**ADA hw4 - 1** B08902065 資工二 洪易

Discuss with

B08902075 林耘平, B08902013 張永達, B08902127 林歆凱, B08902047 汪昊新, B08202029 楊欣翰, B08902063 陳羿穎

B08902135 隋中彧, B08902050 丁羿涵, B08902009 陳若婕

2. NP-Completeness & Reduction

(a)

We can arrange the array in order of [x1, x2, … xn – 1, xn, M, M’, x’1, x’2, … x’n – 1, x’n], as the following chart shown. As vertices in X only connect to two edges, we can see that (Orange part), , and + 1. Therefore, all vertices in X fulfill the constraint as every edge in graph has length of n + 1. For M and M’ (Blue part), we can see that the longest distance for them in array is from itself to the edge of the array, which is not greater than n – 1, also fulfill the constraints. For all vertices in X’, it’s similar to the case of X, , , and . All vertices in the array fulfill the constraints, thus this is a possible arrangement. Hence, it’s a solvable problem.

Next, let’s talk about what if M and M + 1 aren’t put in the n + 1th and n + 2th of the array. Then, the distance from the further edge (1 or 2n + 2) to where M or M + 1 is placed must be longer than n + 1, as all length of edges in graph are n + 1, and all of edges in X or X’ are connected to M and M’, at least one vertex in X or X’ will have a distance to M or M + 1 in array longer than n + 2. Thus, this answer will be invalid. As a result, M and M + 1 must be placed in n + 1th and n + 2th.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | 1 | 2 | … | n-1 | n | n+1 | n+2 | n+3 | n+4 | … | 2n+1 | 2n+2 |
| V | X1 | X2 | … | Xn-1 | Xn | M | M’ | X’1 | X’2 | .. | X’n-1 | X’n |

An example solution of LAP[G1].

(b)

By (a), we can see that M and M’ in both H1 and H2 should be placed in the n + 1th and n + 2th of H. Therefore, we can arrange the array as [x1, x2, … xn – 1, xn, Mx, Mx’, x’1, x’2, … x’n – 1, x’n, A, y1, y2, … yn – 1, yn, My, My’, y’1, y’2, … y’n – 1, y’n], also shown in the following chart. For 4 orange parts, it’s proved in (a) that they fulfill the constraints. For 2 blue parts, their distance to orange part is proved in (a), and their distance to A is smaller than n + 2 (|(2n + 3) – (n + 1)| or |(3n + 5) – (2n + 3)|), so they are also valid. Accordingly, for A (green part), it’s also valid. Thus, this arrangement is valid as all its element fulfill the constraints. This problem is solvable.

By (a), those blue blocks can’t be moved to another index, else it will be invalid. Next, if we move A to another block other than 2n + 3, then either or Mx or My’ will have distance to A longer than n + 2, which cause the whole chart to be invalid. Thus, A can only be placed in 2n + 3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | 1 | 2 | … | n-1 | n | n+1 | n+2 | n+3 | n+4 | … | 2n+1 | 2n+2 |
| V | X1 | X2 | … | Xn-1 | Xn | Mx | Mx’ | X’1 | X’2 | .. | X’n-1 | X’n |
| No. | 2n+3 | 2n+4 | … | 3n+2 | 3n+3 | 3n+4 | 3n+5 | 3n+6 | 3n+7 | … | 4n+4 | 4n+5 |
| V | A1 | Y1 | … | Yn-1 | Yn | My | My’ | Y’1 | Y’2 | .. | Y’n-1 | Y’n |

An example solution of LAP[G2]

(c)

First, let’s assume the copy of set X in H2 as set Y, and the copy of set X’ in H2 as set Y’. Add 2n edges that edges E(new) = , and their lengths are 2n + 5. Without loss of generality, assume all X vertices are put in PH1 and all X’ vertices are put in QH1. By adding those edges, we can see that if we put vertices in Y, take Yi for instance, into QH2, the distance in array of Yi and Xi is at least |(3n + 6) - n| = 2n + 6, which exceed the limit. Thus, all Y vertices must be placed in PH2. Accordingly, all Y’ vertices must be placed in QH2.

However, if we choose 2n + 6 as the vertices length, then we can have a vertex Yi that is in QH2, that connected to Xi in PH1, which make PH1 != PH2, and cause it invalid. As a result, the upper bound we can choose is 2n + 5. In addition, it’s trivial that the previous constraints are all fulfilled and the new added edges won’t violate them, this graph is still solvable.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | 1 | 2 | … | n-1 | n | n+1 | n+2 | n+3 | n+4 | … | 2n+1 | 2n+2 |
| V | X1 ~ Xn | | | | | Mx | Mx’ | X’1 ~ X’n | | | | |
| No. | 2n+3 | 2n+4 | … | 3n+2 | 3n+3 | 3n+4 | 3n+5 | 3n+6 | 3n+7 | … | 4n+4 | 4n+5 |
| V | A1 | Y1 ~ Yn | | | | My | My’ | Y’1 ~ Y’n | | | | |

(d)

