

Lab04-Matroid

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2020.

* If there is any problem, please contact TA Yiming Liu.

* Name: Yijia Diao Student ID: 518030910146 Email: diao_yijia@sjtu.edu.cn

1. Give a directed graph $G = (V, E)$ whose edges have integer weights. Let $w(e)$ be the weight of edge $e \in E$. We are also given a constraint $f(u) \geq 0$ on the out-degree of each node $u \in V$. Our goal is to find a subset of edges with maximal weight, whose out-degree at any node is no greater than the constraint.

- (a) Please define independent sets and prove that they form a matroid.
- (b) Write an optimal greedy algorithm based on Greedy-MAX in the form of *pseudo code*.
- (c) Analyze the time complexity of your algorithm.

Solution. (a)

Definition 1 (Independent Sub-graph). G' is a sub-graph of G . G' is independent if $\forall V' \in G' : \text{out-degree} \leq f(u)$.

Let \mathbf{C} be the set of all independent sub-graph of G . Now proof (G, \mathbf{C}) is matroid.

Proof. (Hereditary): $\forall B \in \mathbf{C}, \forall A \subseteq B$, the out-degree of $A \leq$ that of B . So $\forall V \in A : \text{out-degree} \leq f(u)$, which means $A \in \mathbf{C}$.

(Exchange Property): Consider two independent graph A and B with $|A| < |B|$. Now proof by Contradiction.

Suppose that $\forall e \in B \setminus A, A \cup \{e\}$ is not an independent graph. Therefore, $\forall e \in B$ must be connected with those $v \in A : \text{out-degree of } v = f(u)$, and are out-edge of v . Let k be the number of v above, so $|A| \geq kf(n), |B| \leq kf(n) \Rightarrow |A| \geq |B|$, contradiction.

So we can conclude that (G, \mathbf{C}) is matroid.

(b)

Algorithm 1: Greedy-MAX for Maximal Weight of Edge

Input: $G = (V, E), w(e), f(u)$.

Output: A subset of edges E' with maximal weight, whose out-degree at any node $\leq f(u)$.

```
1 Sort all  $e \in E$  by  $w(e)$  non-decreasingly;
2  $E' \leftarrow \emptyset, \text{sumweight} \leftarrow 0$ ;
3 for all  $e \in E$  do
4    $G' = (V', E' \cup \{e\})$ ;
5   if  $\forall u \in V' : u \leq f(u)$  then
6      $E' \leftarrow E' \cup \{e\}, \text{sumweight} \leftarrow \text{sumweight} + w(e)$ ;
7 return  $E'$ ;
```

- (c) Suppose we use two adjacent vertex to denote an edge, and $|E| = n$. The time complexity of sorting is $O(n \log n)$. For line 5, we can use Red-Black tree to optimize, so its time complexity is $O(\log n)$; thus the time complexity of the **For** loop is $O(n \log n)$.

So we can conclude that the time complexity of Alg.1 is $O(n \log n)$.

□

2. 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 2 (MAX-3DM). Given three disjoint sets X, Y, Z and a nonnegative 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.
- (b) Write a greedy algorithm based on Greedy-MAX in the form of *pseudo code*.
- (c) Give a counterexample to show that your Greedy-MAX algorithm in Q. 2b is not optimal.
- (d) Show that: $\max_{F \subseteq D} \frac{v(F)}{u(F)} \leq 3$. (Hint: you may need Theorem 1 for this subquestion.)

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 $\max_{F \subseteq E} \frac{v(F)}{u(F)} \leq k$, 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 .

Solution. (a)

Definition 3 (Independent Triple Collection). A set S is independent iff. $S \subseteq D$ and $\forall (x_i, y_i, z_i), (x_j, y_j, z_j) \in S$ are disjoint ($i \neq j$).

(b)

Algorithm 2: Greedy-MAX for MAX-3DM

Input: $D, c(\cdot)$.

Output: A collection \mathcal{F}' of disjoint triples.

```

1 Sort all  $(x, y, z) \in D$  by  $c((x, y, z))$  non-decreasingly;
2  $\mathcal{F}' \leftarrow \emptyset, \text{sumweight} \leftarrow 0$ ;
3 for all  $(x, y, z) \in D$  do
4   if  $\mathcal{F}' \cup \{(x, y, z)\}$  is an independent set then
5      $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(x, y, z)\}, \text{sumweight} \leftarrow \text{sumweight} + c((x, y, z))$ ;
6 return  $\mathcal{F}'$ ;
```

(c) $X = Y = Z = \{1, 2\}$, $D = X \times Y \times Z$.

$$c((x, y, z)) = \begin{cases} 9 & (x, y, z) = (1, 1, 1) \\ 1 & (x, y, z) = (2, 2, 2) \\ 8 & \text{otherwise} \end{cases}$$

The result of Alg.2 is $\mathcal{F}' = \{(1, 1, 1), (2, 2, 2)\}$ and the weight is 10, but one of the optimal solution is $\mathcal{F} = \{(1, 2, 1), (2, 1, 2)\}$, weight= 16.

