ShanghaiTech University

CS240 Algorithm Design and Analysis

Spring 2023

Problem Set 4

Due: 23:59, April 30, 2023

1. Submit your solutions to the course Blackboard.
2. If you want to submit a handwritten version, scan it clearly.
3. Late homeworks submitted within 24 hours of the due date will be marked down 25%. Homeworks submitted more than 24 hours after the due date will not be accepted unless there is a valid reason, such as a medical or family emergency.
4. You are required to follow ShanghaiTech's academic honesty policies. You are allowed to discuss problems with other students, but you must write up your solutions by yourselves. You are not allowed to copy materials from other students or from online or published resources. Violating academic honesty can result in serious penalties.

Note: When proving that a problem A is NP-complete, clearly divide your answer into three steps:

1. Prove that A is in NP.
2. Choose an NP-complete problem B. For any instance of B, construct an instance of problem A, so that the yes/no answers to the two instances are the same.
3. Show your construction runs in polynomial time.

Problem 1:

Suppose you are the manager of a clothing store which sells j different jackets and h different shirts. c customers come to the store, and each of them has a set of jackets and shirts which they like. Each customer will either buy one jacket and one shirt from the set he likes, or nothing at all. The customer is satisfied if he buys one jacket and one shirt. In addition, each jacket or shirt can be sold to at most one customer. Design an efficient algorithm which maximizes the number of satisfied customers.

Solution:

One approach to solve this problem is to use a bipartite graph. We can represent the customers on one side of the graph and the jackets and shirts on the other side. An edge between a customer and a jacket (or a shirt) means that the customer likes that jacket (or shirt).

The problem of finding the maximum number of satisfied customers is equivalent to finding the maximum matching in this bipartite graph. A matching is a set of edges in the graph such that no two edges share a common vertex. In our case, a matching represents a set of customers, jackets, and shirts such that each customer is satisfied and each jacket and shirt is sold to at most one customer.

We can use the Hopcroft-Karp algorithm to find the maximum matching in the bipartite graph. The algorithm has a time complexity of $O(\sqrt{V}E)$, where V is the number of vertices in the graph (in our case, c+j+h) and E is the number of edges (in our case, at most c×(j+h)).

Here is the pseudocode for the algorithm:

```
initialize an empty bipartite graph G
add c customers to the left side of G
add j jackets and h shirts to the right side of G
add an edge between a customer and a jacket (or shirt) if the
    customer likes the jacket (or shirt)

initialize an empty matching M
while there exists an augmenting path P in G:
    find an augmenting path P using breadth-first search
    update the matching M using the augmenting path P

output the number of edges in M (which is the maximum number of
    satisfied customers)
```

The algorithm first initializes an empty bipartite graph G and adds the customers, jackets, and shirts to the appropriate sides of the graph. Then it adds an edge between a customer and a jacket (or shirt) if the customer likes the jacket (or shirt).

Next, it initializes an empty matching M. Then it repeatedly finds augmenting paths in the graph and updates the matching M using those paths. An augmenting path is a path in the graph that starts and ends on unmatched vertices and alternates between edges in the matching M and edges not in M.

Finally, the algorithm outputs the number of edges in the matching M, which is the maximum number of satisfied customers. Note that the algorithm assumes that each customer likes at most one jacket and one shirt. If a customer can like multiple jackets or shirts, we can represent each jacket or shirt as a separate node in the bipartite graph and connect each customer to all the jackets and shirts he likes. The rest of the algorithm remains the same.

Problem 2:

Consider a directed graph $G = (V, E)$. Let $X \subseteq V$ be a set of *start nodes*, and $Y \subseteq V$ be a set of *destination nodes*, and assume that X and Y are disjoint. Your goal is to find a set of paths, where each path starts at a node in X and ends at a node in Y, and none of the paths share an edge. Design an efficient algorithm to determine if such a set of paths exist, and find such a set if possible.

Solution:

Algorithm:

1. Initialize an empty set P to store the set of paths.
2. For each node $x \in X$, perform a breadth-first search starting at x until reaching a node $y \in Y$. During the search, maintain a set S of visited nodes and a predecessor function $p(v)$ which records the node that leads to v. If y is reached, add the path from x to y to P, constructed by following the predecessor function from y to x.
3. If P contains $\mid X \mid$paths, return P.
4. Otherwise, return that such a set of paths does not exist.

