Componenti Connesse e MST

Luca Maltempo

October 2021

1 Introduzione

Un grafo è caratterizzato da un insieme di vertici e di archi che li uniscono. Sul grafo sono possibili diverse operazioni, in questo caso ci occuperemo della ricerca delle componenti connesse di un grafo e della ricerca dell'albero di connessione minimo (MST). Esistono diverse applicazioni reali che richiedono il calcolo di queste due operazioni su grafi molto grandi, quindi, è necessario che gli algoritmi implementati siano veloci anche all'aumentare della grandezza del grafo.

2 Cenni Teorici

2.1 Grafo

Per memorizzare un grafo è necessario mantenere in memoria l'insieme di nodi e i diversi archi che collegano i nodi, per fare questo è stata utlizzata la matrice di adiacenza. In particolare, i grafi creati sono non orientati e pesati, ovvero ogni arco che collega due nodi è caratterizzato da un peso scelto casualmente tra 1 e 9.

2.2 Union Find

Le Union Find sono delle strutture dati per insiemi disgiunti. Ogni insieme è identificato da un unico rappresentante, Si possono eseguire tre diverse operazioni:

- Make-Set(x): Crea un insieme contente un unico elemento x.
- Find-set(x): Restituisce il rappresentante dell'insieme contenente x.
- Union(x,y): Unisce i due insiemi, modifica il rappresentante in modo che questo sia lo stesso tra tutti gli elementi dell'insieme.

Per implementare le Union-Find è sono state usate delle liste doppiamente concatenate. Ogni lista rappresenta un insieme disgiunto, ogni nodo della lista contiene l'elemento dell'insieme, un riferimento al prossimo elemento e un riferimento al rappresentante dell'insieme. Le prime due operazioni della Union-Find

sono banali e hanno costo costante, Union(x,y) invece è una operazione costosa poichè richiede la modifica del rappresentante di tutti gli elementi del secondo insieme che viene unito al primo. Un'euristica utile a velocizzare questo processo prevede semplicemente la concatenazione della lista con meno elementi alla lista con più elementi in modo tale che sia necessario modificare il minor numero di riferimenti al rappresentante possibile. Attraverso un'analisi aggregata per una sequenza di m operazioni su n elementi si ottiene un tempo per Union di $O(m + nlogn)$.

2.3 Componenti Connesse

Dato un grafo G=(V,E) le sue componenti connesse sono i sottoinsiemi disgiunti di V tali che ogni sottoinsieme contenga i vertici che possono essere raggiunti l'uno dall'altro. Quindi due nodi appartengono alla stessa componente connessa se e solo se esiste un cammino che li unisce. Per trovare le componenti connesse di un grafo si usa un algoritmo basato sulle Union-Find:

Algorithm 1 ComponentiConnesse(G)

```
for ogni vertice v ∈ G.V do
    Make-Set(v)
end for
for ogni arco (u, v) ∈ G.E do
    if Find-Set(u) ≠ Find-Set(v) then
        Union(u, v)
    end if
end for
```

Il costo temporale di ComponentiConnesse(G) dipende dall'implementazione di Union-Find, come illustrato nella sezione risultati sperimentali l'algoritmo ha un costo lineare con l'aumentare dei nodi e archi del grafo.

2.4 MST

Dato un grafo G=(V,E) non orientato e pesato è possibile trovare un albero di connessione minimo, ovvero un cammino formato da V-1 archi che unisca tutti i nodi del grafo, abbia costo minimo e non presenti cicli. Per fare questo è stato implementato l'algoritmo MST-Kruskal :

Algorithm 2 MST-Krukal(G,w)

```
A ← ∅
for ogni vertice v ∈ G.V do
    Make-Set(v)
end for
ordina gli archi di G.E in senso non decrescente rispetto al
peso w
for ogni arco (u,v) ∈ G.E do
    if Find-Set(u) ≠ Find-Set(v) then
        A ← A ∪ {(u,v)}
        Union(u,v)
    end if
end for
```

Come per ComponentiConnesse(G) anche l'algoritmo di Kruskal per la ricerca dell'albero di connessione minimo ha un tempo di esecuzione lineare al crescere dei nodi e archi del grafo.

3 Esperimenti svolti

Per testare gli algoritmi delle componenti connesse e del MST sono stati generati casualmente grafi di diverse dimensioni, il più piccolo presenta solo 2 nodi e il più grande 4096, i grafi aumentano di dimensione seguendo la potenza del 2. Ogni grafo, fissata la dimensione, presenta 5 versioni, ognuna con una probabilità di arco tra due nodi differente. Per ogni grafo generato verranno ricercate le componenti connesse e si misurerà il tempo necessario per completare l'operazione, infine, per i grafi che presentano una sola componente connessa sarà possibile applicare l'algoritmo di Kruskal, anche in questo caso si raccoglieranno i diversi tempi di esecuzione al variare della dimensione del grafo e della probabilità di arco.

