CS535 Design and Analysis of Algorithms - Assignment 3

Batkhishig Dulamsurankhor - A20543498

October 7, 2024

1. To show that $(S, \mathcal{I})$ is a matroid, it must satisfy the following properties.
   - First, it has to be a finite and non-empty set. It is trivial and non-empty set S already satisfies it.
   - Second is a heredity property. Let's say some two columns of I are dependent on each other. It is clear that those two columns are also a subset of I, in other words $I' \subseteq I$, where I' is set of the two columns. So, by heredity property, if $I \in \mathcal{I}$, then $I' \in \mathcal{I}$ must also satifsy, which is a contradiction since we know that I' is not an independent set and $I' \notin \mathcal{I}$. Therefore, to satisfy the second property, all columns in I must be linearly independent.
   - Let's show that exchange property must satisfy with the similar approach to the previous one. We have a set $|B| > 2$ such that $B \in \mathcal{I}$, but some column b_i, b_j are dependent on each other. Let's imagine that we use heredity property and now we have $B' = \{b_i, b_j\}$ which supposed to be $B' \in \mathcal{I}$. We know that $|B| > |B'|$ and we can't find an element b_k in B that can make B' part of the independent set. In other words, $\forall b_k \in B - B' : B' \cup \{b_k\} \notin \mathcal{I}$, since B' itself contains linearly dependent columns and adding any element can change the fact that it is dependent. This contradicts the property that $\exists b_k \in B - B' : B' \cup \{b_k\} \in \mathcal{I}$ and therefore all columns in B must be linearly independent.

2. (a) Curcuits in the matric matroid are the minimal dependent subsets of columns of matrix A. Curcuit C is set of columns that are linearly dependent, but every possible proper subset of C is linearly independent.

   (b) Suppose $C_1 \neq C_2$, then $C_1 \subset C_2$ or C_1 must be a proper subset of C_2. We know that both C_1 and C_2 are dependent sets. On the other hand, it is defined that removal of any element in C must result in an independent set. So, a proper subset of C_2 results in an independent set, which makes C_1 an independent set. This leads to a contradiction since a set can't be dependent and independent at the same time. Thus, if $C_1 \subseteq C_2$, then $C_1 = C_2$.

   For example, in a graph matroid, minimal dependent subset is a cycle $C = (V, E)$ in the graph. In a cycle, the number vertices and the number of edges must be equal, $|V| = |E|$. Removing any edge will result in a tree which is an independent set, $|V| = |E| - 1$. If a cycle C_1 is a subset of a cycle C_2, the only possible option is that $C_1 = C_2$. Because removing an edge from a cycle will never result in another cycle, but a tree.

3. As mentioned in the problem, jobs do not have start and finish time. So the order of execution doesn't matter as long as we choose jobs that maximize the profit. The algorithm has an issue and will not always yield the optimal result.

   Our goal is to maximize the profit, so at a glance we have to run jobs that gives the most profit per time frame. According to the algorithm, we calculate p_i/t_i for all i jobs and sort them in descending order, which gives us ordered jobs from the most profitable to the least profitable per time frame. By greedily choosing from the most profitable job until we can't fit anymore, we can seem to maximize the profit.

   However, the algorithm sometimes fail to achieve the maximum profit since it can look over more profitable combinations by greedily choosing the most profitable job first. Let's disprove the correctness of this algorithm with an example. Suppose we had machine with span $T = 9$ and 4 jobs with profit $p = \{10, 21, 8, 14\}$ and required time $t = \{2, 7, 2, 7\}$.

- Calculating p_i/t_i gives the following rates: $r = \{5, 3, 4, 2\}$.
- Sorting them by r in descending order gives: $job_1, job_3, job_2, job4$.
- Now we greedily choose select the jobs with above order until we can no longer fit within the timeframe of $T = 9$.
- We choose job_1 and the total profit becomes: $P = p_1 = 10$, and time left: $T_l = T - t_1 = 9 - 2 = 7$.
- Then we choose job_3: $P = p_1 + p_3 = 10 + 8 = 18$, $T_l = T - t_1 - t_3 = 9 - 2 - 2 = 5$.
- Finally we have only $T_l = 5$ left so we can't run anymore jobs since $t_2 = 7, t_4 = 7$.
- The final answer is $P = 18$.

But, we can clearly see that maximum profit we can make is $P = 31$ by choosing job_1, job_2. They have total required time of $t_1 + t_2 = 9$ which fits in $T = 9$. So, this greedy algorithm doesn't result in optimal solution.

4.

References

[1] master.pdf.

[2] Fibonacci sequence, *Wikipedia*, `https://en.wikipedia.org/wiki/Fibonacci_sequence#Relation_to_the_golden_ratio`