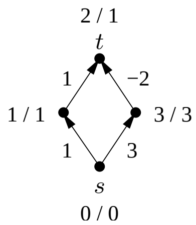
Algoritmer og Datastrukturer 2 (Sommer 2004)

1a

$n = rk + 2$.
$m = 2k + 2(r-1)(k-1)$.
Dijkstra: $O(m \log n) = O((2k + 2(r-1)(k-1)) \log(rk+2)) = O(rk \log(rk))$.

1b



På grafen er angivet "Dijkstra's afstand / rigtige afstand".

1c

Da en kryds-graf er acyklisk, kan de korteste afstande fra s til alle knuder findes vha. DAGshortestPath [GT, Algoritme 7.9] i tid $O(n+m)$. Da $m \leq 2n$ er dette $O(n)$.

2a

Bellman-Ford: $O(nm) = O((rk+2)(2k + 2(r-1)(k-1)+1)) = O(r^2k^2)$.

2b

Lad G' være grafen G med kanten (t,s) fjernet. Afstanden fra u til v i grafen G kan beregnes som $d_G(u,v) = \min\{d_{G'}(u,v), d_{G'}(u,t) + w(t,s) + d_{G'}(s,v)\}$. Da G' er en DAG, kan vi bruge DAGshortestPath [GT, Algoritme 7.9] to gange for at finde den kortest vej fra hhv. u og s til alle øvrige knuder i $O(n+m)$ tid. De korteste afstande fra u kan nu beregnes i $O(1)$ for hver af de n knuder vha. den nævnte formel. Da $m \leq 2n$ bliver den totale tid $O(n)$.

2c

Kør algoritmen fra **2b** n gange, en gang for u værende hver af de n knuder. Da algoritmen fra **2b** tager tid $O(n)$, bliver den totale tid $O(n^2)$.

3a

$i \leftarrow 1$, $j \leftarrow n$
while $i < j$
    fjern letteste kant e fra cycklen (u, v_i, w, v_j, u)
    if e incident til v_i **then** $i \leftarrow i+1$
    else $j \leftarrow j-1$

Hver iteration af **while**-løkken fjerner en af de $2n$ kanter, dvs. $O(n)$ iterationer der hver tager $O(1)$ tid da man kun betragter 4 kanter i hver iteration **while**-løkken.

4a

$k \setminus s$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	S																					
1			S		S				S								S					
2						S		S				S		S				S		S		
3										S		S				S		S				S

4b

procedure $B(n, N)$
  for $k = 0$ **to** n
    for $s = 0$ **to** N
      $A[k, s] \leftarrow$ **false**
      if $k = 0$ **then**
        if $s = 0$ **then** $A[k, s] \leftarrow$ **true**
      else
        if $x_k = 1$ **then**
          if $s > 0$ **and** $A[k-1, s-1]$ **then** $A[k, s] \leftarrow$ **true**
        else
          $p \leftarrow x_k$
          while $p \leq s$
            if $A[k-1, s-p]$ **then** $A[k, s] \leftarrow$ **true**
            $p \leftarrow p * x_k$
  return $A[n, N]$

Da **while**-løkken forøger p med en faktor x_k i hver iteration, gennemløbes denne højest $\log_{x_k} s \leq \log_2 N$ gange. De to **for**-løkker giver total tid $O(nN \log N)$.

4c

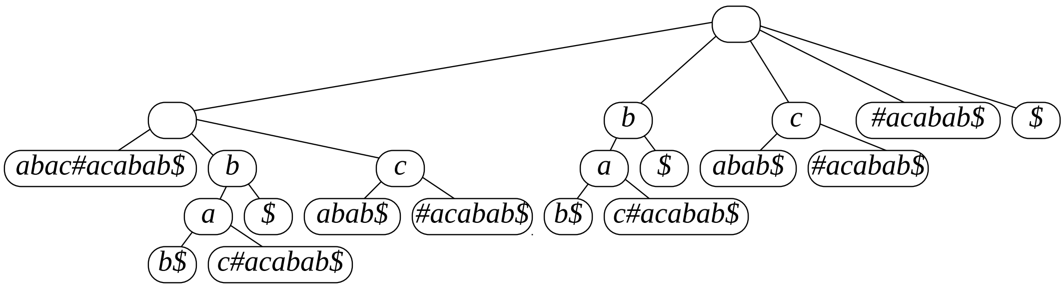
Den sidste linie i pseudo-koden til **4b** erstattes med nedenstående. Som argumenteret i **4b** gentages **while**-løkken højest $\log_2 N$ gange, dvs. total yderligere tid i forhold til **4b** er $O(n \log N)$. Den totale tid forbliver $O(nN \log N)$.

  if $A[n, N]$ **then**
    $s \leftarrow N$
    for $k = n$ **downto** 1
      $i \leftarrow 1$, $p \leftarrow x_k$
      while $A[k-1, s-p] =$ **false**
        $i \leftarrow i + 1$, $p \leftarrow p * x_k$
      $d_k \leftarrow i$, $s \leftarrow s - p$
    return $d_1, \ldots, d_k$
  else
    return "Der findes ingen løsning"

5a

$a\,a\,b\,a\,c\,\#\,a\,c\,a\,b\,a\,b\,\$$
$a\,b\,a\,b\,\$$
$a\,b\,a\,c\,\#\,a\,c\,a\,b\,a\,b\,\$$
$a\,b\,\$$
$a\,c\,a\,b\,a\,b\,\$$
$a\,c\,\#\,a\,c\,a\,b\,a\,b\,\$$
$b\,a\,b\,\$$
$b\,a\,c\,\#\,a\,c\,a\,b\,a\,b\,\$$
$b\,\$$
$c\,a\,b\,a\,b\,\$$
$c\,\#\,a\,c\,a\,b\,a\,b\,\$$
$\#\,a\,c\,a\,b\,a\,b\,\$$
$\$$

5b



5c

Givet to strenge S_1 og S_2, hvor $|S_1| + |S_2| = n$, konstuer suffix-træet for $S = S_1\#S_2\$$ i $O(n)$ tid ifølge antagelsen. I et postorder gennemløb af suffix-træet marker bladene "S_1" eller "S_2" hvis de svarer til suffixer startende i hhv. S_1 eller S_2. Indre knuder markeres S_1 og/eller S_2 hvis mindst et af børnene er markeret S_1 og/eller S_2. I et preorder gennemløb beregn for hver knude længden af strengen fra roden til og med knuden. Husk knuden svarende til den længste streng hvor knuden er markeret både "S_1" og "S_2". For knuden med det længste suffix i både S_1 og S_2 retuneres strengen. Foruden konstruktionen af suffix-træet, så tager både preorder og postorder gennemløbet tid $O(n)$, så den totale tid bliver $O(n)$.