```
SET kruskal(EDGE[] A, int n, int m)
    // EDGE[]:  vettore di archi
(1) SET T ← Set // insieme (inizialmente vuoto) che conterrà gli archi dell'albero minimo
    MFSET M ← Mfset(n) // insieme disgiunto grande

    // ordino per peso crescente
(2) { ordina A[1...m] in modo che A[1].peso ⩽ ··· ⩽ A[m].peso }
    int c ← 0 // quanti archi ho aggiunto
    int i ← 1 // quale arco sto guardando
(3) finché c < n − 1 and i ⩽ m fai // Termina quando l'albero è costruito
        // c < n − 1:  ho raggiunto tutti gli archi necessari per fare un albero
        // i ⩽ m:  ho esaurito tutti gli archi da guardare (per controllo)
        se M.find(A[i].u) ≠ M.find(A[i].v) allora // non fanno parte dello stesso albero
            M.merge(A[i].u, A[i].v)  // unisco gli insiemi disgiunti
            T.insert(A[i])  // inserisco l'arco all'albero
            c ← c + 1  // ho aggiunto un altro arco
        i ← i + 1  // guardo il prossimo arco
    ritorna T  // Ritorna l'albero di copertura minimo
```