Proof of Correctness:

Suppose there exists a set of paths P^*that satisfies the conditions in the problem statement. We claim that the algorithm will find P^*.

For each node$x \in X$, there exists a path inP^*that starts at$x$. By construction, the algorithm will find a path from $x$to a node$y \in Y$for each $x \in X$. Furthermore, since none of the paths in P^*share an edge, none of the breadth-first searches starting from different nodes in X an revisit the same node. Therefore, each path in P^* will be found exactly once and added to P.

Conversely, suppose the algorithm finds a set of paths P. We claim that P satisfies the conditions in the problem statement.

Since P contains exactly one path for each node in X, and each path starts at a node in X, it follows that each node in X is the start of exactly one path in P. Similarly, each path in P ends at a node in Y. Therefore, P satisfies the condition that each path starts at a node in X and ends at a node in Y.

Suppose for the sake of contradiction that two paths p_1 and p_2 in P share an edge. Let x_1 and x_2 be the start nodes of p_1 and p_2 , respectively. Without loss of generality, assume that x_1 is discovered before x_2 in the algorithm. When the breadth-first search starting at x_1 is performed, the algorithm discovers the

edge shared by p_1 and p_2 while searching for the path ending at y_1.However, since p_1 and p_2 share an edge, the breadth-first search starting at $x_2$will also discover the same edge while searching for the path ending at y_2. Therefore, the algorithm will not add p_1 and p_2 to P, which contradicts the assumption that P contains all paths that satisfy the conditions in the problem statement. Therefore,P satisfies the condition that none of the paths share an edge.

Time Complexity:

The algorithm performs a breadth-first search for each node in X, which takes $O(\mid V \mid +E)$ time. Since there are $\mid X \mid$ nodes in X, the total time complexity of the algorithm is $O(\mid X \mid (\mid V \mid +E))$.

Problem 3:

Let C be a finite set and let S be a collection of subsets of C. The Set Packing problem asks whether, for an input value k, there exist k sets from S which are pairwise disjoint (i.e. no two sets share an element). Show that Set Packing is NP-complete, using a reduction from the Independent Set problem.

Solution:

To show that Set Packing is NP-complete, we follow the three steps outlined in the note:

1. Prove that Set Packing is in NP:

Given a set of k sets $S_1, S_2, \ldots, S_k$ as a certificate, we can easily verify in polynomial time whether they are pairwise disjoint.

2. Choose an NP-complete problem B:

We choose the Independent Set problem, which is known to be NP-complete.

The Independent Set problem is defined as follows: given an undirected graph $G = (V, E)$ and an integer k, does there exist a set $I \subseteq V$ such that $|I| \geq k$ and no two vertices in I are adjacent in G?

3. Construct a polynomial time reduction from Independent Set to Set Packing:

Given an instance of the Independent Set problem (G, k), we construct an instance of the Set Packing problem as follows:

Let C be the set of vertices in G, and let S be the set of all cliques in G. For each $v \in V$, let S_v be the set of all cliques in G that contain v. Note that each set S_v can be constructed in polynomial time.

We claim that there exists a set of k vertices in G that form an independent set if and only if there exists a set of k cliques in G that are pairwise disjoint.

($\Rightarrow$) Suppose there exists a set $I \subseteq V$ of k vertices that form an independent set. For each vertex $v \in I$, choose a clique C_v in G that contains v. Since I is an independent set, no two vertices in I are adjacent in G, which means that no two cliques C_u and C_v that correspond to vertices u and v in I share an edge. Thus, the set $\{C_u, C_v, \ldots, C_w\}$ of k cliques is pairwise disjoint.

($\Leftarrow$) Suppose there exists a set $\{C_1, C_2, \ldots, C_k\}$ of k cliques that are pairwise disjoint. For each clique C_i, choose a vertex $v_i \in C_i$. Since the cliques are pairwise disjoint, the vertices $v_1, v_2, \ldots, v_k$ are pairwise non-adjacent in G, which means that they form an independent set.

It remains to show that the reduction can be done in polynomial time. Constructing the set C and the set S of cliques can be done in polynomial time. For each vertex $v \in V$, we can construct the set S_v of cliques that contain v in

polynomial time by checking each clique in S to see if it contains v. Therefore, the entire reduction can be done in polynomial time.

