ALBERO DI CONNESSIONE MINIMO IN GRAFI SPARSI

- DATO UN GRAFO $G=(V,E)$, CON FUNZIONE PESO c, LA PROCEDURA MST-REDUCE (COMPLESSITA' $O(E)$) COSTRUISCE
  - UN GRAFO CONTRATTO $G'=(V',E')$ (CON $|V'| \leq |V|/2$) CON FUNZIONE PESO $c': E' \to \mathbb{R}$
  - UN INSIEME DI ARCHI $T \subseteq E$
  - UN'APPLICAZIONE $orig': E' \to E$

TALE CHE SE A E' UN MST DI (G', c') ALLORA $T \cup \{orig'[x,y] : (x,y) \in A\}$ E' UN MST DI (G,c)

```
MST-REDUCE(G, orig, w, T)
for each v ∈ V[G] do
    mark[v] ← FALSE
    MAKE-SET(v)
for each u ∈ V[G] do
    if mark[u] = FALSE then
        choose v ∈ Adj[u] such that w[u, v] is minimized
        UNION(u, v)
        T ← T ∪ {orig[u, v]}
        mark[u] ← mark[v] ← TRUE
V[G′] ← {FIND-SET(v) : v ∈ V[G]}
E[G′] ← Ø
for each (x, y) ∈ E[G] do
    u ← FIND-SET(x)
    v ← FIND-SET(y)
    if u ≠ v then
        if (u, v) ∉ E[G′] then
            E[G′] ← E[G′] ∪ {(u, v)}
            orig′[u, v] ← orig[x, y]
            w′[u, v] ← w[x, y]
        else if w[x, y] < w′[u, v] then
            orig′[u, v] ← orig[x, y]
            w′[u, v] ← w[x, y]
construct adjacency lists Adj for G′
return G′, orig′, w′, and T
```

```
MST-REDUCE(G, orig, c, T)
for each v ∈ V[G] do
    mark[v] ← FALSE
    MAKE-SET(v)                 SET[v] := v
for each u ∈ V[G] do
    if mark[u] = FALSE then
        choose v ∈ Adj[u] such that c[u, v] is minimized
        UNION(u, v)             SET[u] := SET[v]
        T ← T ∪ {orig[u, v]}
        mark[u] ← mark[v] ← TRUE
V[G′] ← {FIND-SET(v) : v ∈ V[G]}
E[G′] ← Ø
for each (x, y) ∈ E[G] do
    u ← FIND-SET(x)
    v ← FIND-SET(y)
    if (u, v) ∉ E[G′] then
        E[G′] ← E[G′] ∪ {(u, v)}
        orig′[u, v] ← orig[x, y]
        c′[u, v] ← c[x, y]
    else if c[x, y] < c′[u, v] then
        orig′[u, v] ← orig[x, y]
        c′[u, v] ← c[x, y]
construct adjacency lists Adj for G′
return G′, orig′, c′, and T
```

Per la raccolta degli archi del grafo contratto, si utilizza un array B di dimensione $|V'|$, inizializzato a $-\infty$ su tutte le sue componenti.

Per ciascuna componente C
- Per ciascun vertice $v \in C$
  - Per ciascun vertice $u \in Adj[v]$
    - Se Find_Set[v] ≠ Find_Set[u], si verifica se $c(u,v)$ è il miglior candidato incontrato sinora per rappresentare l'arco da Find_set[v] a Find_set[u] (utilizzando opportunamente l'array B, per la ricerca in tempo $O(1)$ dell'eventuale candidato precedente)

Alla fine della fase riguardante C, si reinizializza B utilizzando una lista delle locazioni usate di B

- SI APPLICHI MST-REDUCE k VOLTE DI SEGUITO, ACCUMULANDO GLI ARCHI IN T, E SIA $(G', c', orig')$ IL GRAFO PESATO RISULTANTE, CON RELATIVA MAPPA $orig'$ (COMPLESSITA' $O(kE)$)

- QUINDI SI APPLICHI AL GRAFO (G', c') L'ALGORITMO DI PRIM (COMPLESSITA' $O(E' + V' \lg V')$)

- UTILIZZANDO LA MAPPA $orig'$ E' POSSIBILE RICOSTRUIRE UN MST PER (G, c)

- SI OSSERVI CHE $|V'| \leq \frac{|V|}{2^k}$, $|E'| \leq |E|$

- PER $k = \Theta(\log\log V)$, SI HA
  - $O(kE) = O(E \log\log V)$
  - $O(E' + V' \log V') = O(E + V' \log V) = O\left(E + \frac{V}{2^{\log\log V}} \log V\right)$

    $= O\left(E + \frac{V}{\log V} \log V\right) = O(E + V) = O(E)$

E QUINDI IL TEMPO COMPLESSIVO E' $O(E \log\log V)$

- PER QUALI VALORI DI $|E|$ (IN FUNZIONE DI $|V|$) L'ALGORITMO DI PRIM CON PRE-ELABORAZIONE E' ASINTOTICAMENTE PIÙ EFFICIENTE DELL'ALGORITMO DI PRIM (SENZA PRE-ELABORAZIONE) ?

- VOGLIAMO CHE $E \lg \lg V = o(E + V \lg V)$.

MA: $E \lg \lg V = o(E + V \lg V)$

$$\iff \frac{E + V \lg V}{E \lg \lg V} \to +\infty$$

$$\iff \frac{E}{E \lg \lg V} + \frac{V \lg V}{E \lg \lg V} \to +\infty$$

$$\iff \frac{V \lg V}{E \lg \lg V} \to +\infty$$

$$\iff \boxed{E = o\left(V \frac{\lg V}{\lg \lg V}\right)}$$