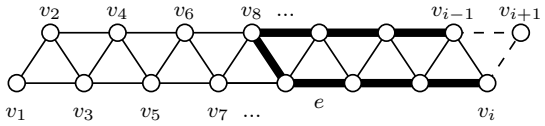
# Algoritmer og Datastrukturer 2 (Sommer 2005)

## 1a

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

## 1b

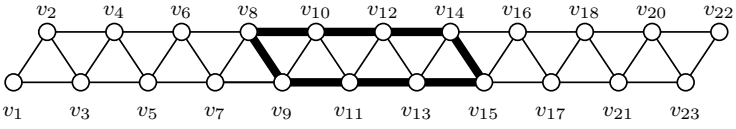
Lad knuder fra venstre-mod-højre være $v_1, v_2, \ldots, v_n$, og lad $G_i$ bestå af delgrafen indeholdende knuderne $v_1, v_2, \ldots, v_i$, dvs. de $i - 2$ første trekanter. Først konstrueres i $O(1)$ tid et MST for $G_3$ ved at fjerne den tungeste kant fra den venstre trekant. Herefter konstrueres 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 $C_{8,15}$.

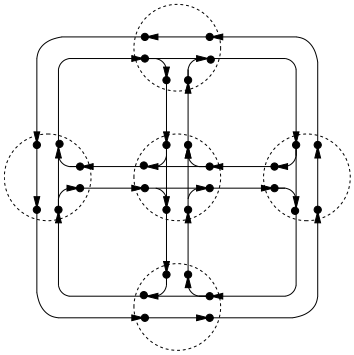



Vi finder den letteste simple cykel ved at kigge på $G_3, G_4, \ldots, G_n$ som i spørgsmål 1b), hvor man husker a) hvad den letteste simple cykel er i $G_i$, og b) den letteste simple cykel i $G_i$ der indeholder $v_i$. Den letteste simple cykel i $G_{i+1}$ indeholdende $v_{i+1}$ er enten trekanten $C_{i-1,i+1}$ 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

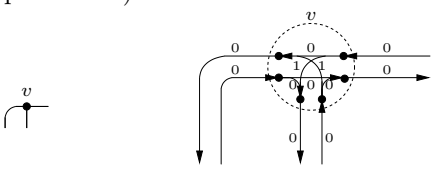


## 2c

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 |

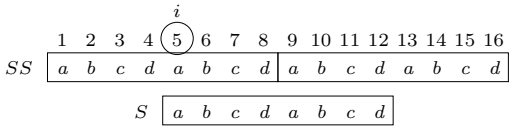
## 3c

*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 $O(|SS|) = O(n)$.

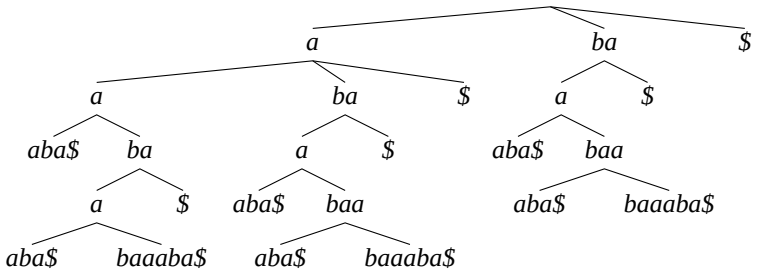



# 4b

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

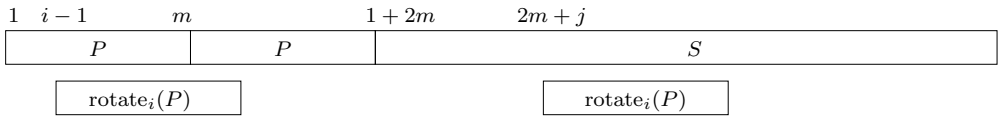
# 4c

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: $O(2m+n) = O(n)$.