Capitolo 2.4-2.5-2.6ish

Lorenzo Vaccarecci

25 Marzo 2024

1 Algoritmo di Prim

Minimo albero ricoprente Dato un grafo G connesso, non orientato e pesato, un minimo albero ricoprente di G è un albero ricoprente di G in cui la somma dei pesi degli archi è minima.

Un minimo albero ricoprente di G è quindi un sottografo di G tale che:

- sia un albero libero, ossia connesso e aciclico
- contenga tutti i nodi di G
- la somma dei pesi degli archi sia minima

Simile a Dijkstra, ma si prende ogni volta, fra tutti i nodi adiacenti a quelli per cui si è già trovato il minimo (neri), quello connesso a un nodo nero dall'arco di costo minimo, cioè si cerca il nodo "più vicino" all'albero già costruito (poi, come in Dijkstra, si aggiornano gli altri nodi).

```
Prim(G,s)
    for each (u nodo in G) marca u come non visitato
    for each (u nodo in G) dist[u] = inf
    parent[s] = null; dist[s] = 0
    Q = heap vuoto
    for each (u nodo in G) Q.add(u, dist[u])
    while(Q non vuota)
        u = Q.getMin()
        marca u come visitato (nero)
        for each ((u,v) arco in G)
            if(v non visitato && c_{u,v} < dist[v])
                parent[v] = u; dist[v] = c_{u,v}
                Q.changePriority(v, dist[v])
```

A differenza di quanto accade in Dijkstra, occorre un controllo esplicito che i nodi adiacenti al nodo u estratto dalla coda non siano già stati visitati, perchè non è detto che per un nodo v già visitato il test $c_{u,v} <$ dist[v] sia falso.

1.1 Correttezza

Sia T l'albero di nodi neri (non in Q) corrente. L'invariante è composta di due parti:

1. $T \subseteq MST$ per qualche MST minimo albero ricoprente di G
2. per ogni nodo $u \neq s$ in Q dist[u] = costo minimo di un arco che collegaa u a un nodo nero

L'invariante vale all'inizio:

1. vale banalmente perchè T è vuoto (non ci sono nodi neri)
2. vale banalmente perchè non ci sono nodi neri e la distanza è per tutti infinito, tranne che per s

L'invariante si mantiene:

1. Viene estratto dalla coda un nodo u con dist[u] minima, cioè connesso a un nodo nero y da un arco di costo minimo tra tutti quelli che "attraversano la frontiera", cioè uniscono un nodo nero a un nodo non nero. L'arco (y, u) diventa quindi parte dell'albero di nodi neri corrente.
2. L'insieme dei nodi risulta modificato per l'aggiunta di u, quindi occorre ripristinare l'invarinate (2), controllando se per qualche nodo v non nero e adiacente a u l'arco (u, v) ha costo minore del precedente arco che univa v a un nodo nero, e in tal caso aggiornare l'arco e la distanza.

Postcondizione: all'uscita dal ciclo tutti i nodi sono neri, quindi T connette tutti i nodi, cioè è un albero ricoprente di G. Per l'invariante (1), $T \subseteq MST$ per qualche MST minimo albero ricoprente, quindi $T = MST$.
L'analisi della complessità è esattamente la stessa dell'algoritmo di Dijkstra, quindi $O((n+m) \log n)$.

2 Algoritmo di Kruskal

3 Ordinamento topologico