```cpp
#include <vector>
#include <queue>
#include <algorithm>
#include <iostream>

using namespace std;

#define MAXK 10000
#define MAXM 100000
#define MAXN 10000

queue <int> coda; // Coda usata per la BFS
vector <int> grafo[MAXN+1]; // Struttura per il grafo
bool visitato[MAXN+1];  // Array che segna quando un vertice è già stato visitato con la BFS
int dist[MAXN+1]; // Array dove vengono salvate le distanze nella BFS
int distPartenza[MAXN+1]; // Array per la distanza parziale, salvata per la prima BFS




// BFS che salva le distanze sul vettore dist

void BFS(int start) {
    coda.push(start);
    visitato[start]=1;
    while (!coda.empty()) {
        int vert=coda.front();
        coda.pop();
        for (unsigned int i=0; i<grafo[vert].size(); i++) {
            if (!visitato[grafo[vert][i]]) {
                coda.push(grafo[vert][i]);
                dist[grafo[vert][i]]=dist[vert]+1;
                visitato[grafo[vert][i]]=1;
            }
        }
    }
}



int compra(int N, int M, int K, int supermercati[], int da[], int a[]) {

    // Salvo il grafo utilizzando le liste di adiacenza
    for (int i=0; i<M; i++) {
        grafo[da[i]].push_back(a[i]);
        grafo[a[i]].push_back(da[i]);
    }

    BFS(1); // BFS per trovare la distanza minima di ogni vertice dalla partenza

    // Azzero i vettori dist e visitato per poter ripetere di nuovo la BFS
    for (int i=1; i<=N; i++) {
        distPartenza[i]+=dist[i];
        dist[i]=0;
        visitato[i]=0;
    }

    BFS(N); // BFS per trovare la distanza minima di ogni vertice dalla casa della nonna


    // Cerco la somma minima tra dist e distPartenza tra i supermercati
    int res=MAXM;
    for (int i=0; i<K; i++) res=min(res,dist[supermercati[i]]+distPartenza[supermercati[i]]);

    return res;
}
```