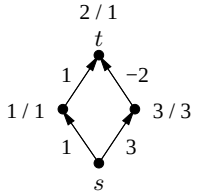
Algoritmer og Datastrukturer 2 (Sommer 2004)

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] ← false
      if k = 0 then
        if s = 0 then A[k, s] ← true
      else
        if x_k = 1 then
          if s > 0 and A[k − 1, s − 1] then A[k, s] ← true
        else
          p ← x_k
          while p ≤ s
            if A[k − 1, s − p] then A[k, s] ← true
            p ← 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

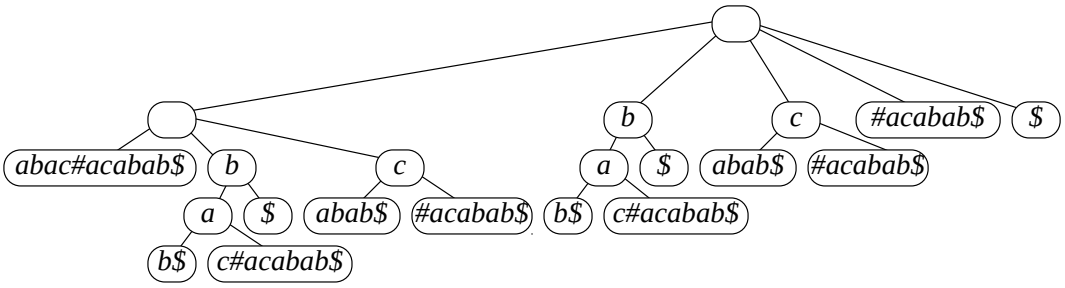
Den sidste linie i pseudo-koden til **4b** erstattes med nedenstående. Som argumenteret i **4b** gentages **while**-løkken højest $\log_2 N$ gange, dvs. total yderligere tid i forhold til **4b** er $O(n \log N)$. Den totale tid forbliver $O(nN \log N)$.

```
  if A[n, N] then
    s ← N
    for k = n downto 1
      i ← 1, p ← x_k
      while A[k − 1, s − p] = false
        i ← i + 1, p ← p ∗ x_k
      d_k ← i, s ← s − p
    return d_1, ..., 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 $O(n)$.

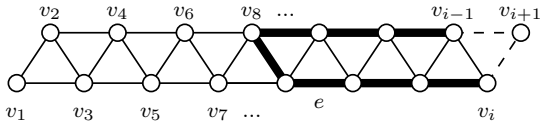
Algoritmer og Datastrukturer 2 (Sommer 2005)

1a

$n = k + 2$.
$m = 2k + 1$.
Kruskal: $O(m \log n) = O((2k + 1) \log(k + 2)) = O(k \log k)$.

1b

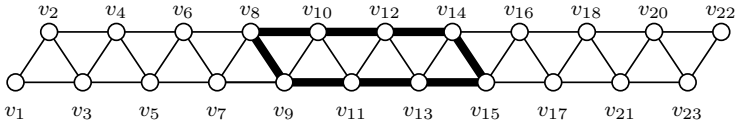
Lad knuder fra venstre-mod-højre være $v_1, v_2, \ldots, v_n$, og lad G_i bestå af delgrafen indeholdende knuderne $v_1, v_2, \ldots, v_i$, dvs. de $i - 2$ første trekanter. Først konstrueres i $O(1)$ tid et MST for G_3 ved at fjerne den tungeste kant fra den venstre trekant. Herefter konstrueres for $i = 3, 4, 5, \ldots, n-1$ et MST for G_{i+1} ud fra et MST for G_i. For G_i huskes den tungeste kant e i MSTet på stien mellem v_{i-1} og v_i. MST for G_{i+1} konstrueres nu ved et af følgende to tilfælde:
a) Hvis $w(v_i, v_{i+1}) \geq \max(w(e), w(v_{i-1}, v_{i+1}))$, så tilføjes (v_{i-1}, v_{i+1}) til MST og e sættes til den tungeste af kanterne e og (v_{i-1}, v_{i+1}).
b) Hvis $w(v_i, v_{i+1}) < \max(w(e), w(v_{i-1}, v_{i+1}))$, så tilføjes (v_i, v_{i+1}) til MST. Hvis $w(v_{i-1}, v_{i+1}) < w(e)$ fjernes e fra MST og (v_{i-1}, v_{i+1}) tilføjes. Til sidst sættes e til (v_i, v_{i+1})



Tid: $O(n)$ da vi bruger tid $O(1)$ på hver af de k trekanter, og $n = k + 2$.

