Given a set of days $D = \{[s_1, e_1, w_1], [s_2, e_2, w_2], ..., [s_n, e_n, w_n]\}$ such that s_i is the start of the i^{th} day, e_i is the sum of the i^{th} day and the period of time one must wait after collecting lollies on that day, hereafter refered to as the end of the i^{th} day, and w_i is the number of lollies to be collected on the i^{th} day, hereafter refered to as the weight of the i^{th} day, we want to find a subset $S \subseteq D$ such that:

$d_1 \cap d_2 = \phi$, for $\forall d_1, d_2 \in S$, and
$\sum_{d \in S} d.w$ is maximized.

Let $D \neq \phi$ be given, then optimal solution $S \subseteq D$ must contain some days because $w_i \geq 0, \forall i$. Consider some $s \in S$. Then all days in S $\{s\}$ are either left of s or right of s since no two days in S overlap. Thus let

$L(s) = \{d \in D : d \cap s = \phi$ and d is left of $s\}$
$R(s) = \{d \in D : d \cap s = \phi$ and d is right of $s\}$

Which gives the sets $S \cap L(s)$, $S \cap R(s)$, and $\{s\}$ which partitions S. Now we prove that $S \cap L(s)$ is an optimal solution to the problem restricted to $D \cap L(s)$. Consider another optimal selection of days S' of $L(s)$ such that

$\sum_{d \in S'} d.w > \sum_{e \in L(s)} e.w$

Then since $S' \cap (\{s\} \cup R(s)) = \phi$ by construction we could have constructed S as $S' \cup R(s) \cup \{s\}$ which implies

$\sum_{d1 \in S'} d1.w + \sum_{d2 \in R(s) \cup \{s\}} d2.w > \sum_{e1 \in L(s)} e1.w + \sum_{e2 \in R(s) \cup \{s\}} e2.w$
$\sum_{d1 \in S'} d1.w + \sum_{d2 \in R(s) \cup \{s\}} d2.w > \sum_{e \in L(s) \cup R(s) \cup \{s\}} e.w$
$\sum_{d1 \in S'} d1.w + \sum_{d2 \in R(s) \cup \{s\}} d2.w > \sum_{e \in S} e.w$

Which contradicts our assumption that S is an optimal solution so no such $S' \neq L(s)$ exists. Similar logic applies to show $R(s)$ is optimal for D as well.

However since we don't know s we will narrow down our choice of s so that s is the rightmost element of S so that $R(s) = \phi$. This will allow us to optimize our solution since we know that if $L(s)$ is optimal then $L(s) \cup \{s\}$ is optimal for all days left of s and s. Therefore we will be able to construct

our solution by considering $\forall d \in D$ in increasing order of s_i.

Suppose that $|D| = n$ then we start by constructing an $n+1$ by 2 matrix max such that $max[i][1] = s_i, max[i][2] = 0$. Another row can be created to track the days choosen. We also initialize $max[n+1][1] = \infty, max[n+1][2] = 0$. We then construct our matrix values starting at $max[2][2]$, since no day ends when it starts, by considering all days that end at or before $max[i][1]$ and after $max[i-1][1]$. Here we notice that sorting D by the end point leads to faster search times. We use $max[n+1][1] = \infty$ so that we consider $\forall d \in D : d.e \in (max[n][1], \infty)$ because $max[n][1]$ is the start of the last day and at least one day, the last day, will end after this time. Thus we end up with the recurrance relations

$$max[i] = max(\{max[j] + d_k.w : d \in D, d.e \in (max[i-1][1], max[i][1]], max[j] = d.s\} \cup \{max[i-1]\})$$

We note that an inequality would work on $max[j] <= d.s$, and we can be circumspect of some partially invalid data this way, however we do so in the implementation, not in the design. We also note that there are two ways to create the algorithm, one of which is $O(nlogn)$ and the other is $O(n^2)$. The algorithm presented with this document uses the later as it is easier. The only difference is how data is indexed and looked up. Either way we construct the max matrix along it's second row from $i = 1$ to $n + 1$.