ale-cci

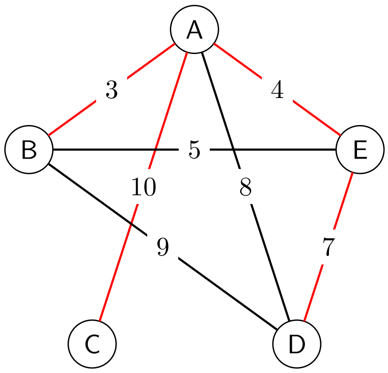
Modelli Algoritmi per il Supporto alle Decisioni

March 31, 2020

Introduction

TODO

Minimum Spanning Tree



Algoritmo di Kruskal (Greedy)

```python
from utils import num_vertices

def kruskal(edges: list, N: int) -> list:
    connected = set()
    mst = []

    edges = sorted(graph)
    for edge in edges:
        weight, lhs, rhs = edge

        # Two nodes already connected
        if lhs in connected and rhs in connected:
            continue

        mst.append(edge)
        connected.update({lhs, rhs})

        if len(mst) == N:
            break

    return list(mst)


if __name__ == '__main__':
    graph = [(10, 'A', 'C'), (8, 'A', 'D'),
             (7, 'D', 'E'), (4, 'A', 'E'),
             (3, 'B', 'A'), (9, 'B', 'D'), (5, 'B', 'E')]
    N = num_vertices(edges=graph)

    print(kruskal(graph, N))
```

Correttezza algoritmo di Kruskal

Supponiamo per assurdo che esista un' diverso MST $T' = (V, E_{T'})$ di peso inferiore a $T = (V, E_T)$, quello restituito dall'algoritmo greedy.

Siccome i due alberi hanno costo diverso, differiscono di almeno un' arco. Indichiamo con e_h l'arco a peso minore appartenente a $\{E_T - E_{T'}\}$. Dato che T' è un MST, esiste un ciclo C in $\{e_h\} \cup E_{T'}$ contenente l'arco e_h. Siccome anche T è un albero, quindi non ha cicli, allora $C \cap E_T \neq \emptyset$. Chiamiamo e_r l'arco a peso minore appartenente a $C \cap \{E_T - E_{T'}\}$. Necessariamente $w_{e_r} \leq w_{e_h}$, altrimenti l'algoritmo greedy applicato a T avrebbe selezionato prima e_h al posto di e_r. Sostituendo in T' l'arco e_r con e_h ottengo un nuovo albero di peso inferiore.

Questo va contro l'ipotesi T' è l'albero di supporto a peso minore.

Analisi complessità algoritmo greedy

$O(E \cdot \log(E))$, dovuta all'ordinamento degli archi in ordine di peso. Il controllo dell'esistenza di cicli è effettuato in $O(1)$.

Foresta di supporto

Viene chiamata foresta di supporto di un grafo G un grafo parziale $F = (V, E_F)$ privo di cicli. In particolare, un albero di supporto è una foresta con una sola componente connessa.

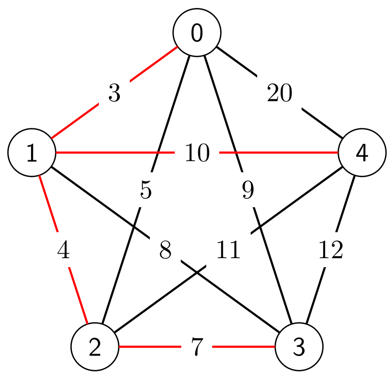
Teorema

Indichiamo con $(V_1, E_1), \ldots, (V_k, E_k)$ le componenti connesse di una foresta di supporto $F = (V, E_F)$ del grafo G. Sia inoltre (u, v) un arco a peso minimo tra quelli con un unico estremo in V_1. Allora esiste almeno un albero di supporto a peso minimo appartenente a $\bigcup_{i=1}^{k} E_i$, che contiene (v, v).

Dimostrazione

Per assurdo, l'albero a peso minimo non contiene (u, v). Ma aggiungendo (u, v) a tale albero si forma un ciclo contenente un altro arco (u', v') con un solo estremo in V_1. Se si toglie questo arco e si lascia (u, v) si ottiene un albero di peso minore, contraddicendo l'ipotesi.

Algoritmo MST-1




```python
from collections import defaultdict

def argmin(array: list) -> int:
    arg = min((value, idx) for idx, value in enumerate(array))
    return arg[1]

def mst_1(w: list) -> list:
    V = set(range(len(w)))  # {0, 1, 2, 3, 4}
    c = [0] * len(V)        # [0, 0, 0, 0, 0]
    U = {0}
    mst = []

    while U != V:
        weight, u = min((w[v][c[v]], v) for v in V - U)
        U.add(u)
        mst.append((u, c[u]))
```

```python
        for v in V - U:
            if w[v][u] < w[v][c[v]]:
                c[v] = u

    return mst

if __name__ == '__main__':
    w = [[ 0,  3,  5,  9, 20],
         [ 3,  0,  4,  8, 10],
         [ 5,  4,  0,  7, 11],
         [ 9,  8,  7,  0, 12],
         [20, 10, 11, 12,  0]]

    print(mst_1(w))
```

Correttezza MST-1

Contents