Insegnamento di Laboratorio di algoritmi e strutture dati

Implementazione algoritmo di Kruskal

Roberto Posenato

ver. 0.7, 29/01/2008

Sommario

Implementazione algoritmo di Kruskal

Introduzione

- L'algoritmo determina l'albero di copertura (spanning tree) minimo di un grafo pesato non diretto;
- È un semplice esempio di algoritmo *greedy*;
- Ha complessità in tempo $O(|E|\log|E|)$;
- Il funzionamento dell'algoritmo si assume noto dal corso di teoria;
- In questa esercitazione si spiegano i dettagli di come si può implementare l'algoritmo usando parte della classi sviluppate nelle esercitazioni precedenti.

Implementazione algoritmo di Kruskal

Idea algoritmo generico

- L'idea è costruire l'albero a partire dall'insieme vuoto aggiungendo di volta in volta un arco sicuro di peso minimo;
- Un arco è sicuro rispetto all'albero se la sua aggiunta non determina un ciclo nell'albero;
- Kruskal ha proposto un modo efficiente per determinare l'arco ad ogni ciclo.

Procedura `Generic-MST(`G, w`)`

1: $A = \emptyset$;
2: **while** A *non è uno spanning tree* **do**
3: cerca un arco sicuro di peso minimo (u, v) per A;
4: $A = A \cup (u, v)$;
5: **endw**
6: **return** A;

Implementazione algoritmo di Kruskal

Idea algoritmo di Kruskal

- L'algoritmo generico di fatto costruisce una foresta a partire dai singoli nodi.
- È possibile che la foresta diventi un unico albero solo all'ultimo ciclo.
- L'idea di Kruskal è di gestire gli alberi che compongono la foresta mediante disjoint set;
- In questo modo l'operazione di ricerca arco minimo sicuro si può fare in modo efficiente.

Implementazione algoritmo di Kruskal

Disjoint set

- L'ADT Disjoint set permette di gestire una collezione di insiemi disgiunti dinamici.
- Ogni insieme è identificato da un rappresentante.
- Siano x, y due elementi. Le operazioni fondamentali sono:
  - **MakeSet(x)**: se x non è già in un insieme, crea un nuovo insieme ponendo x come rappresentante;
  - **Union(x, y)**: si fondono l'insieme che contiene x con l'insieme che contiene y per formare un nuovo insieme. Dato che si deve operare su insiemi disgiunti, l'insieme unione sostituisce i due insiemi.
  - **FindSet(x)**: restituisce il rappresentante dell'insieme che contiene x se esiste, nulla altrimenti.

Implementazione algoritmo di Kruskal

Algoritmo di Kruskal

Procedura `Kruskal-MST(`G, w`)`

```
1:  A = ∅;
2:  foreach v ∈ V do MakeSet(v); //Costruzione foresta
3:  E' = Ordina E rispetto a w in modo non decrescente;
4:  foreach (u, v) ∈ E' do
5:      if (FindSet(u) ≠ FindSet(v)) then
6:          A = A ∪ (u, v);
7:          Union(u, v);
8:      endif
9:  endfch
10: return A;
```

Implementazione algoritmo di Kruskal

Complessità algoritmo di Kruskal

- La complessità in tempo dipende dall'implementazione dell'adt disjoint set;
- Se si implementa il disjoint set con l'euristiche union-by-rank e path-compression, si ottiene:
  - L'inizializzazione richiede tempo $O(|V|)$;
  - L'ordinamento richiede tempo $O(|E|\log|E|)$;
  - Ogni operazione sui disjoint-set richiede tempo $\alpha(|E|,|V|) \approx O(\log|E|)$, dove $\alpha()$ è l'inverso della funzione di Ackermann;
  - Nel caso peggiore ci sono $3|E|$ operazioni sui disjoint set;
  - Il tempo totale è quindi $O(|E|\log|E|)$.

Implementazione algoritmo di Kruskal

Verso l'implementazione: le euristiche per il disjoint set

Euristica union-by-rank

- Ogni insieme è rappresentato da un pseudo-albero: nodo = elemento, radice = rappresentante dell'insieme.
- Ogni nodo ha il riferimento father e può essere indirizzato direttamente. Perché?
- L'operazione union-by-rank prevede di fondere due alberi trasformando la radice dell'albero di dimensione inferiore in figlio della radice dell'altro.
- L'inconveniente di questa idea è dover conoscere la dimensione degli alberi per poter fare l'unione in modo efficiente.

