DD1352 Algoritmer, datastrukturer och komplexitet, hösten 2013
Lösningsförslag till mästarprov 2: komplexitet

1. **Hur kort kan texten bli om man blandar om den?**
   Egen övning.

2. **UKÄ-problemet**
   Vi kan verifiera en ja-lösning till beslutsproblemet i polynomisk tid på följande sätt. Låt lösningen vara en delmängd av exjobben av storlek k, vilket ska motsvara dom utvalda exjobben. Verifikatorn kollar först att alla exjobb i lösningen är med i indata och sedan för varje mål att minst ett av exjobben i lösningen har bristande kvalitet för det målet. Detta görs enkelt i polynomisk tid.

   För att visa NP-svårighet kan vi enklast reducera hörntäckning (ett exjobb per hörn och ett mål per kant), mängdtäckning (ett exjobb per delmängd och ett mål per element) eller 3-CNF-SAT (två exjobb per variabel, ett mål per variabel och klausul).

   Låt oss till exempel visa en reduktion av 3-CNF-SAT till UKÄ.

   Idén är att vi skapar två exjobb för varje variabel där bara ett av varje par exjobb ska kunna vara med i delmängden. Detta kan vi ordna genom att ha ett mål för varje variabel för vilket bara dessa två exjobb har bristande kvalitet. Vilket av exjobben i paret som är med motsvarar variabelns värde i den satisfierande tilldelningen. För varje klausul har vi ett mål som har bristande kvalitet bara för dom tre exjobb som motsvarar literalerna som är med i den klausulen.

   3CNFSAT$(c_1, \ldots, c_q)$ = // Anta att variablerna heter $x_1, \ldots, x_p$
   $m \leftarrow p + q$
   $k \leftarrow p$
   for $i \leftarrow 1$ **to** p **do**
   $x_i^+ \leftarrow \{\text{Hög}, \text{Hög}, \ldots, \text{Hög}\}$ // $p+q$-vektor med bara hög kvalitet
   $x_i^- \leftarrow \{\text{Hög}, \text{Hög}, \ldots, \text{Hög}\}$
   $x_i^+[i] \leftarrow$ Bristande; $x_i^-[i] \leftarrow$ Bristande
   for $j \leftarrow 1$ **to** q **do**
   // anta att $c_j = x_a^A \vee x_b^B \vee x_c^C$ där A är $+$ om x_a förekommer onegerat och $-$
   // om x_a förekommer negerat i klausulen och motsvarande för B och C.
   $x_a^A[p+j] \leftarrow$ Bristande; $x_b^B[p+j] \leftarrow$ Bristande; $x_c^C[p+j] \leftarrow$ Bristande
   return UKÄ$(m, \{x_i^+, x_i^-\}_1^p, k)$

   Bevis av reduktionen: Anta att det finns en satisfierande variabeltilldelning, visa att det finns en delmängd med p exjobb där minst ett exjobb har bristande kvalitet för varje mål.

   För varje i, ta i delmängden med det exjobb av x_i^+ och x_i^- som motsvarar värdet på variabeln x_i i variabeltilldelningen (dvs ta x_i^+ om x_i är sann och x_i^- om x_i är falsk). Eftersom det finns p variabler så består delmängden av p exjobb. Eftersom ett av varje par x_i^+ och x_i^- är med i delmängden så finns det ett exjobb som har bristande kvalitet för vart och ett av dom p första målen. Eftersom varje klausul c_j är satisfierad så är en av literalerna i c_j sann, vilket innebär att det motsvarande exjobbet är med i delmängden, och det exjobbet har bristande kvalitet för mål $p + j$. Därmed har för varje mål minst ett exjobb bristande kvalitet.

   Åt andra hållet: Anta att det finns en delmängd med p exjobb som uppfyller villkoret att det för varje mål finns minst ett exjobb som har bristande kvalitet. För varje mål $i \in [1..p]$ innebär det att antingen exjobb x_i^+ eller x_i^- är med i delmängden, eftersom bara dessa exjobb har bristande kvalitet för mål i. Och eftersom p är antalet variabler så kan bara en av x_i^+ och x_i^- vara med i delmängden för varje i. För varje mål $p + j \in [p + 1..p + q]$ innebär det att antingen exjobb x_a^A, x_b^B eller x_c^C är med i delmängden, förutsatt att $c_j = x_a^A \vee x_b^B \vee x_c^C$.

Skapa en variabeltilldelning där variabel x_i sätts till sann om x_i^+ är med i delmängden och falsk om x_i^- är med i delmängden. Det innebär att varje klausul måste vara satisfierad, dvs det är en satisfierande variabeltilldelning.

Reduktionen går i linjär tid i formelns storlek. Alltså är det en polynomisk Karpreduktion.

Problemet är alltså NP-fullständigt.

3. **Konstruktion av minimal mängd exjobb**
Minimala antalet exjobb måste ligga mellan 1 och totala antalet exjobb. Binärsök därför mellan 1 och n.

OptUKÄ$(m, \{e_1, \ldots, e_n\}) =$
    min $\leftarrow 1$; max $\leftarrow n$
    while min $<$ max **do**
        INV=lösningen finns i intervallet $[min..max]$
        $M \leftarrow \lfloor(\min + \max)/2\rfloor$
        if UKÄ$(m, \{e_1, \ldots, e_n\}, M)$ **then** max $\leftarrow M$ **else** min $\leftarrow M + 1$
    return min

Binärsökningen gör $\lceil \log n \rceil$ stycken anrop av UKÄ.

Vi konstruerar en optimal lösning genom att först ta reda på det optimala antalet exjobb och sedan ta bort ett exjobb i taget och se om det fortfarande finns en lösning med optimalt antal exjobb. Om det finns det så finns det för varje mål minst ett exjobb med bristande kvalitet och vi fortsätter att ta bort ytterligare exjobb.

Construct-UKÄ$(m, \{e_1, \ldots, e_n\}) =$
    $opt \leftarrow$OptUKÄ$(m, \{e_1, \ldots, e_n\})$
    $S \leftarrow \emptyset$
    for $i \leftarrow 1$ **to** n **do**
        INV=$\exists$ lösning bestående av exjobben i S och $opt - |S|$ av exjobben $\{e_i, \ldots, e_n\}$
        if UKÄ$(m, S \cup \{e_{i+1}, \ldots, e_n\}, opt)$ =nej **then**
            $S \leftarrow S \cup e_i$
    return S

$O(n + \log n)$=O(n) anrop av UKÄ görs, så tidskomplexiteten blir $O(nB(n, m))$.

Korrektheten för algoritmen inses lätt med hjälp av invarianterna.