SOLUTION

The problem is solved using dynamic programming. Let M(n, state1, state2) be the largest amount of coal that can be produced after n−1 food shipments have already been distributed; state1 describes which shipments have gone to mine 1 so far, and state2 describes which shipments have gone to mine 1 so far. Furthermore, let value(a, b) denote the amount of coal produced when food of type b arrives to a mine in state a. The following recursive formula holds:

```
M(n, state1, state2) = max {
        M(n+1, newstate1, state2) + value(state1, type of shipment n),
        M(n+1, state1, newstate2) + value(state2, type of shipment n)
}
```

One could write a simple recursion based on the above formula (calculate M(0, empty, empty)). Such an algorithm has $O(2^N)$ complexity and would score 45 points.

Note that, for calculating the value of M, it suffices to describe the state of a mine by the last two shipments only. This is because any shipment, other than the last two, cannot influence the amount of coal produced. The total number of different states is thus reduced to only 3*3+1=10 per mine (3 types of food in each of the last two shipments, and a special state describing an empty mine).

This allows us to drastically reduce the total number of configurations: N shipments × 10 states (for the first mine) × 10 states (for the second mine). For each of the configurations, we can calculate the value of the configuration once and store it for reuse.

However, with N as high as 100 000, using so much memory (100N integers) for storing the values of the configurations would break the memory limit and score between 70 and 85 points.

To get the full score, we note that, when calculating M(n, *, *), we only use M(n+1, *, *). If we calculate M in decreasing order of n, we need only 2·100 integers.