Assignment 5 - AlgDat

Simon Warg
Nicklas Jacobsen
Robert Rasmussen
Christian Enevoldsen

3. juni 2014

Task 1

Vi beviser at et MST T er en del af den oprinelige graf G ved hjælp af et modstridsbevis. Så vil vise at T ikke nødvendigvis er en del af G. Antag at T ikke er en del af G, således at der findes en kant e i T der ikke findes i mængden af kanterne E i G. Men da T er et MST så er det per definition en delmængde $T \subseteq E$. Vi ankommer således til et modstrid, og vores antagelse at $T \nsubseteq E$ må være falsk. Vi kan derfor konkludere at T er en del af G.

Task 2

Let G be the original graph
Let H be a matrix of shortest path distance between breweries
Let MST be the graph created by Prim's Algorithm
getEdge: retrieves a random edge of weight findSize from matrix m, which is not in MST

Algorithm 1 FindLastEdge

```
function findLastEdge(G, m, MST)
    findSize = G.weight − MST.weight
    lastEdge = getEdge(findSize, m, MST)
    G2.add(lastEdge)
end function
```

Exame outline - MST - Nicklas W. Jacobsen

Definition af Minimum spanning tree

Et MST af en givet graf G, er en non-cyklisk del-graf T som forbinder alle knuder i G, med andre ord er T et træ som udspænder G.

Dyrkning af en MST for en givet graf G

Hvis vi vil finde en MST af en givet graf $G = (V, E)$, med en vægt givet som $w : E \to \mathbb{R}$, definerer vi en A som værende et sub-sæt af en eller anden MST for G. Herefter følger vi en grådig metode, hvor vi tilføjer en sikker kant til A, en efter en indtil vi har en MST af G. Vi definerer en sikker kant for A, som værende en kant der ikke bryder invarianten.

Algorithm 2 GENERIC-MST

function GENERICMST(G, w)
    $A = \emptyset$
    while A does not form a MST, find an edge (u, v) this is safe for A **do**
        $A = A \cup (u, v)$
    end while
    Return A
end function

Initialisering: Efter linje 2, holder A trivielt invarianten.

Vedligeholdelse: I hver iteration på linje 3 og 4, bliver invarianten holdt da det kun er sikre kanter der bliver tilføjet til A.

Terminering: Interationen stopper når A er en MST af G, og A bliver retuneret. Derfor må funktionen retunerer MST af G.

Kruskal's algoritme

Algoritmen bruger "dis-join" datatype til at holde styr på hvorvidt om det sikkert at tilføje en kant til A. Hvis det ikke er forsætter den til næste kant

Algorithm 3 MST-KRUSTAL

```
function MST-KRUSTAL(G, w)
    A = ∅
    for each vertex e ∈ G.V do Make-Set(v)
    end for
    Sort G.E in ascending order by weight w
    for each (u, v) ∈ G.E do
        if Fïnd-Set(u) ≠ Find-Set(v) then
            A = A ∪ (u, v)
            Union(u,v)
        end if
    end for
    Return A
end function
```

Exam outline - Robert Rasmussen

points

- What is Minimum Spanning Trees
- Prim's algorithm
- Kruskal's algorithm

Problem Instance

A weighted graph to run Prim's on.

Exam outline - Christian Enevoldsen

Disposition

- Definition
- Generisk Algoritme
- Prim's algoritme: Tidskompleksitet

- Kruskal's algoritme: Tidskompleksitet
- Problem instans: Kruskal's algoritme paa en vaegtet graf.

Exam-outline - Simon Warg

Minimum spanning trees

A MST is a weighted tree T containing the nodes $T.V$ of a graph $G = (V, E)$ and a subset of the graph's edges $T.E \subseteq G.E$ such that the sum of G's weight $\sum\limits_{(u,v) \in G.E} \omega(u, v)$ is minimized.

Prim's algorithm

A MST can be optained by following Prim's algorithm. It maintains a priority queue Q where all not-yet visited verticies resides. It then iterates Q until it's empty. Each iteration, it picks the edge with lowest edge connecting cut T and $V - T$ and then add it to T.