Lab04-Matroid

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2021.

$*$ If there is any problem, please contact TA Haolin Zhou.

$*$ Name: Yanjie Ze   Student ID:519021910706   Email:zeyanjie@sjtu.edu.cn

1. *Property of Matroid.*

   (a) Consider an arbitrary undirected graph $G = (V, E)$. Let us define $M_G = (S, C)$ where $S = E$ and $C = \{I \subseteq E \mid (V, E\backslash I) \text{ is connected}\}$. Prove that M_G is a **matroid**.

   Proof.
   Hereditary: Suppose $A \subset B, B \in C$.

   B satisfies that $(V, E\backslash B)$ is connected.

   $A \subset B$, so $(V, E\backslash A)$ is connected too.

   Therefore, $A \in C$.

   Exchange Property: Denote two subsets $A, B \in C$, $|A| < |B|$. Assume $\{e_1, e_2, ..., e_k\} = B/A, k \geq 1$. Denote $G_A = (V, E\backslash A), G_B = (V, E\backslash B)$, and they are connected.
   Since $|A| < |B|$, there's at least one cycle in G_A. What's more, G_B must have removed some edges(at least one) that are in a cycle of G_A, from $\{e_1, e_2, ..., e_k\}$, as shown in Fig. 1(1). If the edges from $\{e_1, e_2, ..., e_k\}$ are not in a cycle of G_A, which is to say, G_B removes the cut edge in G_A, then G_A must have done the same thing, and this will lead to $|A| = |B|$, as shown in Fig 1(2).
   So we can select the edge with red cross in Fig 1(1), denoting as e. $A \cup \{e\}$ satisfies that $(V, E\backslash(A \cup \{e\}))$ is connected.

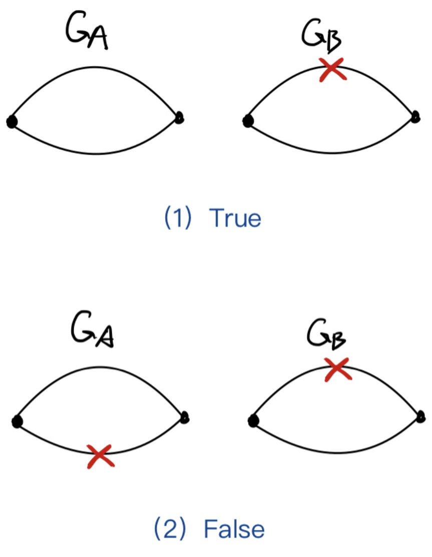

Figure 1: Graph Matroid Exchange Property

□

(b) Given a set A containing n real numbers, and you are allowed to choose k numbers from A. The bigger the sum of the chosen numbers is, the better. What is your algorithm to choose? Prove its correctness using **matroid**.

Remark: Denote $\mathbf{C}$ be the collection of all subsets of A that contains no more than k elements. Try to prove $(A, \mathbf{C})$ is a matroid.

Solution.
The greedy algorithm is optimal in this problem, shown in Alg. 1.

Algorithm 1: Greedy Choice

```
1 Sort all numbers in $A$ into ordering $x_1 \geq x_2 \geq ... \geq x_n$;
2 $S \leftarrow \emptyset$;
3 for $i = 1$ to $k$ do
4     $S \cup \{x_i\}$;
5 return $S$;
```

$\square$

Proof.
Denote $\mathbf{C}$ be the collection of all subsets of A that contains no more than k elements.We try to prove $(A, \mathbf{C})$ is a matroid.

Hereditary: Denote $A \subset B \in \mathbf{C}$. B contains no more than k elements. So A also contains no more than k elements.

Exchange Property: Denote $A, B \in C$, $|A| < |B|$. Assume $x \in B, x \notin A$.

Since $|A| < |B| \leq k$, $|A \cup \{x\}| \leq k$. Therefore, $A \cup \{x\} \in C$.

Thus, $(A, \mathbf{C})$ is a matroid.

Lemma: If $(S, \mathbf{C}, c)$ is a weighted matroid, then Greedy-MAX algorithm performs the optimal solution.

In this problem, cost function $c(x_i) = x_i$, meaning the larger the number, the more weights it has. And we have proved that $(A, \mathbf{C})$ is a matroid. Therefore, Alg. 1 is optimal.

$\square$

2. *Unit-time Task Scheduling Problem.* Consider the instance of the **Unit-time Task Scheduling Problem** given in class.

(a) Each penalty ω_i is replaced by $80 - \omega_i$. The modified instance is given in Tab. 1. Give the final schedule and the optimal penalty of the new instance using Greedy-MAX.