1c

En simpel cykel identificeres entydigt ved knuden v_i længst til venstre og knuden længst til højre v_j. Vi betegner cyklen $C_{i,j}$, f.eks er nedenstående $C_{8,15}$.

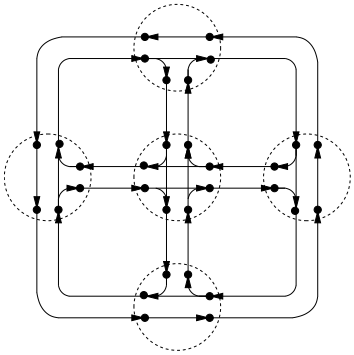


Vi finder den letteste simple cykel ved at kigge på $G_3, G_4, \ldots, G_n$ som i spørgsmål 1b), hvor man husker a) hvad den letteste simple cykel er i G_i, og b) den letteste simple cykel i G_i der indeholder v_i. Den letteste simple cykel i G_{i+1} indeholdende v_{i+1} er enten trekanten $C_{i-1,i+1}$ eller den letteste simple cykel $C_{k,i}$ i G_i indeholdende v_i hvor man fjerne kanten (v_{i-1}, v_i) og tilføjer (v_{i-1}, v_{i+1}) og (v_i, v_{i+1}), i.e. $C_{k,i+1}$. For hver af de nævnte cykeler huskes den første og sidste knude og vægten af cyklen. Disse kan vedligeholdes i $O(1)$ tid når man går fra G_i til G_{i+1}.

2a

$n = st - 4$
$m = 2st - s - t - 4$
Dijkstra: $O(m \log n) = O(st \log(st))$

2b

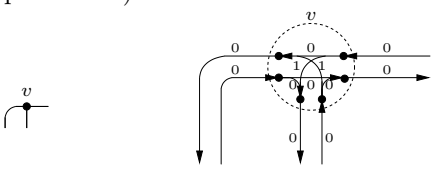


2c

Lav den tilsvarende orienterede graf, hvor knuder af graf 4 erstattes af 8 knuder, jvf. opgave 2b. Udfør et DFS (eller BFS) gennemløb fra hver af de 8 knuder der repræsenterer startknuden u. Hvis og kun hvis mindst ét af de otte gennemløb når en af de 8 knuder der repræsenterer v, så kan v nås fra u uden venstre sving. Da den nye graf har højst $8n$ knuder og højst $12n$ kanter, tager algoritmen tid $O(n)$.

2d

Lav en orienteret graf som i 2c, hvor alle kanter har vægt 0, og tilføj kanter svarende til venstre sving som har vægt 1. Kør Dijkstra's algoritme på den resulterende graf med hver af de 8 knuder repræsenterende startknuden u. Den fundne sti fra en knude repræsenterende u til en knude repræsenterende v med korteste afstand s, vil have netop s venstre sving og være en sti der har færrest mulige venstresving. Da grafen har højst $8n$ knuder og $16n$ kanter tager Dijkstra's algoritme $O(n \log n)$ tid, og den totale tid bliver $O(n \log n)$ (den totale tid kan reduceres til $O(n)$ da alle kanter har vægt 0 eller 1, hvilket medfører at prioritetskøen altid kun kan indeholde to forskellige prioriteter).



