Exercise 3

Computation of Group SALES By key

Because we don't compute any aggregation inside the group By, I think the query will return the same number of rows than the relation SALES. It is then useleless to use the Advanced hash algorithms for aggregation (last question in exercise of homework 3).
We instead use the simple hash (previous question) and we have seen that the I/O cost is 3N – 2N/k with k = (B-sqrt(B^2 – 4N)/2.

Use of the CUBE function

The CUBE function will compute the 2^3 different group By queries : ALL, A1, A2, A3, (A1,A2), (A1,A3), (A2,A3), (A1,A2,A3).

Maximal Cost

The maximal cost will then be 8*(3N – 2N/k)

Roll-Up (or drill down) strategy for grouping

We can minimize the above cost by implementing a smarter group by depending of the strategy we took.
Suppose we took the Roll up strategy. After aggregating on A1, the group (A1,A2) can be compute in a smarter way without using 3N-2N/K I/Os (we can store groups on disk on the same pages and pay only 1 I/O to read a group)