Since we have shown that Set Packing is in NP and that there is a polynomial time reduction from Independent Set to Set Packing, we conclude that Set Packing is NP-complete.

Problem 4:

Suppose you are organizing a summer sports camp which offers n different sports, and you need hire camp counselors who are skilled at these sports. You receive applications from m people, where each person is skilled at some subset of the sports. For an input value $k \leq m$, you want to determine whether it is possible to hire k people, so that for each sport you hire at least one person skilled in that sport. Show that this problem is NP-complete, using a reduction from the Vertex Cover problem.

Solution:

To prove that the summer sports camp problem is NP-complete, we need to follow the three steps outlined in the note:

1. Prove that the problem is in NP:

To prove that the summer sports camp problem is in NP, we need to show that given a potential solution (i.e. a set of k people), we can efficiently verify whether the solution is correct or not. To do this, we can check whether each sport has at least one person in the solution who is skilled at that sport. This verification can be done in polynomial time, so the problem is in NP.

2. Choose an NP-complete problem B and construct an instance of the summer sports camp problem:

We will choose the Vertex Cover problem as B. Given an instance of Vertex Cover, which is an undirected graph $G = (V, E)$ and a positive integer k, the question is whether there exists a vertex cover of size at most k. A vertex cover is a set of vertices such that every edge in the graph has at least one endpoint in the set. We will construct an instance of the summer sports camp problem as follows:

For each vertex v_i in G, create a sport s_i. For each edge $(v_i, v_j) \in E$, create a pair $(p_{i,j}, p_{j,i})$ of people who are skilled at the sports s_i and s_j respectively. The set of all such pairs is denoted as P. Let k be the given parameter for the Vertex Cover problem.

3. Show that the construction runs in polynomial time:

The construction takes polynomial time because it involves creating a set of pairs for each edge in the graph, which can be done in polynomial time.

We now need to show that the yes/no answers to the two instances are the same, i.e. there exists a vertex cover of size at most k in G if and only if it is possible to hire k people for the summer sports camp.

($\Rightarrow$) Suppose there exists a vertex cover V' of size at most k in G. For each vertex $v_i \in V'$, we can select one of the people in the pairs $(p_{i,j}, p_{j,i})$ for each

edge $(v_i, v_j) \in E$. Since V' is a vertex cover, every edge in the graph has at least one endpoint in V', so we can select at least one person from each pair without any conflicts. Therefore, we can hire k people for the summer sports camp such that for each sport, there is at least one person skilled in that sport.

($\Leftarrow$) Suppose it is possible to hire k people for the summer sports camp such that for each sport, there is at least one person skilled in that sport. Let S be the set of sports for which we have selected at least one person. Then $|S| \leq k$, since we have selected k people. For each vertex v_i in G, if $s_i \in S$, we include v_i in the vertex cover V'. Since we have selected at least one person skilled in each sport, every edge in the graph has at least one endpoint in V', so V' is a vertex cover of size at most k in G.

Therefore, the summer sports camp problem is NP-complete.

Problem 5:

SIST wants to hire graduate students to TA courses. Each course needs exactly two TAs, and there is a list of pairs of students who can TA a course together (i.e. if two students are not a pair from the list, they cannot TA a course together). For a given input value k, we want to determine whether there exists a *TA cycle* with k distinct students $s_1, ..., s_k$, such that s_1 and s_2 TA a course together, s_2 and s_3 TA a course together, ..., and s_k and s_1 TA a course together (note that each student in the cycle will TA two courses). Show that this problem is NP-complete.

Solution:

We can show that the TA Cycle problem is NP-complete by reducing from the Hamiltonian Cycle problem.

First, we prove that the TA Cycle problem is in NP. Given a set of k distinct students, we can verify in polynomial time whether there exists a TA cycle with these students, simply by checking whether each student is paired with the correct student in the cycle and whether the cycle is closed.

Next, we choose the NP-complete Hamiltonian Cycle problem as our starting point. Given an instance of Hamiltonian Cycle, which is a graph $G = (V, E)$, we construct an instance of the TA Cycle problem as follows:

For each vertex $v_i \in V$, we create a student s_i who is skilled in the sport associated with v_i.