3a

for $i = 1$ to n
  $B(i, i) \leftarrow 0$
  $y_{\min} \leftarrow y_i$
  $y_{\max} \leftarrow y_i$
  for $j = i + 1$ to n
    if $y_j < y_{\min}$ then $y_{\min} \leftarrow y_j$
    if $y_j > y_{\max}$ then $y_{\max} \leftarrow y_j$
    $B(i, j) \leftarrow (x_j - x_i) * (y_{\max} - y_{\min})$

Tid: $O(n^2)$

3b

s \ t	1	2	3	4	5	6
1	0	4	8	20	30	36
2	0	0	1	8	9	14
3	0	0	0	1	2	7

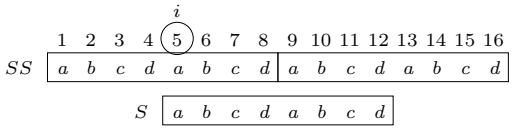
3c

Beregn $B(i, j)$ for alle i, j i tid $O(n^2)$ (spørgsmål a)
for $t = 1$ to n
  $A(1, t) \leftarrow B(1, t)$
for $s = 2$ to k
  $A(s, 1) \leftarrow B(1, 1)$
  for $t = 2$ to n
    $A(s, t) \leftarrow A(1, 1) + B(2, t)$
    for $i = 3$ to t
      if $A(s, t) > A(s - 1, i - 1) + B(i, t)$ then $A(s, t) \leftarrow A(s - 1, i - 1) + B(i, t)$
return $A(k, n)$

Tid: $O(n^2 + n + k * n * n) = O(k * n^2) = O(n^3)$ da vi kan antage $k < n$ (for $k \geq n$ er $A(k, n) = 0$)

4a

Søg efter S i strengen SS vha. KMP algoritmen. Hvis S forekommer på en position i, $2 \leq i \leq |S|$, så er $S = \text{rotation}_{i-1}(S)$, se eksempel hvor $S = \text{rotate}_4(S)$. Da KMP tager tid $O(n+m)$, bliver tiden $O(|SS|) = O(n)$.

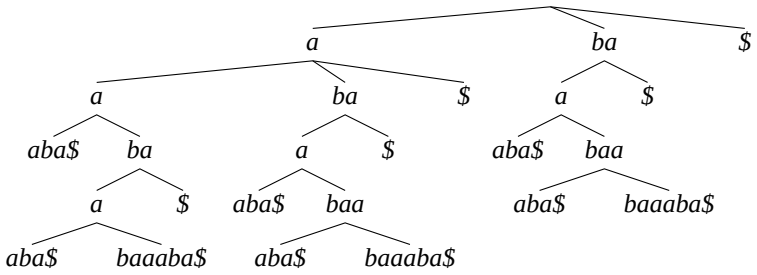


4b

<u>aba</u>aaba
a<u>baa</u>aba
aba<u>aab</u>a
abaa<u>aba</u>

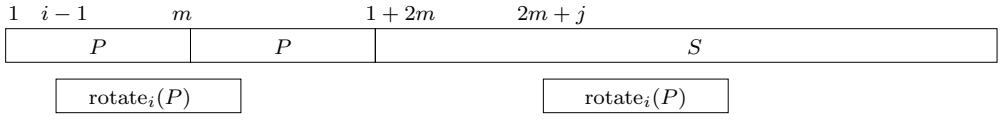
4c

aaaba$
aabaaaba$
aabaabaaaba$
aaba$
abaaaba$
abaabaaaba$
abaabaabaaaba$
aba$
a$
baaaba$
baabaaaba$
baabaabaaaba$ ba$
$



4d

Hvis P forekommer som $\text{rotate}_i(P)$ i S på position j, så forekommer $\text{rotate}_i(P)$ på position $i-1$ og $2m+j$ i $PPS\$$. I suffix-træet findes der så en knude v hvor strengen stavet ned til v er $\text{rotate}_i(P)$, og bladende svarende til suffixerne af $PPS\$$ startende i position $i-1$ og $2m+j$ er i v's undertræ.



For at afgøre om P forekommer som rotation i S bygges suffix-træet for $PPS\$$, og alle knuder annoteres med om der i deres undertræ findes 1) blade der er suffixer startende i position i hvor $2 \leq i \leq m$ og 2) blade der er suffixer startende i position j hvor $1+2m \leq j$. Der returneres at P forekommer som rotation i S hvis der findes en knude der er mareket både 1) og 2) og hvor strengen fra roden ned til knuden har længde $\geq m$. Da annoteringen kan foretages i tid $O(n)$ ved et postorder gennemløb af suffix-træet har vi total tid: $O(2m+n) = O(n)$.

