

Zad. 12

$$1. TSP(G) \leq 2 \cdot MST(G)$$

Wierzę, że można to udowodnić. Zdeblamy jego wszystkie krawędzie wtedy każdy wierzchołek ma maksymalnie 2 krawędzie. Z tw. posiada on ścieżkę Eulera, ta ścieżka może być rozciętą na TSP.

$$2. MST(G) \leq TSP(G)$$

MST(G) jest ~~sumą~~ najmniejszą sumą krawędzi potrzebnych aby graf był spójny. Zatem gdyby  $TSP(G) < MST(G)$  to można by usunąć cykle i uzyskać lepsze (mniejsze) MST.

Zad. 6

Drewno rozpinające:


Jeżeli G jest grafem spójnym to algorytm może „rozciąć” go (it.). Gdyby wzięty graf nie byłby drewnem to musiałby zawierać cykl, istniałoby więc pewna krawędź której usunięcie nie rozciąłoby grafu, musiałaby ona zostać usunięta wtedy jest to drewno rozpinające.

Minimalna waga:

Pokazujemy, że w ~~każdej~~ iteracji graf zawiera MST:

1. Na początku iteracji ten jest spełniony, każdy graf ma MST.
2. Zał. że w pewnym momencie iteracji ten jest prawdziwy, mamy MST zawarte jako T, a usuwamy kraw. e:

i) e nie należy do T, więc dalej mamy MST  $T' = T$ . 

ii) Wyp. Różni usuwając e, T zawiera cykl, więc f z tego cyklu, usuwamy f & T. 

$T' = T - e + f$  jest MST bo:

a) T' rozpinane bo usuwając kraw. e dodajemy kraw. f nie powstaje cykl (cykle dalej mamy drewno rozpinające)

b)  $w(e) < c(f)$  nie ma problemu bo iterujemy nie w kraw. nie mającej

$c(f) > c(e)$  ——— T by nie było MST

więc  $c(f) = c(e)$  więc  $T$  &  $T'$  jest MST

Udowodnimy indukcyjnie, że graf w każdej iteracji zawiera MST a skoro nam jest drewno rozpinające to jest MST.



Zad. 3

~~G~~  $G$  - graf,  $G[n]$  - wierzchołki  $1 \dots n$

$L$  - lista uporz. wierzchołków grupy (kt. mają  $indeg 0$ ), ostatni  $n$ .

$S$  - set wierzchołków bez krawędzi wychodzących

$I$  - lista zbliż. krawędzi wychodzących do degree  $indeg$

While  $S$ :

$u := pop(S)$

$L.add(u)$

$v$  in  $G(u)$

for  ~~$v$  in  $G(u)$~~   $v$  in  $G(u)$ :

$Ideg[v]--1$

if  $Ideg[v] == 0$ :

$S.add(v)$

SORTOWANIE  
TOPOLOGICZNE

Zad. 7

~~Znaleźć~~ Znaleźć krawędzie na ~~na~~ prostej wartości i unidirectional dandy algorytm do  
szukania MST. myś. algorytm KRUSKALA

Zad. 9

for  $(u, v)$  in edges:

$dist[u][v] = w(u, v)$

$next[u][v] = v$

for  $v$  in vertices:

$dist[v][v] = 0$

$next[v][v] = v$

ODZYSKIWANIE ŚCIEŻKI

def Path( $u, v$ ):

path = [ $u$ ]

while  $u \neq v$ :

$u = next[u][v]$

path.add( $v$ )

return path

for  $k$  in  $(1, m)$ :

for  $i$  in  $(1, n)$ :

for  $j$  in  $(1, n)$ :

if  $dist[i][j] > dist[i][k] + dist[k][j]$

$dist[i][j] = dist[i][k] + dist[k][j]$

$next[i][j] = next[k][j]$

zad. 2

Idąc zryłem DFS/BFS. Wierzę, że ma być i tak, tak że wierzę, że  
wierzę, że na pewno do wychożenia. Teraz dwa wiązki są połączone i są, tego  
czego. Wtedy to graf nie jest dwudzielny, w p.p. jest dwudzielny podział węzłów  
na dwa portale węg. Wierzę, że to jest