Capitolo 2.3

Lorenzo Vaccarecci

18 Marzo 2024

1 Microripasso Heap

- heap = albero binario
- ogni nodo ha priorità minore dei figli (minimo=radice)
- quasi completo a sinistra
- bilanciato (altezza $\log n$)

2 Algoritmo di Dijsktra

Grafo pesato Un grafo pesato è un grafo $G = (V, E)$ dove a ogni arco è associato un peso o costo attraverso una funzione $c\_ : E \to \mathbb{R}$. Il costo di un cammino da s a t è la somma dei pesi o costi degli archi che compongono il cammino. Un cammino da s a t si dice *minimo* se ha peso minimo (detto distanza) fra tutti i cammini da s a t.

Problema: dato un grafo orientato pesato G, con pesi non negativi e dati un nodo di partenza s e un nodo di arrivo t, trovare un cammino minimo da s a t. Generalizzazione: dati un grafo orientato pesato G con pesi non negativi e un nodo di partenza s, trovare per ogni nodo u del grafo il cammino minimo da s a u.
Algoritmo di Dijkstra, idea: è una visita in ampiezza in cui a ogni passo:

- per i nodi già visitati si ha la distanza e l'albero T di cammini minimi
- per ogni nodo non ancora visitato si ha distanza provvisoria = lunghezza minima di un cammino in T più un arco
- si estrae dalla coda un nodo a distanza provvisoria minima (che risulta essere la distanza)
- si aggiornano le distanze provvisorie dei nodi adiacenti al nodo estratto tenendo conto del nuovo arco in T

2.1 Pseudocodice

```
Dijkstra(G,s)
    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()
        for each((u,v) arco in G)
            if(dist[u] + c_{uv} < dist[v])
                parent[v] = u
                dist[v] = dist[u] + c_{uv}
                Q.changePriority(v,dist[v])
```

2.2 Correttezza

Sia $d(u)$ la distanza (lunghezza di un cammino minimo) da s a u. L'invariante è composta di due parti:

1. $\forall u \notin H \quad dist[u] = d(u)$
2. per ogni nodo u in Q $dist[u] = d_{\setminus Q}(u)$= lunghezza di un cammino minimo da s a u i cui nodi, tranne u, non sono in Q, ossia sono nodi "neri".

L'invariante vale all'inizio:

1. vale banalmente perchè non ci sono nodi neri (H vuoto)
2. vale banalmente perchè la distanza è per tutti infinito, tranne che per s per cui vale 0 (non ci sono cammini che usano solo nodi neri)

Postcondizione: al termine dell'esecuzione dell'algoritmo la coda è vuota, quindi tutti i nodi sono neri. Poichè vale l'invariante (1), per ogni nodo è stato trovato il cammino minimo da s.