Algoritmer og Datastrukturer 2 (Sommer 2006)

1a

Antal knuder: $n = 2t + 1$.
Antal kanter: $m = 3t$.
Kruskal's algoritme: $O(t \log t)$.

1b

Algoritme: Slet den tungeste kant i hver trekant.
Tid: Hver kant betragtes præcis en gang, dvs tid $O(m) = O(n)$, da $m \leq 2n$.

1c

Algoritme: Find et minimum udspændende træ for terkant-træet uden den ekstra kant vha. algoritmen fra 1b. Lav DFS for at finde stien S fra u til v som ikke indeholder kanten (u, v). Indsæt den ekstra kant (u, v). Fjern den tungest kant fra cyklen der består af (u, v) og S.
Tid: $O(n)$ da 1b og DFS tager tid $O(m) = O(n)$.

2a

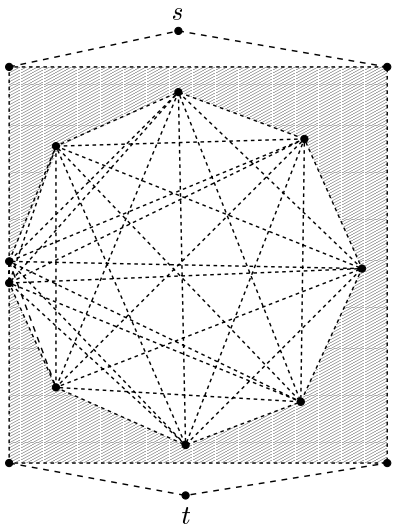
Algoritme: Udfør Dijkstra's algoritme på synlighedsgrafen, med s som kilde og hvor algoritmen anvender et array som prioritetskø.
Tid: $O(m + n^2) = O(n^2)$.

2b

Algoritme: Udfør Dijkstra's algoritme på synlighedsgrafen, med t som kilde og hvor algoritmen anvender et array som prioritetskø. Retuner den robot der har kortest afstand til t.
Tid: Antal knuder $n' = n+k+1$ og antal kanter $m' = O((n+k)^2)$, dvs. tid $O(m'+n'^2) = O((n+k)^2)$.

2c

Ide: Placer $\Theta(n)$ knuder på en cirkel, som alle kan se hinanden. Synlighedsgrafen har så $\Theta(n^2)$ kanter.
I eksemplet er der kun en polygon, men der er $n-4$ polygonpunkter der ligger på en cirkel og som alle kan se hinanden.



3a

k	1	2	3	4	5	6	7	8
$U(k)$	0	9	1	1	2	1	3	2

3b

Algoritme: $U(1) = 0$
for $k = 2$ to n
  $U(k) = U(1) + (x_k - x_1 - W)^2$
  for $i = 2$ to $k - 1$
    $U(k) = \min\{U(k), U(i) + (x_k - x_i - W)^2\}$

Tid: $O(n^2)$, da den indre for-løkke gennemløbes $\leq n$ gange for hvert k.

3c

Udfør algoritmen fra 3b og kald proceduren report(n).

Algoritme: procedure report(k)
if $k = 1$ then
  udskriv x_1
  return
for $i = 1$ to $k - 1$
  if $U(k) = U(i) + (x_k - x_i - W)^2$ then
    report(i)
    udskriv x_i
    return

Tid: $O(n^2)$, da der er $\leq n$ rekursive kald fordi k er aftagende, og for-løkken i et rekursivt kald højest gennemløbes n gange.

4a

streng	position
abacab	4
bacab	5
aacab	-
abcab	13
abaab	1
abacb	8
abaca	4

4b

Algoritme: Hvis $n < m - 1$, rapporter "ingen forekomster". Ellers kør KMP algoritmen med T og P. For $k = 1, 2, \ldots, m$ kør KMP med T og P med det $k$te tegn fjernet. Totalt køres KMP $m + 1$ gange.
Tid: $O(mn)$, da hvert at de $m + 1$ brug af KMP tager tid $O(n)$.

