CS6033 Assign No.3

Minghe Yang

March 2021

1 Hamiltonian path

1.3

Algorithm 1 DAE Hamiltonian Path

Input: directed acyclic graph G with n nodes
Output: Hamiltonian path availability

```
while n > 0 do
    for i ∈ (1, n) do
        if in[i] == 0 and num[in[i] == 0] == 1 then
            Delete n_i from G
            n ← n − 1
        else return False
        end if
    end for
end while
return True
```

1.4 Consider the worst situation: Total complexity $=(|V| * |E|/2) + (|V| - 1)(|E| - |V| + 1) + \ldots$
Since $|E| \geq |V|$ for the most cases, we have complexity $O(|V||E|)$

1.5
For a directed acyclic graph like in this question, it is a P problem.
For a random graph, it belongs to a NP problem.

2 Critical thinking

2.1
It's not bounded by a polynomial.
From Stirling's Approximation we can get:
$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n \leq en^{n+1/2} * e^{-n}$

So $(\log n)! \leq \epsilon(\log n)^{0.5+logn} * e^{-\log n}$
$(\log n)^{\log n} = (e^{\log(\log n)})^{\log n} = e^{(\log n)(\log(\log n))}$
So the order of growth will be around:$e^{(\log n)(\log(\log n))-\log n} = n^{log(logn)-1}$
Since $\log\log n$will eventually exceed any positive integer, it's not bounded by polynomial.

2.2
Set $\log*(\log n) = k$, from the chapter ppt we can get that:
$\log(\log* n) = \log(1 + \log*(\log n)) = \log(1 + k)$
When k is large enough, it's just a comparison between k and $\log k$.
When$n = 16$,$\log n = 4, k = \log*4 = 2, \log(1 + k) = \log(3)$<2,
Since $k > logk$, we can conclude that it is asymptotically larger.

2.3
Needs at least twice weighting. pseudocode should be like this.

Algorithm 2 Light ball detection

Input: Eight balls (a, b, c, d, e, f, g, h)
Output: The lighter ball

```
weight {a, b, c} and {e, f, g}
if weight{a, b, c} ≤ weight{d, e, f} then
    weight a and b
    if weight a<weight b then
        return a
    else
        if weight a = weight b then
            return c
        else
            return b
        end if
    end if
else
    if weight{a, b, c} = weight{d, e, f} then
        weight g and h
        if g<h then
            return g
        else
            return h
        end if
    else
        weight d and e
        if weight d<weight e then
            return d
        else
            if weight d = weight e then
                return f
            else
                return e
            end if
        end if
    end if
end if
```

3 Rubik's Cube

3.0:Introduction

On the original classic Rubik's Cube, each of the six faces was covered by nine stickers, each of one of six solid colours: white, red, blue, orange, green, and yellow. Some later versions of the cube have been updated to use coloured plastic panels instead, which prevents peeling and fading.
The original (3×3×3) Rubik's Cube has eight corners and twelve edges. There are 8! (40,320) ways to arrange the corner cubes. Each corner has three possible orientations, although only seven (of eight) can be oriented independently; the orientation of the eighth (final) corner depends on the preceding seven, giving 37 (2,187) possibilities. There are 12!/2 (239,500,800) ways to arrange the edges, restricted from 12! because edges must be in an even permutation exactly when the corners are. (When arrangements of centres are also permitted, as described below, the rule is that the combined arrangement of corners, edges, and centres must be an even permutation.) Eleven edges can be flipped independently, with the flip of the twelfth depending on the preceding ones, giving 211 (2,048) possibilities.

$Re : https : //en.wikipedia.org/wiki/Rubik\%27s_Cube$

3.algorithm 1:Intuitive solving
Cross
Place four edges of one face, at their correct location and orientation. It's ok if they are rotated, but in the correct order, because you will always be able to align them by just rotating the chosen face.
You can notice that if an edge is on the opposite face in the correct orientation, you can move it to the correct face with the correct orientation with a half turn. If an edge is in the slice between those faces, you can always move it to the correct face with the right orientation with one quarter turn. Of course, you may need to rotate the destination face so that the edge arrives at the desired location, relative to other edges you have already placed.
You will obtain a cross. Turn the cube so that it is at the bottom.
Slots
After aligning the edges with the centers next to them, you will get four slots, one for each corner, that can contains two pieces : an edge piece and a corner piece. Notice that you can move some piece out one slot, for example the front-right-bottom slot (if you started with the bottom face), by doing R U' R or F' U F. This three moves are the following :
Taking the piece out of the slot to the upper face
Moving this piece out of the way
Bring back the bottom edge at it's correct place (inverse of step 1)

Conversely, you can put a pair of pieces into one slot :
Move the slot out, so that the destination is now on the scrambled face

Move some piece into the destination
Bring the destination back (inverse of step 1)
Of course, you will need to build pairs, so that you can insert them into the slots. To do so, notice that :
You can break a pair when pieces are not well oriented relative to each other by hiding the corner piece from the scramble face by moving the appropriate slot out, moving the edge with the scrambled face, and the bring the slot back You can build one pair by hiding a corner by moving a slot out, bring an edge next to it, and then bring the slot back If an edge or a corner is already in place, you can use the keyhole method, which can be done intuitively. That's all you need to know. There are shortcuts of course, notably when you can form the pair and put it in the slot in three moves (see F2L 4).

