

```

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
        { ordina A[1][m] in modo che A[1].peso ≤ ... ≤ A[m].peso }

(2)   int c ← 0 // quanti archi ho aggiunto
        int i ← 1 // quale arco sto guardando
        while c < n - 1 and i ≤ m do // 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)
            if M.find(A[i].u) ≠ M.find(A[i].v) then // 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
        return T // Ritorna l'albero di copertura minima

```