4c

Byg et suffix træ for T. Søg efter P i suffix træet. For $k = 1, 2, \ldots, m$ søg efter P med det $k$te tegn fjernet i suffix træet. Marker de fundne knuder (eller børnene hvor de aktuelle kanter fører hen til). Løb suffix træet igennem og rapporter alle blade hvor der er en markeret forfar.
Tid: $O(n)$ for at konstruere suffix træet, $O(m)$ for hvert af de $m + 1$ søgninger i suffix træet, og $O(n)$ for rapporteringen. Totalt $O(n + m^2)$.

4d

Algoritme: i) Søg efter P i T, ii) for $k = 1, 2, \ldots, m$ søg i T efter P med det $k$te tegn fjernet, ii) for alle par (k, ℓ) hvor $1 \leq k < \ell \leq m$ søg i T efter P med det $k$te og ℓte tegn fjernet. Hver søgning kan laves med enten KMP eller en søgning i et forudberegnet suffix træ for T.
Tid: Ved brug af KMP $O(nm^2)$, eller ved brug af et suffix træ $O(n + m^3)$.

Algoritmer og Datastrukturer 2 (Sommeren 2007)

Opgave 1

Spørgsmål a:

Antal knuder: $n = k \cdot \ell$.

Antal kanter: $m = k \cdot (\ell - 1) + \ell \cdot (k - 1) = 2\ell k - \ell - k$.

Dijkstra's algoritme: $O(k\ell \log(k\ell))$.

Spørgsmål b:

Korteste vej fra P til E:

$$P \to Q \to R \to \hat{S} \to \hat{N} \to O \to J \to E$$

Vægt: $2 + 4 + 3 + 2 + 2 + 3 + 1 = 17$

Spørgsmål c:

Korteste vej uden to sving:

$$P \to Q \to R \to \hat{S} \to N \to I \to \hat{D} \to C \to \hat{B} \to G \to \hat{L} \to M \to N \to \hat{O} \to J \to E$$

Vægt: $2 + 4 + 3 + 2 + 2 + 4 + 1 + 3 + 2 + 1 + 3 + 4 + 2 + 3 + 1 = 37$

Spørgsmål d:

En løsning er at repræsentere hver knude som 12 knuder, der repræsenterer de sidste to skridt taget man er kommet fra til en givet knude. Knuden H bliver til følgende 12 knuder, hvor pilene indikerer de sidste to skridt taget.

$H_1 : \to\to$	$H_2 : \to\uparrow$	$H_3 : \to\downarrow$
$H_4 : \leftarrow\leftarrow$	$H_5 : \leftarrow\uparrow$	$H_6 : \leftarrow\downarrow$
$H_7 : \uparrow\uparrow$	$H_8 : \uparrow\to$	$H_9 : \uparrow\leftarrow$
$H_{10} : \downarrow\downarrow$	$H_{11} : \downarrow\to$	$H_{12} : \downarrow\leftarrow$

Så knuden H_1 repræsenterer at man er gået to gange til højre, og derfor nu kan gå hvorhen man vil. Lad O være knuden over H, U knuden under H, L knuden til venstre for H, og R knuden til højre for H. Fra H_1 kan man gå til O_2 man kan gå til R_1 og til U_3. Fra knuden H_2 kan man gå til O_7. Fra knude H_3 kan man gå til U_{10}. Og så fremdeles. Hvis den nederste venstre knude betegnes s så findes en kortest sti ved at køre Dijkstra's algoritme med s_1 og s_7 som start knuder. Udførselstid for Dijkstra på denne graf bliver $O(\ell k \log(\ell k))$ da hver knude skaleres til $O(1)$ knuder.

Opgave 2

Spørgsmål a:

$$V_1 = \{A, E, J, K\} \quad V_2 = \{B, C, D, F, G, H, I\}$$

Spørgsmål b:

Vælg en tilfældig knude s. Lav DFS gennemlæøb af grafen startende i s. Lad V_1 være knuder med en lige dybde i DFS træet og V_2 knuder med en ulige dybde i DFS træet. Grafen er todelt hvis og kun hvis der ikke findes en kant der forbinder to knuder med lige dybde i DFS træet eller to knuder med ulige dybde. Løses med DFS i tid $O(m)$.