For each edge $(v_i, v_j) \in E$, we create a pair of students (s_i, s_j) who can TA a course together.

We set $k = |V|$.

We claim that there exists a Hamiltonian cycle in G if and only if there exists a TA cycle with k distinct students.

($\Rightarrow$) Suppose G has a Hamiltonian cycle $C = (v_{i_1}, v_{i_2}, \ldots, v_{i_n}, v_{i_1})$. We can construct a TA cycle with k distinct students as follows: s_{i_1} is paired with s_{i_2}, s_{i_2} is paired with s_{i_3}, and so on, until s_{i_n} is paired with s_{i_1}. Since C is a Hamiltonian cycle, each vertex in G appears exactly once in C, and thus each student in our constructed TA cycle is paired with exactly one other student. Furthermore, since C is a cycle, the TA cycle is also closed. Therefore, the yes/no answers to the Hamiltonian Cycle instance and the TA Cycle instance are the same.

($\Leftarrow$) Suppose there exists a TA cycle with k distinct students. We can construct a Hamiltonian cycle in G as follows: let $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$ be the distinct students in the TA cycle, in the order in which they appear in the cycle. Then

the Hamiltonian cycle C is $(v_{i_1}, v_{i_2}, \ldots, v_{i_k}, v_{i_1})$. Since each student in the TA cycle is paired with a student skilled in the sport associated with the next vertex in the cycle, the corresponding vertices in G are connected by edges. Therefore, C is a Hamiltonian cycle in G. Again, since the yes/no answers to the Hamiltonian Cycle instance and the TA Cycle instance are the same, this completes the reduction.

Finally, we need to show that the reduction runs in polynomial time. It is clear that we can construct the instance of the TA Cycle problem in polynomial time from the instance of Hamiltonian Cycle. Therefore, the TA Cycle problem is NP-complete.

Problem 6:

In the Knapsack problem, we have n items, and the j'th item has weight w_j and value v_j $(j = 1, \ldots, n)$. All w_j, v_j values are positive integers. The question is whether there exists a subset of the n items with total weight at most W and total value at least V, for input values W and V. Show that Knapsack is NP-complete.

Solution:

1. Prove that Knapsack is in NP:

Given a set of items with their weights and values, a subset of items with a total weight at most W and total value at least V can be verified in polynomial time by checking that the total weight of the subset is at most W and the total value of the subset is at least V. Therefore, Knapsack is in NP.

2. Choose an NP-complete problem B. For any instance of B, construct an instance of problem Knapsack, so that the yes/no answers to the two instances are the same.

We will use the Subset Sum problem as the NP-complete problem B. The Subset Sum problem is defined as follows: Given a set of n positive integers $A = \{a_1, a_2, ..., a_n\}$ and a target value T, determine if there exists a subset $S \subseteq A$ such that the sum of elements in S is equal to T.

Given an instance of the Subset Sum problem with set A and target value T, we construct an instance of Knapsack as follows:

Let $W = T$ and for each $a_i \in A$, let $v_i = w_i = a_i$. Then, a subset $S \subseteq A$ with a sum of elements equal to T exists if and only if there exists a subset of items with a total weight at most W and total value at least V.

Proof:

Suppose there exists a subset $S \subseteq A$ such that the sum of elements in S is equal to T. Then, we can select the items corresponding to the elements in S in the Knapsack instance, and their total weight and value are both equal to T, which satisfies the constraints of the Knapsack instance.

Conversely, suppose there exists a subset of items with a total weight at most W and total value at least V. Let S be the subset of A corresponding to the selected items. Then, the sum of elements in S is equal to the total value of the selected items, which is at least V. Since the total weight of the selected items is at most $W = T$, we have that the sum of elements in S is at most T. Therefore, S is a subset of A such that the sum of elements in S is equal to T, which satisfies the constraints of the Subset Sum instance.

3. Show that the construction runs in polynomial time.

The construction of the Knapsack instance from the Subset Sum instance can be done in polynomial time, since we only need to create an instance with the same number of items as the Subset Sum instance, and each item can be created in constant time. Therefore, the reduction is polynomial-time.

Since Subset Sum is NP-complete, and we have shown that Knapsack is polynomial-time reducible to Subset Sum, Knapsack is also NP-complete.