Table 1: Task

a_i	1	2	3	4	5	6	7
d_i	4	2	4	3	1	4	6
ω_i	10	20	30	40	50	60	70

Solution.
Let S be a set of unit-time tasks with deadlines and $\mathbf{C}$ the set of all independent tasks

of S. Then $(S, \mathbf{C})$ is matroid and Alg. 2 Greedy-MAX is optimal.

Algorithm 2: Greedy-MAX

```
1 Sort all penalties into ordering $w_1 \geq w_2 \geq ... \geq w_n$;
2 $S \leftarrow \emptyset$;
3 for $i = 1$ to $n$ do
4     if $S \cup \{w_i\} \in \mathbf{C}$;
5      then
6         $S \leftarrow S \cup \{x_i\}$;
7 return $S$;
```

For the tasks in table. 1, we can start to select from $i = 7$. Greedy-MAX selects a_7, a_6, a_5, a_4, a_3, then rejects a_2, a_1.

The final schedule is $< a_5, a_4, a_6, a_3, a_7 >$, which is shown in Fig. 2.

The optimal penalty is $w_2 + w_1 = 30$.

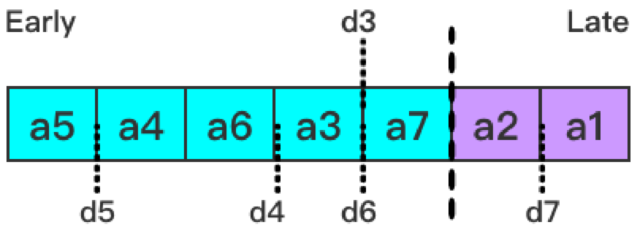

Figure 2: Task Scheduling

□

(b) Show how to determine in time $O(|A|)$ whether or not a given set A of tasks is independent. (**Hint**: You can use the lemma of equivalence given in class)

Solution.
Lemma: For any set of tasks A, the two statements are equivalent.

i. The set A is independent.
ii. For $t = 0, 1, 2, ..., n$, $N_t(A) \leq t$.

Based on the lemma, we are able to design an algorithm to utilize the property of $N_t(A)$, shown in Alg. 3.

Algorithm 3: Independent Judging

```
Input: task set $A$ ;
Define array $a[|A|]$;
Define array $N[|A|]$;
Define $sum := 0$;
for $i = 1$ to $|A|$ do
    $a[d_i] + +$;
for $i = 1$ to $|A|$ do
    $sum+ = a[i]$;
    $N[i] := sum$;
    if $N[i] > i$ then
        return False
return True
```

Explanation of the algorithm:

First, we define an array a to record the deadlines. It has the size of $|A|$, because the maximum of the deadline is not more than $|A|$. $a[i]$ means the number of tasks with the deadline i.

Second, we define an array N to function as $N_t(A)$ in the lemma. $N[i]$ means the number of tasks in A whose deadline is i or earlier. This array is actually not necessary and we can only use *sum*, but this array helps understand the algorithm and does not increase the upper bound of the time complexity.

Third, we will explain the loops. The first loop is to count the appearances of deadlines. The second loop is proposed to calculate the cumulative deadlines.

Time complexity: $O(|A|)$

The two loop both have a $O(|A|)$ time complexity.

Space complexity: $O(|A|)$

To count deadlines, we need at least one array a, with size equal to $|A|$. The array N is not necessary as mentioned before and it does not effect the complexity. Therefore, the overall space complexity is $O(|A|)$.

□

3. *MAX-3DM.* Let X, Y, Z be three sets. We say two triples (x_1, y_1, z_1) and (x_2, y_2, z_2) in $X \times Y \times Z$ are *disjoint* if $x_1 \neq x_2$, $y_1 \neq y_2$, and $z_1 \neq z_2$. Consider the following problem:

Definition 1 (MAX-3DM)**.** *Given three disjoint sets X, Y, Z and a non-negative weight function $c(\cdot)$ on all triples in $X \times Y \times Z$,* ***Maximum 3-Dimensional Matching*** *(MAX-3DM) is to find a collection $\mathcal{F}$ of disjoint triples with maximum total weight.*

(a) Let $D = X \times Y \times Z$. Define independent sets for MAX-3DM.

Solution.
Define $\mathbf{H} = \{F \subseteq D | \forall (x_1, y_1, z_1), (x_2, y_2, z_2) \in F, \textit{they are disjoint.}\}$

Hereditary: Denote $A \subset B, B \in \mathbf{H}$. Since triples in B are all disjoint, triples in A are disjoint. Thus $A \in \mathbf{H}$.