4 Risultati sperimentali

Numero di Componenti Connesse					
Nodi	p = 0.01	p = 0.02	p = 0.04	p = 0.08	p = 1.00
2	2	2	2	2	1
4	3	4	4	3	1
8	8	8	8	7	1
16	14	12	13	10	1
32	29	21	16	2	1
64	47	26	6	2	1
128	65	18	1	1	1
256	24	6	1	1	1
512	3	1	1	1	1
1024	1	1	1	1	1
2048	1	1	1	1	1
4096	1	1	1	1	1

Table 1: Numero di componenti connesse al variare della probabilità di arco tra nodi p.

Tempo ComponentiConnesse(G) in secondi					
Nodi	p = 0.01	p = 0.02	p = 0.04	p = 0.08	p = 1.00
16	3.3e−5	6.7e−5	4.0e−5	4.6e−5	1.2e−4
32	6.3e−5	1.1e−5	1.2e−4	1.6e−4	4.0e−4
64	2.3e−4	3.4e−4	3.8e−4	4.5e−4	1.4e−3
128	8.9e−4	1.1e−3	1.3e−3	1.4e−3	5.6e−3
256	4.0e−3	4.0e−3	3.5e−3	5.0e−3	3.1e−2
512	1.3e−2	1.8e−2	1.2e−2	2.4e−2	9.6e−2
1024	4.9e−2	4.1e−2	7.3e−2	6.1e−2	3.7e−1
2048	2.1e−1	2.7e−1	2.0e−1	2.3e−1	1.46
4096	1.04	0.88	0.83	1.35	5.75

Table 2: Tempo(s) per ricerca di componenti connesse al variare della probabilità di arco tra nodi p.

Tempo MST-Kruskal(G,w) in secondi					
Nodi	p = 0.01	p = 0.02	p = 0.04	p = 0.08	p = 1.00
16	-	-	-	-	1.5e−4
32	-	-	-	-	5.3e−4
64	-	-	-	-	2.7e−3
128	-	-	2.4e−3	2.5e−3	8.3e−3
256	-	-	5.3e−3	5.9e−3	3.5e−2
512	-	2.1e−2	2.0e−2	2.0e−2	1.4e−1
1024	5.3e−2	6.8e−2	8.5e−2	9.1e−2	5.8e−1
2048	2.3e−1	2.6e−1	2.7e−1	3.9e−1	2.41
4096	1.04	9.8e−1	1.44	1.85	10.13

Table 3: Tempo(s) per ricerca dell'albero ricoprente minimo al variare della probabilità di arco tra nodi p. Il trattino − viene usato nel caso in cui il grafo con n nodi ha più di una componente connessa.

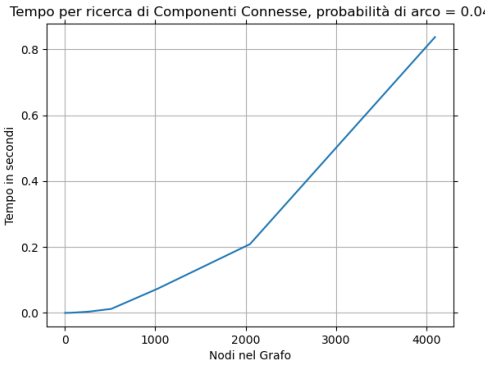

Figure 1: Tempo per ricerca componenti connesse, $p = 0.04$

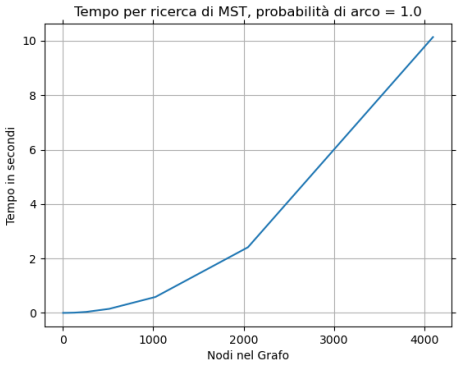

Figure 2: Tempo per ricerca di MST, $p = 1.00$

5 Conclusioni

Analizzando la prima tabella si nota che tutti i grafi con numero di nodi superiore elevato presentano una sola componente connessa anche in caso di bassa probabilità di archi tra nodi, per questi grafi è quindi possibile cercare il MST. Dai due grafici riportati nella sezione precedente si nota che entrambe le operazioni, ComponentiConnesse(G) e MST-Kruskal(G,w), vengono eseguite in un tempo lineare rispetto al numero dei nodi.