# HW8A

## Section II

Yanhao Wang 175002614

Zheng Qi 174003950

Mohan Xiao 174005865

## 7.23

The minimum vertex cover can be solved the same way. Solve for the maximum matching. Consider all of the vertices in the graph besides s and t. Call the set of these vertices that are an endpoint of an edge in the maximum matching A, and call the set of vertices that are not endpoints of any of the maximum matching edges B.

There can be no edges between distinct elements in B. Assume for contradiction that there was. That edge could then have been added to increase the size of the maximum matching, because it runs between two vertices that were untouched by the maximum matching edges. This contradicts our assumption that the matching found was the maximum matching, so it is impossible to have edges between distinct elements of B.

Therefore, all edges in the graph must either be included in the maximum matching or must be be- tween a vertex that is an endpoint of a maximum matching edge and a vertex in B. Note that it is also impossible for a maximum matching e dge to have both endpoints have edges to elements in B, otherwise the size of the matching could have been increased by removing the edge from the maximum matching and adding the two edges from its endpoints to the vertices in B. Therefore, at most one endpoint vertex of every maximum matching edge can have an edge to a vertex in B.

Therefore, when constructing the minimum vertex cover, it is sufficient to take every maximum matching edge and choose one of its endpoints for the minimum vertex cover. Specifically, if one of the endpoints has and edge to a vertex in B, choose that vertex. Otherwise, it doesnt matter which one you choose.

Because an endpoint was selected from every edge in the maximum matching, then the vertex cover must cover all of the maximum matching edges.

Furthermore, if a vertex had an edge to an element in B, then that vertex was chosen. There can be no edges between elements in B, as we proved earlier.

Therefore, the vertex cover chosen in this way must cover every edge in the graph. Furthermore, it must be minimal, as removing any of the vertices would mean that neither endpoint vertex of one of the maximum matching edges was in the set of vertices, because only one endpoint vertex was chosen per maximum matching edge. This means that the resultant set of vertices would not cover every edge in the graph. Therefore, if we can solve maximum matching, we can solve minimum vertex cover. Since maximum matching is solved by max flow, then minimum vertex cover reduces to max flow.

## 8.1

Let T be the sum of all edge weights in the graph G. The minimum tour is at least 0, and also bounded by T. We first query TSP with $b = T/2$. If the query returns no, that means the minimum tour lies between T/2 and T. If yes, it lies between 0 and T/2. We can continue the routine and narrow the value of the true weight sums of the minimum tour, and call TSP to return the final minimum tour. This procedure calls TSP at most $log(T)$ times, and since T can be represented in at most $|E|logK$ bits , assuming $K$ is the largest weight in the graph, $log(T)$ is polynomial regarding of the input size. Thus we will only make polynomial number of calls. So if TSP can be solved in polynomial time, then so can TSP-OPT.

## 8.2

We can assume that the graph have Rudrata road and remove the edges one by one to find it. If after remove an edge from the current graph,the graph still has a Rudrata path, we remove the edge permanently and update the current graph. If the new graph does not have a Rudrata path, the we can recognize that this edge is one of the Rudrata path. Hence, we keep the status of the graph that the graph always has a Rudrata path (if the given graph did). When it is impossible to remove any edges except a single Rudrata path, we will find a Rudrata path in the end.

## 8.3

STINGY SAT can be verified a true assignment in polynomia time, So it's NP.

If there is a SAT with k variables, it can be transformed to STINGY SAT, which only has k variables and k of STINGY SAT is k.

- If the SAT with k variables has a satisfying assignment, the corresponding STINGY SAT have a satisfiying assignment with true variables number smaller than k, because of the maximum number of variables is k.
- If the SAT with k variables has no satisfying assignment, the STINGY SAT has no satisfying assignment.

Because SAT is NP-Complete, the STINGY SAT is NP-Complete.

## 8.4

- a. The Clique-3 can be verified in polynomia time so it's NP.

  Get a graph and a set of vertex, we can verify in polynomia time whether these vertices are a clique. If these vertices are clique, we can judge wheter these vertices all have in-dgree lower of equal to 3 in linear time.

  So it's NP.

- b. The reduction should be clique to clique-3, not clique-3 to clique.

