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

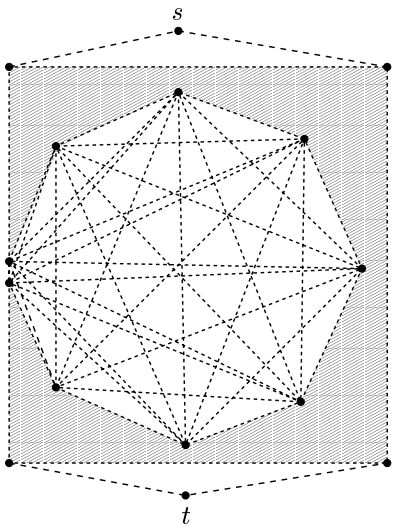
Algoritme: Udfør Dijkstra's algoritme på synlighedsgrafen, med s som kilde og hvor algoritmen anvender et array som prioritetskø.
Tid: $O(m + n^2) = O(n^2)$.

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 $O(n + m^3)$.