Implementazione algoritmo di Kruskal

Verso l'implementazione: le euristiche per il disjoint set

continua euristica union-by-rank

- Un modo per aggirare tale problema è mantenere in ciascun nodo un valore (rank) che è un upper bound all'altezza del nodo che approssima il logaritmo della dimensione dell'albero che ha il nodo come radice.
  - Quando il nodo è foglia, il suo rank è 0;
  - Quando un nodo x diventa figlio di un nodo y, il rank di y (maggiore o uguale a quello di x) viene incrementato di 1 solo se è pari a quello di x;

Implementazione algoritmo di Kruskal

Verso l'implementazione: le euristiche per il disjoint set

Euristica path-compression

- In generale la costruzione degli alberi mediante la union-by-rank può portare a alberi degeneri (leggi liste);
- Gli alberi generici possono rendere inefficienti le operazioni. Perché?
- Un modo per evitare tali situazioni, è il seguente:
  - ad ogni operazione FindSet(x),
  - per ciascun nodo che si incontra da x fino alla radice,
  - si assegna come padre del nodo la radice dell'albero.
- Questa operazione non richiede la modifica del rank del nodo. Perché?

Implementazione algoritmo di Kruskal

Verso l'implementazione: analisi strutture dati necessarie

L'implementazione dell'algoritmo richiede:

- Una classe per rappresentare l'ADT grafo in cui le seguenti operazioni dovranno essere rese più efficienti possibili:
  - accesso al nodo;
  - accesso agli adiacenti di un nodo;
  - accesso all'insieme dei nodi;
  - accesso ad un arco;
  - accesso all'insieme degli archi;
- Una classe per rappresentare l'ADT disjoint set in cui le operazioni già viste dovranno essere implementate secondo le euristiche proposte;
- Una classe che offra l'algoritmo di Kruskal come metodo statico;

Implementazione algoritmo di Kruskal

Verso l'implementazione: interfacce

- Si definisca un'interfaccia minima per ciascuna delle seguenti ADT:
  - **nodo** di un grafo;
  - **arco** di un grafo;
  - **grafo**;
  - **disjoint set**;

Implementazione algoritmo di Kruskal

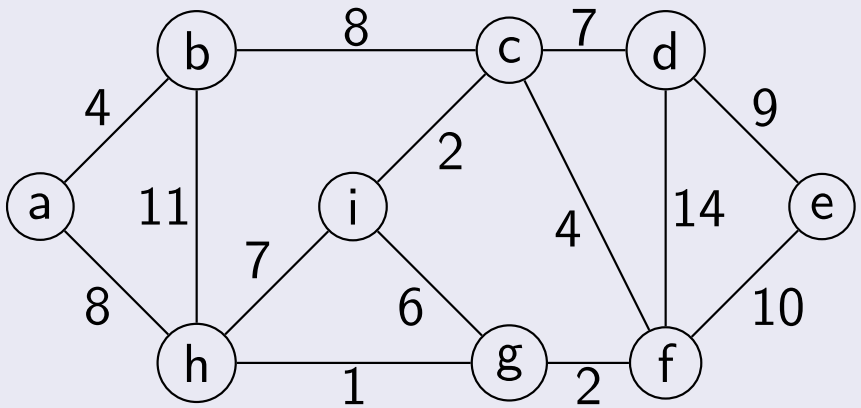
Verso l'implementazione: implementazioni

- Si implementi quindi ciascuna delle interfacce precedenti, osservando:
  - le indicazioni delle slide precedenti
  - riusando, dove possibile, le classi sviluppate nelle esercitazioni precedenti.

Implementazione algoritmo di Kruskal

Esercitazione

- Si implementi, infine, l'algoritmo di Kruskal:
  - eseguire dei test di correttezza con grafi vuoti/solo 1 nodo/lineari.
  - determinare qual è l'albero di ricoprimento minimo del grafo



  - stampare l'albero nel formato definito nell'esercitazione degli alberi con i pesi degli archi
  - stampare il peso dell'albero