- c. Given a graph A->B->C and b = 1, B is a vertex cover of size b, but A and C absolutely ont a clique. So vertex cover can not reduct a clique.
- d. If in-dgree of all vertices in G is not bigger than 3, the clique cannot have more than 4 vertices. So find all the possible combination of 4 vertices. and check if it's a clique. This algorithm has time complexity $O(|V|^4)$.

## 8.6

(a)

If each literal in 3SAT appears at most once, then we can catagorize into 5 different cases of any variable a:

- If $a$ only appears in clause c, then set a = true and ignore c.
- If $\neg a$ only apperas in c, then set $a$ = false and ignore c.
- If $\neg a$ and $a$ only appears in c, then $a$ can either be true or false, and we can also ignore c.
- If $\neg a$ and $a$ apperas in two different clauses, for intance, $(a \bigcup b \bigcup c)$ and $(\neg a \bigcup d \bigcup e)$, we can take out variable $a$ and combine the rest and set the value of $a$ in the future, if cannot be sovled.
- If $a$ and $\neg a$ are the only variables in two clauses, then it is not satisfiable.

Thus, applying above approach, we can eliminate each variable in linear time and thus the problem can be solved in polynomial time.

(b). If we look back to the process of reducing 3SAT to Independent Set, if each literal at most appears 2 times, then the degree of any vertices will not be larger than 4. So since 3SAT problem remains NP-Complete when literals appear at most twice, then Indepent Set also remains NP-Complete when all the nodes have degree at most 4.

## 8.10

a) We can thinking this problem like a CLIQUE problem that we can use (G,k) as a CLIQUE problem, and H have V vertices and each of the vertices have a connected edge. Then we can get H which must be the subgraph of G. So this problem can become a NPC problem just like CLIQUE problem.

b)This is a generalization of Rudrata Path problem. Take some graph $G = (V, E)$. The Rudrata Path problem takes this graph G and finds some simple path of length g=|V|,

c） This is a generalization of SAT problem. We can just make g to become the size of clause,then we can just transfer this problem into SAT(the size of SAT is g) . Since SAT is NP-complete, Max SAT is also NP-complete.

d)This also a generalization of CLIQUE problem. So we can use a CLIQUE problem that (G, a) and b = a(a − 1)/2. Any vertices in the G must at least have a edge connected each pair which is also a clause. So because CLIQUE problem is a NPC, this problem should also be NPC.

e)This is a generalization of INDEPENDENT set. we can have a instance of Independent set(G,k),we can let k = a, b = 0. Then we can get (G, a, b) which is an instance for sparse subgraph, which is a Independent set problem when b = 0. So because the Independent set is a NPC. this problem can also be NPC

f)Obviously, it can generalizes VERTEX COVER. we can think that each vertex as a set can have a edge upon it. Then we can translate is to vertex cover question. Because Vertex cover question is NPC, so the Set cover is also NPC.

g) Rudrata cycle. Given a graph(V,E), set b equals to n and $d_{ij}$ equals to 1 for all (i,j) in E and other edges $d_{ij}$ to 2. And all $r_{ij}$ to 2.

Then we can pick some edges which sum up of weight is n and between every points i and j, there must be 2 vertex-disjoint paths, which is a cycle. Because of the weights is n, the graph can at most have n edges. So, it must have at least one cycle contains all the vertices and n edges. This is a Rudrata cycle.

## 8.16

To show the transformation from 3SAT to EXPERIMENTAL CUISINE, first, given a clause ($U \bigcup V \bigcup W$), we can generate 7 corresonding ingredients, $\{\neg UV\neg W, U\neg V\neg W, \neg U\neg VW, UV\neg W, \neg UVW, U\neg VW, UVW\}$, which are the 7 cases that satisfy the clause ($U \bigcup V \bigcup W$). Set the discord value between two variable in the same clause to 1. And set the contradiction variable in two clauses's discord value to one. Assume we have n clauses. And set p =0, and if we can find n number of ingredients with penalty<=p, then it means that 3SAT is satisfiable. And since verifying if 3SAT is satisfiable is equal to finding the satisfiable assignment. Thus if EXPREIMENTAL CUISINE is solvable in polynomial time, then 3SAT is solvable in polynomial time.