```
int vertex_num = get_vertex_num();
int max_degree = get_max_degree();

// array for saving upper bound of vertex core number
ub = new int[vertex_num];
// array for saving count number of vertex
cnt = new int[vertex_num];

// an array for loading neighbors for each vertex
int* nbr = new int[max_degree];

// an array for computing the core number of each vertex
int* nbrCnt = new int[max_degree+1];

// initialize array ub and cnt by vertex degree and 0 respectively
for (int u = 0; u < vertex_num; ++u){
        ub[u] = get_vertex_degree(u);
}
memset(cnt,0,sizeof(int)*vertex_num);

bool update = true;
while(update){
        update = false;
        for (int u = 0; u < vertex_num; ++u){
                if(cnt[u]>=ub[u]){
                        continue;
                }

                int originUb = ub[u];

                // get neighbors of vertex u from hard disk
                // the neighbor id of u will is put into array "nbr"
                // the degree of u is put into variable "degree"
                int degree;
                loadNbr(u,nbr,degree);

                // get the core distribution for neighbors' contribution

                memset(nbrCnt,0,sizeof(int)*(originUb+1));
                for (int j = 0; j < degree; ++j){
                    int v = nbr[j];
                    ++nbrCnt[ub[v]<ub[u]?ub[v]:ub[u]];
                }

                // calculate new ub and new cnt
                cnt[u] = 0;
                for (int i = originUb; i > 0; --i){
                    cnt[u] += nbrCnt[i];
                    if(cnt[u] >= i){
                        ub[u] = i;
                        break;
                    }
                }

                // update the cnt of neighbors if necessary
                if(ub[u]<originUb){
                    update = true;
                    for (int i = 0; i < degree; ++i){
                        v = nbr[i];
                        if(ub[v]>ub[u] && ub[v]<= originUb && cnt[v] >= ub[v]){
                            --cnt[v];
                        }
                    }
                }
        }
}

delete[] nbrCnt;
delete[] nbr;

// When the algorithm is finished, the values in array ub is the core numbers for vertices.
```