$(D, \mathbf{H})$ is an **independent system**. Each subset F in $\mathbf{H}$ is **independent**. □

(b) Write a greedy algorithm based on Greedy-MAX in the form of *pseudo code*.

Solution.
Greedy-MAX has been mentioned in Alg. 2. Greedy-MAX combined with 3DM problem is shown in Alg. 4, called Greedy-MAX-3DM. The basic ideas in them are the same.

Algorithm 4: Greedy-MAX-3DM

```
1 Sort all weights into ordering: $c(x_1, y_1, z_1) \geq c(x_2, y_2, z_2) \geq ... \geq c(x_n, y_n, z_n)$;
2 $\mathcal{F} \leftarrow \emptyset$;
3 for $i = 1$ to $n$ do
4     if $\mathcal{F} \cup \{(x_i, y_i, z_i)\} \in \mathbf{H}$ then
5         $\mathcal{F} \leftarrow \mathcal{F} \cup \{(x_i, y_i, z_i)\}$;
6 return $\mathcal{F}$;
```

□

(c) Give a counter-example to show that your Greedy-MAX algorithm in Q. 3b is not optimal.

Solution.
The instance setting is shown in table. 2.

Greedy-MAX-3DM: $\mathcal{F} = \{(0,0,0), (3,3,3)\}, weight = 5 + 2 = 7$

Optimal: $\mathcal{F} = \{(1,0,1), (2,2,0), (3,3,3)\}, weight = 4 + 3 + 2 = 9$

Table 2: Task

x_i	y_i	z_i	$weight$
0	0	0	5
1	0	1	4
2	2	0	3
3	3	3	2

□

(d) Show that: $\max\limits_{F \subseteq D} \frac{v(F)}{u(F)} \leq 3$. (Hint: you may need Theorem 1 for this subquestion.)

Solution.
We construct 3 matroids and construct the independent system based on the 3 matroids.

Define $\mathbf{S} = \{F \subseteq D | \forall (x_1, y_1, z_1), (x_2, y_2, z_2) \in F, x_1 \neq x_2\}$

Define $\mathbf{R} = \{F \subseteq D | \forall (x_1, y_1, z_1), (x_2, y_2, z_2) \in F, y_1 \neq y_2\}$

Define $\mathbf{T} = \{F \subseteq D | \forall (x_1, y_1, z_1), (x_2, y_2, z_2) \in F, z_1 \neq z_2\}$

We prove that $(D, \mathbf{S})$, $(D, \mathbf{R})$, $(D, \mathbf{T})$ are all matroids.

Proof.
For $(D, \mathbf{S})$:

Hereditary: Denote $A \subset B \in \mathbf{S}, |A| < |B|$.

Because: $\forall (x_1, y_1, z_1), (x_2, y_2, z_2) \in B, x_1 \neq x_2$.

Thus: $\forall (x_1, y_1, z_1), (x_2, y_2, z_2) \in A, x_1 \neq x_2$.

Therefore, $A \in \mathbf{S}$.

Exchange Property: Denote two subsets $A, B \in \mathbf{S}, |A| < |B|$.
Since $|A| < |B|$, there must exist $(x, y, z) \in B$ which satisfies $\forall (x_i, y_i, z_i) \in A, x_i \neq x$.
Therefore, $A \cup \{(x, y, z)\} \in \mathbf{S}$.
Finally we prove that $(D, \mathbf{S})$ is a matroid.

Similarly, we can use the same way to prove $(D, \mathbf{R})$, $(D, \mathbf{T})$ are all matroids.

Now we have constructed three matroids, and naturally we want to use them to construct the target independent system:
$\mathbf{S} \cap \mathbf{R} \cap \mathbf{T} = \{F \in D | \forall (x_1, y_1, z_1), (x_2, y_2, z_2) \in F, x_1 \neq x_2, y_1 \neq y_2, x_3 \neq y_3.\} = \mathbf{H}$
From Theorem 1, we can induce that $\max\limits_{F \subseteq D} \frac{v(F)}{u(F)} \leq 3$. $\square$

$\square$

Theorem 1. *Suppose an independent system $(E, \mathcal{I})$ is the intersection of k matroids $(E, \mathcal{I}_i)$, $1 \leq i \leq k$; that is, $\mathcal{I} = \bigcap_{i=1}^{k} \mathcal{I}_i$. Then $\max\limits_{F \subseteq E} \frac{v(F)}{u(F)} \leq k$, where $v(F)$ is the maximum size of independent subset in F and $u(F)$ is the minimum size of maximal independent subset in F.*

Remark: You need to include your .pdf and .tex files in your uploaded .rar or .zip file.