



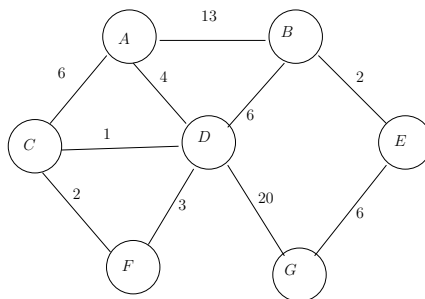
## Stof

- slides college 10
- boek 5.3, 5.1
- boek 6.2
- boek 7.1.1
- zie ook boek 9,4 voor LCS

(Twee verwijderde opgaven zijn ook echt verwijderd nu; de nummering is daardoor aangepast.)

## Opgaven

1. Geef optimale haakjes voor een rijtje van matrix-vermenigvuldigingen met dimensies 5, 10, 3, 12, 5.
2. Geef een efficient algoritme dat als input neemt een rijtje matrixen die vermenigvuldigd kunnen worden, en dat als output geeft dat rijtje met optimale en volledige haakjes.
3. Geef een alternatief algoritme voor houthakken dat top-down werkt maar subresultaten hergebruikt.
4. Bepaal een Longest Common Subsequence (LCS) van  $[1, 0, 0, 1, 0, 1, 0, 1]$  en  $[0, 1, 0, 1, 1, 0, 1, 1, 0]$  met het algoritme.
5. Pas Dijkstra's kortste pad algoritme toe op de volgende graaf, met  $A$  als startknoop. Geef stap voor stap aan wat er gebeurt, door in elke stap de graaf met de relevante data te tekenen.



6. Geef een klein voorbeeld waaruit blijkt dat Dijkstra's kortste pad algoritme niet (correct) werkt voor grafen met negatieve gewichten.
7. Je zou kunnen overwegen de situatie uit de vorige opgave te redden door alle gewichten op te hogen met  $n$  als  $-n$  het kleinste gewicht is. Dat levert een graaf met niet-negatieve gewichten. Leg uit waarom op die graaf Dijkstra's algoritme toepassen wel of niet werkt.
8. Stel we veranderen Dijkstra's algoritme door in plaats van 'zolang  $Q$  niet leeg' te gebruiken 'zolang  $Q$  meer dan 1 element'. Is het algoritme dan nog correct?

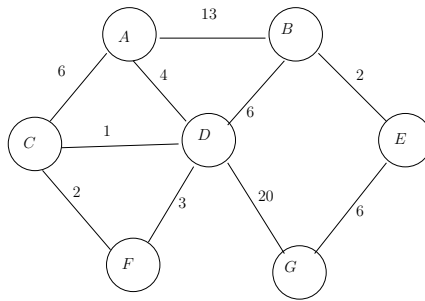
### Nog meer opgaven

1. (Dit is ongeveer opgave R-5.2.)  
Hoe kan het TaskSchedule schema uit het boek (5.1.2) geïmplementeerd worden in  $\mathcal{O}(n \log(n))$ ?
2. Toon aan dat een expressie van  $n$  elementen met een binaire operator precies  $n - 1$  paren haakjes heeft.
3. Bekijk de recursie-boom voor merge-sort van opgave 11 uit werkcollege 6.  
Leg uit waarom memoization (hergebruiken van eerdere berekende deelresultaten) niet veel zin heeft in het geval van merge-sort.
4. Wat is de tijdscomplexiteit van het algoritme voor Longest Common Subsequence in termen van grote-Oh?
5. Hoe kun je het algoritme voor LCS aanpassen zó dat het de lengte van het langste monotoon (strict) stijgende gemeenschappelijke deelrijtje geeft?
6. Een aanpassing van het knapsack01 probleem.  
Gegeven een verzameling  $S$  voor het knapsack01 probleem met de volgende eigenschap: de volgorde van de items indien gesorteerd met oplopend gewicht ( $w$ ), is hetzelfde als de volgorde van de items indien gesorteerd met afnemende waarde ( $b$ ). Voorbeeld:

	$b$	$w$
$s_1$	3	1
$s_2$	2	2
$s_3$	1	3

Geef voor deze aanpassing van knapsack01 een efficiënt algoritme.

7. Gegeven is de graaf



zoals ook in de opgave over Dijkstra's algoritme hierboven. Geef de edge-list implementatie (schematisch), en ook de adjacency-list implementatie.x