AuW-pg05-bf

Der Weihnachtsmann möchte m Haushalte mit Geschenken beliefern. Seine Fabrik am Nordpol hat dazu n verschiedene Geschenke hergestellt. Das i-te Geschenke wurde dabei g_i-mal produziert. Dabei soll der j-te Haushalt mit h_j vielen Geschenken beliefert werden. Der Weihnachtsmann fragt sich nun, ob eine Zuteilung der Geschenke auf die verschiedenen Haushalte möglich ist, sodass kein Haushalt zweimal mit dem gleichen Geschenk beliefert wird.

Definieren Sie ein Netzwerk $N = (V, A, c, s, t)$, mit dessen Hilfe die Lösbarkeit des gegebenen Zuweisungsproblems durch Bestimmung eines maximalen Flusses entschieden werden kann. Die Anwendung eines Algorithmus zur Bestimmung eines maximalen Flusses in diesem Netzwerk soll entweder eine mögliche Aufteilung der Geschenke auf die Haushalte liefern, oder einen Beweis, dass eine solche Aufteilung nicht möglich ist. Zeigen Sie, dass das von Ihnen definierte Netzwerk diese Bedingungen erfüllt.

We recall the definition for a network is a tuple $N = (V, A, c, s, t)$, such that

$$\begin{array}{ll} G = (V, A) & \text{"directed graph"} \\ s \in V & \text{"source"} \\ t \in V \setminus \{s\} & \text{"target"} \\ c : A \to \mathbb{R}_0^+ & \text{"capacity function"} \end{array}$$

Let us define these elements

$$\begin{aligned} s &= \text{"super source"} \\ t &= \text{"super sink"} \end{aligned}$$

$$\begin{aligned} \mathbb{P} &= \{p_i \mid i \in [1, n]\} \\ \mathbb{K} &= \{k_j \mid j \in]n, m]\} \end{aligned}$$

$$V = \{s, t, p \in \mathbb{P}, k \in \mathbb{K}\}$$

$$\begin{aligned} X &= \{s \times \mathbb{P}\} \\ Y &= \{\mathbb{P} \times \mathbb{K}\} \\ Z &= \{\mathbb{K} \times t\} \end{aligned}$$

$$A = X \cup Y \cup Z$$

$$c(a) = \begin{cases} g_i, & a \in X \\ 1, & a \in Y \\ h_j, & a \in Z \end{cases}$$

Thus our network $N = (V, A, c, s, t)$ solves the problem, since, either a distribution of presents can be achieved, in which case the maximum flow at t must be equal to the sum of all h_j and the types of presents for each household are those, that flow into it (the capacity of 1 ensures that no present can be given to a house twice), or it's impossible to distribute the presents, in which case the maximum flow at t is not equal to the sum of all h_j.

The following algorithm solves the problem, given a described network N and returns a distribution of presents for it or, if not possible `false`.

```
-- @input
-- N = {V, A, c, s, t}
-- H = sum(h[1, ... m])
--
-- returns a boolean,
-- indicating whether the tour
-- is possible and, if yes,
-- what presents each kid gets
function deliveryPossible(N, H)
        possible = H == N.computeMaximumFlow(s, t)
        presents = getPresents(N)
        return possible and presents
end

function getPresents(N)
        presents = {}
        for j=1, m do
                presents[j] = {}
                for i=1, n do
                        presents[j][i] = N.getFlow(j)
                end
        end
        return presents
end

function computeMaximumFlow(s, t)
        -- runs Ford Fulkerson on network N
        -- returns maximum flow from s to t
end

function getFlow(v)
        -- returns the flow at vertex v
end
```

The runtime is $O(|E| \cdot f)$, as the Ford-Fulkerson algorithm which dominates the rest runs in that time.
□