Opgave 3

Spørgsmål a:

Minimum udspændende træ:

$$(B, C), (A, C), (F, H), (D, F), (B, D), (G, H), (A, I), (J, K), (E, G), (E, J)$$

Vægt: 56

Spørgsmål b:

Givet $G = (V, E)$ og $MST(G)$. Hvis $e \notin MST(G)$ gøres intet. Ellers fjern e fra $MST(G)$. Lav DFS fra en knude u i $MST(G)$ og marker alle knuder der findes med V_1. Lad V_2 være resten. V_1 og V_2 definerer et snit(opdeling). Skan E og find kant e' mellem V_1 og V_2 med mindst vægt. Indsæt e' i $MST(G)$. Udførselstid $O(m)$.

Opgave 4

Spørgsmål a:

Tabel:

$L(i,j)$	0	1	2	3
0	0	1	2	3
1	1	2	3	3
2	2	2	3	4
3	3	3	4	4
4	4	4	4	5

Spørgsmål b:

Følgende pseudkode skulle du:

```
Maketable(S,T)
  for i=0 to m
     for j=0 to n
        if (i = 0) L[i,j]  := j
        else if (j = 0) L[i,j] := i
        else if (x_i != y_j) L[i,j] := 1 + min{L[i-1,j],L[i,j-1]}
        else L[i,j] := 1 + L[i-1,j-1]
  return L[m,n]
```

Udførselstid for Maketable: $O(n \cdot m)$

Spørgsmål c:

Følgende metode tager tabellen L lavet i opgave b: og tallene n, m.

```
Print(L,i,j)
  if (i = 0 and j = 0) return
  if (i = 0) write(y_1,...y_j) return
  if (j = 0) write(x_1,...x_i) return
  if (x _i != y_j)
     if (1 + L[i-1,j] = L[i,j])
        Print(L,i-1,j)
        write(x_i)
     else
        Print(L,i,j-1)
        write(y_j)
  else
     Print(L,i-1,j-1)
     write(x_i)
```

Udførselstid for Print(L,m,n): $O(n + m)$

Samlet udførselstid: $O(n \cdot m)$

Opgave 5

Spørgsmål a:

Suffixerne i sorteret orden:

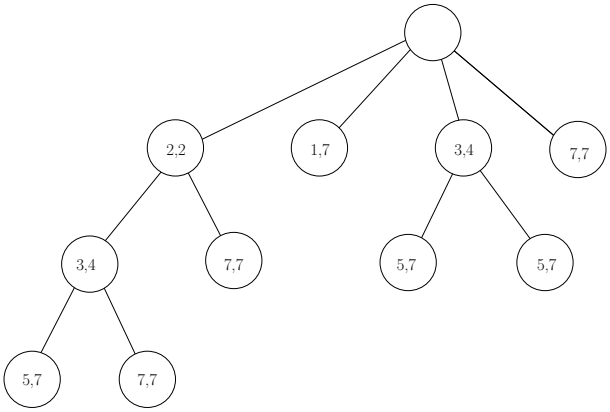
Streng	Start Index
ANANAS	2
ANAS	4
AS	6
BANANAS	1
NANAS	3
NAS	5
S	7

Det giver suffix array:

2	4	6	1	3	5	7

Spørgsmål b:

Suffix-træ:



Spørgsmål c:

To løsninger:

1) Den letteste. Iterer over alle delstrenge af S af længde k. Der er $n-k+1$ af dem. For hver kør Knuth-Morris-Pratt algorithmen og tæl alle forekomster. Returner den der forekommer flest gange. Udførselstid: $O(n^2)$.

2) Den hurtigste. Lav suffixtræ for S i $O(n)$ tid. For alle knuder i træet hvor stien fra roden til og med knuden repræsenterer en streng af længde k eller mere. Tæl antal blade i undertræet. Returner for den knude, der har flest blade under sig, de k første tegn langs stien fra roden til knuden. Alt dette kan klares i et konstant antal lineære gennemløb af træet. Udførselstid $O(n)$.