

datastructuren en algoritmen
2010 11 17
college 13

- datastructuren
- algoritmen
- tijdscomplexiteit

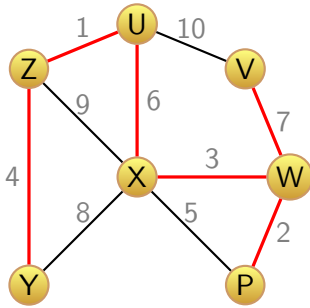
schema

- minimum spanning tree
- wat hebben we gezien?
- over het tentamen
- materiaal

minimum spanning tree

- **uitgangspunt:**
graaf $G = (V, E)$
ongericht, simpel, gewogen
- **spanning tree:**
subset $T \subseteq E$
die alle knopen bevat en acyclisch is (dus een 'boom')
- **minimum spanning tree:**
spanning tree T
met gewicht $w(T) = \sum_{(u,v) \in T} w(u, v)$ minimaal
- **toepassingen:**
communicatienetwerken, transportnetwerken

spanning tree: voorbeeld



minimum spanning tree algoritmen: idee

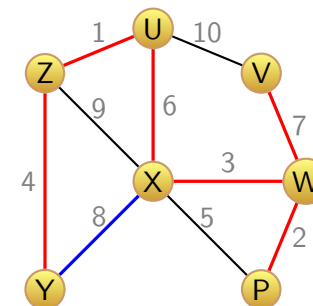
- **probleem:**
gegeven een graaf $G = (V, E)$ (ongericht, gewogen, simpel)
vind een minimum spanning tree
- **greedy algoritmen:**
voeg in elke stap een kant toe aan een subset $A \subseteq E$
- **invariant:**
voor elke iteratie is A een subset van een minimum spanning tree
- **iteratie:**
voeg aan A een kant e toe die **veilig** is
dwz zo dat $A \cup \{e\}$ is weer een subset van een mst

welke kanten zijn veilig?

- **gegeven:**
 $A \subseteq E$ subset van een minimum spanning tree
 $X \subseteq V$ de knopen in A
- **veilige kant voor A :**
een kant (x, y) met $x \in X$ en $y \in V \setminus X$ met minimum gewicht
- **bewijs:**
Zij $T \supseteq A$ een minimum spanning tree voor G .
Onderscheid gevallen
 $(x, y) \in T$ (makkelijk: dan klaar)
 $(x, y) \notin T$ (construeer mst T')

cykel eigenschap

- **cykel eigenschap:**
als een mst T samen met een kant $e \notin T$ een cykel vormt
dan heeft elke kant $f \in T$ op de cykel een kleiner gewicht dan E
- **bewijs:**
stel niet
neem dan e in plaats van f
dit levert spanning tree met kleiner gewicht dan T
tegenspraak met aanname dat T een mst is



MST algoritme van Kruskal

- **init:**
elke knoop $v \in V$ is een cluster
priority queue met kanten met gewicht als key
- **iter:**
zolang A nog minder dan $n - 1$ kanten:
neem kant met minimaal gewicht
verbind bijbehorende clusters als die verschillend zijn

MST algoritme van Prim en Jarnik

- lijkt op Dijkstra's kortste pad algoritme
- **init:**
neem een knoop $v \in V$ en neem $A = \{v\}$
afstand tot A : $D[v] = 0$ en $D[u] = \infty$ voor $u \neq v$
- **iter:**
neem een knoop $u \notin A$ met $D[u]$ minimaal
update $D[z] := w(u, z)$ voor alle burens z van u
als dat $D[z]$ kleiner maakt