Orienting edges before last F2L pair

To orient remaing edges, notice that if you store an edge in a slot, and take it out using the adjacent face, it will be rotated. For example, if you want to flip the front-top edge using the front-right-bottom slot, you can do the following sequence : R U' R' then F' U F. By doing so, you will flip other edges too, so you must take into account the orientation of the edge currently in the slot so that you bring it out of the slot to the top face with the right orientation (yellow on top if the last layer is yellow). So if you have in the front-right-bottom an edge the has its yellow color to the right, the you must take it out with the F face. If it's in the other direction, you must take it out with the R face. When you take the edge out, you must take care that it does not replace the other edge that you will place back in the slot. You may need to rotate the upper face to move it out of the way. So each orientation will be 3 or 4 moves long :

move some edge out of the way (optional)
take the edge out of the slot
bring another edge instead
put this edge in the slot (inverse of step 2)

You are done when all edges the belongs to the top layer and are in the top layer are oriented (yellow on top). If one edge of the top face remains the the slot, it's ok. You will only need to take care of using the face the brings it out in the right orientation when doing the last pair.
Finally, do the last pair using only two faces : the upper face and another face the allows to bring out the content of the slot.

Rotating edges

Using the method above, the edges should be already oriented. If not, you will need a commutator. It can be composed with A = "flip an edge in the top face without changing anything else in the top face" and B = "rotate the top face to place another edge to be flipped at the intersection". If you do not care about the orientation of corners, you can still rotate three edges without using a commutator. So you can place the edges correctly. By rotating the last layer, you can obtain one of these situations:

all edges are already placed correctly
three edges need to be rotated
two pair of edges need to be swapped
The third one can be solved by rotating any three edges, and then you are back in situation 2.

To rotate edges, notice that when you take one solved pair out of its slot, you can move it out of the way by rotating the upper face and put the slot back to its position. Then, you can turn the upper face to replace it with the edge currently in the slot. When you move the slot out, you can bring back the pair without breaking it and put it back in it's correct place. For example, using the front-right-bottom slot, you can rotate the top-right edge, top-left edge and top-back edge like this : R U2 R' then U' then R U' R'. By doing this, you will rotate corners, so if they are alread correctly oriented, you need to use a conjugate of commutator to flip the edges without disturbing the corners. The commutator can be composed with A = "exchange an edge on the top face with an edge that is not on the top face, while keeping the rest of the top face unchanged" and B = "rotate the top face to put another edge in the intersection". The conjugation will used to place one of the three top edges at the location of the exchange. You can also compose the commutator with A = "swap two edges on the top face" and B = "rotate the edges so that one swapped edge will be swapped with a third one". In this case, you need to be able to rotate those edges, so they must be either on the same face, or on the same slice.
Solving corners
Finally, You will need commutators to solve the corners. You can orient corners by using a commutator like A = "rotate a corner without changing the rest of the top face" and B = "put another corner at the intersection". To swap corners, use the same principle used for the commutator of edges : a conjugate to move a corner out of the top face, and a commutator with A = "exchange one corner of the top face with the corner not on the top face" and B = "move another corner in the intersection". You can also rotate and orient corners at the same time by carefully choosing a conjugate and a commutator.

$Re : https : //www.speedsolving.com/wiki/index.php/Intuitive_solving$

3.algorithm 2 Edge first

Solve the cross

Solve 3 E layer edges

Orient and (then) permute the last layer edges while solving the last E layer edge. This can be done intuitively

Rotate the cube so you have at least 2 D layer corners in the D layer and on FUR another D layer corner

Use RUR'U' (Sexy Move) and D setup moves to solve 3 corners of the D layer. Be sure to use the unsolved corner to fix the edges that may have been affected by the sexy move (if it was not done 3 OR 6 times)

Turn the cube over so that the 3/4 solved layer is now on top and the unsolved corner is on FUR

Again use RUR'U' and D setup moves to solve the rest of the corners

Parity: If the D layer is solved but not the cube you have parity. Repeat sexy move untill the cube is solved exept for 2 corners

Hold the 2 Corners in D layer and solve them using RUR'U'

$Re : https : //www.speedsolving.com/wiki/index.php/Edges\_First$

4 The NP Class

4.1
It is a NP problem. First, with n nodes, the posibility could build max up to $n!$, which is not sure to be solved in polynomial time.

Secondly, if you have a hint (like a simple path is this question), it can be easily proved within polynomial time.

4.3
It's not limited to prove it in polynomial time, but if you have a hint like the right answer, all you need to do is to check all the sides surrounded by those nodes or not, then it belongs to polynomial time. So it is a NP question.

5 PRIMES is in P

5.1
The answer is no in this case.If you want verify it, you'll need trail division from $2, 3, \ldots, \sqrt{n}$;
For binary Turing maching, the input should be $\log(n)$, while the division complexity is $(\log n)^2$, while you have $\sqrt{n}$ inputs.
So for the computation, you'll need complexity $O((\log n)^2 * \sqrt{n})$, it's not polynomial time solvable compared to input.
From Prime Number Theorm we can know that $\lim_{n \to +\infty} \frac{\pi(n)}{n/\ln n} = 1$, honestly it

did't help much with the solve of this question.
In all, it's not effencienty to prove Prime is in P.