \end{cases}$$

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

2.3

Input : Eight balls with one lighter in weight than each other
Output: The lighter ball

1. Divide into 2 groups of 4 balls. Weight two groups on the balance. Pick out the lighter group of 4 balls.
2. Divide the lighter group into 2 groups of 2 balls. Weight two groups on the balance. Pick out the lighter group of 2 balls.
3. Weight the two on the balance.
4. **return** the lighter ball

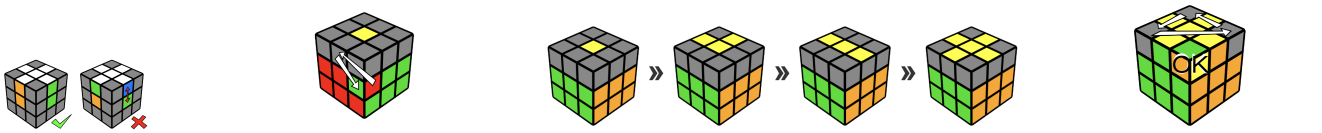
3

Rubik's Cube is a 3D combination puzzle. On a Rubik's Cube, each of the six faces was covered by nine stickers, with six solid colors. An internal pivot mechanism enables each face to turn independently, thus mixing up the colours. For the puzzle to be solved, each face must be returned to have only one colour for its nine stickers.

Two algorithms of solving Rubik's cube are listed as followed.

3.1

1. Get a white(other color is OK) plus(or cross namely) on the top face. Get the white corner back to the right place to make a whole white face. [Figure 1(a)]



(a) White Cross (b) Middle Layer's goal (c) Top cross (d) The goal of Step 4

2. Flip the cube to make white on the bottom. Get all the pieces in the middle layer to its right place. [Figure 1(b)] Make the cross on the top face. [Figure 1(c)]

3. By sune and antisune, move four of the yellow cross blocks to its right place that the color of another face of the block matches the color of the cubic's face this face belongs to. [Figure 1(d)]

4. Last, get all the corners of yellow to its right place and right direction.

3.2 ZZ method

1. This stage orients all edges on the cube (EO) while simultaneously placing the down-front and down-back edges (Line).

2. It completes the first two layers by building two 1x2x3 blocks on either side of the Line made in the previous step.

3. Solve the last layer.

References

[1] cube3x3.com/%E5%A6%82%E4%BD%95%E8%A7%A3%E5%86%B3%E9%AD%94%E6%96%B9/

[2] en.wikipedia.org/wiki/Rubik%27s_Cube

[3] cube.crider.co.uk/zz.php?v=beginner

4

4.1

It is $\mathcal{NP}$. Let y be a path in the graph. $\mathcal{O}(|V|)$ is needed for traverse to verify whether y is a simple path is $True$

4.2

Not done

4.3

Let y be k vertices in the graph. Then all edges are traversed to test whether each edge contains a vertex cover of size k. The time complexity is $\mathcal{O}(|E|\,k)$. Hence, it is $\mathcal{NP}$.

5

It is not sufficient. Because for an integer n, when n becomes bigger, the complexity of division for n is $\log_2 n$.

```
int Pos_Div(int x,int y)
{
  int ans=0;
  for(int i=log(n)/log(2);i>=0;i--)
  {
    if((x>>i)>=y)
    {
      ans+=(1<<i);
      x-=(y<<i);
    }
  }
  return ans;
}
```

Trial division can be solved in $\mathcal{O}(n)$ if and only if division of n is $\mathcal{O}(1)$, hence trial division is not $\mathcal{O}(n)$. Hence, it is not sufficient to conclude that $PRIME$ is $\mathcal{P}$.
By prime number theorem, for a binary base n digits number, the algorithm requires

$$\pi\left(2^{n/2}\right) \approx \frac{2^{n/2}}{\left(\frac{n}{2}\right)\ln 2}$$

times of division. The time complexity of division is $\mathcal{O}(\log_2(2^n)) = \mathcal{O}(n)$. The overall time complexity can be expressed as

$$T(n) = \pi\left(2^{n/2}\right) \times \mathcal{O}(n) = \mathcal{O}(2^{n/2})$$

It is not $\mathcal{P}$.