MST algoritme van Kruskal: complexiteit

in $\mathcal{O}(m \log(n))$

met m het aantal kanten en n het aantal knopen

algoritme van Prim-Jarnik: complexiteit

in $\mathcal{O}(m \log(n))$

met m het aantal kanten en n het aantal knopen

analyse: lijkt op analyse van Dijkstra's algoritme

schema

- minimum spanning tree
- wat hebben we gezien?
- over het tentamen
- materiaal

vectoren, lijsten, rijtjes

- ADTs: idee, niet alle operaties letterlijk
- implementaties met gelinkte lijsten
analyse van bijbehorende complexiteit

datastructuren: stacks en queues

- ADTs met bijbehorende operaties
- implementaties met array, circulair array
- ADT versus implementatie

bomen

- **algemene bomen**: ADT, diepte, hoogte,
- traversals: preorder, postorder
- **binaire bomen** ADT
- **traversals**: preorder, postorder, inorder, Euler
- implementaties, analyse complexiteit

priority queues

- idee, basis ADT operaties
- priority queue sorteerschema

heaps

- wat is een heap
- toevoegen, wortel verwijderen
- implementatie priority queue met heap
- heap-sort
- bottom-up heap construction

hashing

- hash tables, collision
- lineair probing, quadratic probing, double hashingx

binaire zoekbomen

- wat is een binaire zoekboom
- toevoegen, verwijderen
- complexiteit van operaties

AVL bomen

- een manier om de hoogte van een BST binnen de perken te houden
- toevoegen, verwijderen, complexiteit daarvan

paradigma's

- verdeel en heers
- gulzig
- dynamisch programmeren

sorteren

- sorteeralgoritmes
merge, quick, bubble, insertion, selection, bucket, radix
- complexiteitsanalyse met recurrente betrekking of recursieboom
- ondergrens op comparison-based sorteren
- zoek

grafen

we gebruiken maar weinig echte grafen-theorie

- knopen, kanten,
- van een knoop:
buren, uitgaande kanten, inkomende kanten, graad,
- van een graaf:
gewogen, gericht of ongericht, simpel, eenvoudige resultaten (Thm 6.6)
edge list versus adjacency list implementatie

graafalgoritmen

- Dijkstra's kortste pad
- Bellman-Ford kortste pad
- alle kortste paden
- Kruskal's MST
- Prim-Jarnik MST

text processing

- brute force pattern matching algoritme
- Boyer-Moore pattern matching algoritme met last-functie
- Knuth-Morris-Pratt pattern matching algoritme met failure-functie
- Huffman's coderingsalgoritme
- longest common subsequence

datastructuren

- **lineair**
stacks, queues, priority queues, vectoren, lijsten, rijtjes, arrays, strings, hash-tables
- **bomen**
bomen, binaire bomen, binaire zoekbomen, AVL-bomen, heaps, Huffman boom, (standard, compressed, suffix) tries
- **grafen**
al bekend van eerdere colleges;
kan zijn: gericht/ongericht, gewogen, samenhangend

algoritmen

- **op lineaire datatypen:**
sorteren, zoeken, comprimeren
- **op bomen:**
traversals, structuur herstellen, zoeken
- **op grafen:**
kortste paden, minimale opspanning

schema

- minimum spanning tree
- wat hebben we gezien?
- over het tentamen
- materiaal

communicatie

- **vanaf nu:**
updates worden expliciet vermeld op de webpage
- **femke @ T446**
niet op woensdagen
- **femke @ cs.vu.nl**
voor vragen en opmerkingen, liefst DS oid in subject

stof

- alles wat op college en werkcollege behandeld is
- **slides**
bevatten soms dingen die niet in het boek staan
- **werkcollege opgaven**
- **boek zoals vermeld op werkcollege-blaadjes en slides**

tentamen

- **niveau:**
denk aan werkcollege-opgaven, iets moeilijker dan voortentamen
- **soort vragen:**
manipuleer met bekende datastructuur
pas algoritme A toe
analyseer een (gegeven of bekend) algoritme
geef datastructuur en/of algoritme voor probleem P
etcetera

uit het hoofd of niet?

ik ga ervan uit dat je zonder extra gegevens kunt toepassen ten minste:

- traversals van bomen
- toevoegen aan en verwijderen uit, inclusief eventueel herstructureren, bijv AVL bomen en heaps
- hashing met verschillende manieren om collision te vermijden
- sorteeralgoritmes merge, quick (ook select), insertion, selection, radix, bucket
- fractional knapsack, knapsack01, task scheduling
- string pattern matching met hint over BM (met last) of KMP (met failure)

materiaal

- boek 7.3.1, 7.3.2

schema

- minimum spanning tree
- wat hebben we gezien?
- over het tentamen
- materiaal