Add three edges that connect three vertices in S’ to A which has length of n + 4. Therefore, if all three vertices are placed in P1, the farthest vertices of three to A must be longer than n + 4, therefore makes it invalid. Thus, one of those three must be move to Q1 in this case and fulfill the problems constraint. Let’s look at the following chart that represent the linear array. If we add three vertices from S’ to A that have length of n + 4, three vertices in S’ can only be placed in three yellow boxes or even more right, else there will be a vertex that has a distance of n + 5 to A and make it invalid. Thus, at least one vertex will be in both S’ and Q1. In addition, the previous constraints are all fulfilled and the new added edges won’t violate them, this graph is still solvable.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | 1 | … | n-2 | n-1 | n | n+1 | n+2 | n+3 | … | 2n+2 | 2n+3 |
| Distance from A | 2n+2 | … | n+5 | n+4 | n+3 | n+2 | n+1 | n | … | 1 | 0 |
| Vertex | P1 | | | | | Mx | M’x | Q1 | | | A |

(e)

Add total 2n edges with weight n+3 into the graph, 2 edges for each Ai, which connected to yi and xi on its left side. As the length of their new edges are n+3, for each pair xi and yi in Hi, there must exist at least one vertex to be placed in Qi. By problem (c), the vertex once appears in Q, it must also be placed in all other Q, but not in P. By problem (a), P and Q both have n vertices. According to these constraints, there will be at least n vertices be placed in Q (at least 1 for all i in {1, n}), however, at most n vertices can be placed in each Q, Therefore, n vertices will be placed in both P and Q, and all xi and yi must be placed by one in P and one in Q.

Take the following possible solution with n = 2 for example. I add four edges, which are (A1, X1), (A1, Y1), (A2, X2), (A2,Y2) as the fifth row shown. In this case, X1 and Y1 must have at least 1 in Q1, however, according to those edges in H2, X2 and Y2 must also have at least one be placed in Q1. Thus, I choose {Y1, Y2} to be placed in Q1 and {X1, X2} to be placed in P1. Accordingly, {X1, X2}, {Y1, Y2} will also be place in P2 and Q2, respectively. In addition, we can see that this solution will fulfill all previous constraints, and make this problem solvable.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 1 | n | n+1 | n+2 | n+3 | 2n+2 | 2n+3 | 2n+4 | 3n+3 | 3n+4 | 3n+5 | 3n+6 | 3n+7 | 4n+6 |
| Vertex | X2 | X1 | M1 | M1’ | Y1 | Y2 | A1 | X1 | X2 | M2 | M2’ | Y2 | Y2 | A2 |
| Vertices Group | P1 | | Q1 | | P2 | | Q2 | |
| H | H1 | | | | | | H2 | | | | | |
| Added Edges  (Connected to) |  | A1 |  |  | A1 |  | X1, Y1 |  | A2 |  |  | A2 |  | X2, Y2 |

(f)

Construct the graph with n + r + 1 copies of H. The first n copies of H have all edges in G5 from problem (e), and each copy in next r copies have all edges mentioned in G4 from problem (d). We can see that the first n partitions will make the whole n + r + 1 partitions consistent, as every pair of (xi, xi’) must be split into P and Q part (By (e)). If we take P as False (0) and Q as True (1), then we can make sure x and x’ aren’t both true or false, which is inconsistent. Next, the next r copies of H force to satisfy B. In every Hi, three vertices are chosen to connect to Ai with an edge of weight n + 4, corresponding to the three literals in a clause in 3-CNF-SAT. For example, if there exist a partition that at least one of three chosen vertices are in Q, that also indicates the corresponding clause is true.

For a solution to LAP [G], we can transform it into a corresponding solution of 3-CNF-SAT. We choose every vertex xi in P as false and xi in Q as true literals in the solution of 3-CNF-SAT. On the other hand, we can transform a solution of 3-CNF-SAT into a corresponding solution of LAP [G]. We put every literal that is true in 3-CNF-SAT in Q and others in P. In this case, we can show that if B is satisfiable, LAP [G] is solvable, and it’s also true on the other direction.

(g)

We need to check the reduction from LAP to 3-CNF-SAT and from 3-CNF-SAT to LAP are both in polynomial time. It’s known that 3-CNF-SAT is a NP-complete problem.

Let’s start it in order. In reduction from LAP to 3-CNF-SAT, we change a graph into a Boolean formula. First, we can see that r = O(n3), as there’s at most 2 \* n3types possibility of clause. Next, we go through r copies of H, which has O(n) vertices in it, and find three vertices that are connected to Ai with length n + 3, and change it into three literals in the corresponding clause, which is also in polynomial time. Therefore, the whole process of reducing LAP to 3-CNF-SAT is in polynomial time, this show that LAP is a NP problem.

Next, let’s look at the other case. In reduction from 3-CNF-SAT to LAP, we change a Boolean formula into a graph. First, build n copies of H with constraints in G5­, which takes O(n2) time, as there are O(n) edges in a H and we need to build n copies. Next, build r copies of H. Each of them also takes O(n) time, as only three additional edges are added into the copy, the rest amount of edges is still O(n). According to previous example, r = O(n3), therefore, all operations we do are still in polynomial time. We can state that the process of reducing 3-CNF-SAT to LAP is in polynomial time. This show that LAP is a NP-hard problem.

Because LAP is both NP and NP-hard, we can conclude that LAP is a NP-complete problem. QED