(d) **Proof.** Define $D_x \in D$: any two triples in D_x has different x ; $D_y \in D$: any two triples in D_y has different y ; $D_z \in D$: any two triples in D_z has different z .

Define $\mathcal{C}_1, \mathcal{C}_2$ and \mathcal{C}_3 are the collection of D_x, D_y and D_z ; and define \mathcal{C} is the collection of independent subset of D . So we have: $\mathcal{C} = \bigcap_{i=1}^3 \mathcal{C}_i$.

Now proof $(D, \mathcal{C}_1), (D, \mathcal{C}_2)$ and (D, \mathcal{C}_3) are matroids. Since $\mathcal{C}_1, \mathcal{C}_2$ and \mathcal{C}_3 are equivalent, we only consider \mathcal{C}_1 .

Hereditary: $\forall B \subseteq D_x, \forall A \subset B$, since $\forall (x_i, y, z), (x_j, y, z) \in B (i \neq j) : x_i \neq x_j$, we have the same property for A . So $A \in \mathcal{C}_x$.

Exchange Property: (by contradiction) $\forall A, B \in \mathcal{C}_1, |A| < |B|$, suppose $\forall (x, y, z) \in B \setminus A, A \cup \{(x, y, z)\} \notin \mathcal{C}_1$, which equals $\forall (x, y, z) \in B, \exists (x_a, y_a, z_a) \in A : x = x_a$. And since A and B are subsets of D_x , their cardinality equals to the number of different x . So we have $|B| \leq |A|$, contradiction.

Therefore we can conclude that (D, \mathcal{C}_1) , (D, \mathcal{C}_2) and (D, \mathcal{C}_3) are matroids.
 According to Thm.1, we have $\max_{F \subseteq D} \frac{v(F)}{u(F)} \leq 3$.

□

3. **Crowdsourcing** is the process of obtaining needed services, ideas, or content by soliciting contributions from a large group of people, especially an online community. Suppose you want to form a team to complete a crowdsourcing task, and there are n individuals to choose from. Each person p_i can contribute v_i ($v_i > 0$) to the team, but he/she can only work with up to c_i other people. Now it is up to you to choose a certain group of people and maximize their total contributions ($\sum_i v_i$).

- (a) Given $\text{OPT}(i, b, c)$ = maximum contributions when choosing from $\{p_1, p_2, \dots, p_i\}$ with b persons from $\{p_{i+1}, p_{i+2}, \dots, p_n\}$ already on board and at most c seats left before any of the existing team members gets uncomfortable. Describe the optimal substructure as we did in class and write a recurrence for $\text{OPT}(i, b, c)$.
- (b) Design an algorithm to form your team using dynamic programming, in the form of *pseudo code*.
- (c) Analyze the time and space complexities of your design.

Solution. (a) **Optimal substructure:**

Case 1: OPT selects p_i .

- collect contribution v_i ,
- calculate the minimal people that could be added,
- must include OPT in $\{p_1, p_2, \dots, p_{i-1}\}$

Case 2: OPT does not select p_i

- must include OPT in $\{p_1, p_2, \dots, p_{i-1}\}$

Recurrence function:

$$\text{OPT}(i, b, c) = \begin{cases} 0 & c = 0 \text{ or } i = 0 \\ \text{OPT}(i - 1, b, c) & c_i < b \\ \max\{v_i + \text{OPT}(i - 1, b + 1, \text{minc}), \text{OPT}(i - 1, b, c)\} & \text{otherwise} \end{cases}$$

in which $\text{minc} = \min\{c - 1, c_i - b\}$.

(b)

Algorithm 3: Crowdsourcing

Input: $n, c_i, v_i (1 \leq i \leq n)$.

Output: maximized $\sum_i v_i$

```

1 Initialize  $\text{OPT}[i, b, c]$  as 3-dimensional array  $[n, n, n]$ ;
2  $\text{OPT}[0, b, c] \leftarrow 0, \text{OPT}[i, b, 0] \leftarrow 0$ ;
3 for  $i = 1$  to  $n$  do
4   for  $b = n$  to  $c_i + 1$  do
5     for  $c = n$  to  $1$  do  $\text{OPT}[i, b, c] \leftarrow \text{OPT}[i - 1, b, c]$ ;
6   for  $b = c_i$  to  $0$  do
7     for  $c = n$  to  $1$  do
8        $\text{minc} \leftarrow \min\{c - 1, c_i\}$ ;
9        $\text{OPT}[i, b, c] \leftarrow \max\{v_i + \text{OPT}[i - 1, b + 1, \text{minc}], \text{OPT}[i - 1, b, c]\}$ 
10 return  $\text{OPT}[n, 0, n]$ ;
```

- (c) Time complexity: Since all operations in For-loop are $O(1)$, the time complexity of Alg.3 is $O(n^3)$.
Space complexity: $O(n^3)$

□

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