DD2350 Algoritmer, datastrukturer och komplexitet, hösten 2019
Lösningsförslag till mästarprov 2: komplexitet

1. **Nattparkering av långt godståg**
   Nattparkeringsproblemet som beslutsproblem:
   INDATA: Vagnslängder $v[1..n]$, antal spår d och mål m, där alla tal i indata är positiva heltal och $d < n$.
   FRÅGA: Går det att parkera alla vagnarna på d spår av längd m så att det står minst en vagn på varje spår?

   Vi bevisar att problemet ligger i NP genom att visa hur man i polynomisk tid kan verifiera en ja-instans givet en vagnsparkering $P[1..n]$ genom att kontrollera att alla villkor är uppfyllda. $P[i]$ är numret på det spår som vagn i ska parkeras på.

   VerifyNightParking($v[1..n], d, m, P[1..n]$) =
       $level \leftarrow 1$; $x \leftarrow 0$
       for $t \leftarrow 1$ **to** d **do** $used[t] \leftarrow 0$
       for $i \leftarrow 1$ **to** n **do**
           if $P[i] < 0$ **or** $P[i] > d$ **then return false**
           $used[P[i]] \leftarrow used[P[i]] + v[i]$
       for $t \leftarrow 1$ **to** d **do**
           if $used[t] = 0$ **or** $used[t] > m$ **then return false**
       return true

   Verifieringen går i linjär tid med både enhetskostnad ($O(n)$) och bitkostnad.

   Vi visar sedan att problemet är NP-svårt genom att reducera problemet delmängdssumma till nattparkeringsproblemet.

   Vi gör reduktionen i två steg. Först Karpreducerar vi delmängdssumma till problemet mängdpartitionering (lika vikt) på samma sätt som i övningsmästarprov 2. Därefter Karpreducerar vi mängdpartitionering till nattparkeringsproblemet på följande sätt:

   Partition($v[1..n]$) =
       $d \leftarrow 2$
       $\sigma \leftarrow \sum_{i=1}^{n} v_i$
       $m \leftarrow \sigma/2$
       return NightParkingPossible($v[1..n], d, m$)

   Vi låter alltså vagnslängderna vara samma som dom positiva heltalen i indata till mängdpartitioneringsproblemet och frågar oss om det går att parkera dessa vagnar på två spår som var och ett har plats för halva tågets längd.

   Reduktionen är uppenbart polynomisk (linjär i indatas längd). Låt oss visa att den är korrekt, det vill säga att det existerar en mängdpartitionering om och endast om det existerar en nattparkering på två spår.

   Anta att det finns en mängdpartitionering. Parkera då vagnarna som motsvarar talen i ena partitionen på ena spåret och övriga vagnar på andra spåret. Vagnarna kommer då att få plats på två spår eftersom längderna på spåren är halva tågets längd. Därmed finns det en giltig nattparkering.

   Åt andra hållet, anta att det finns en nattparkering på två spår. Eftersom spårens längd är halva tågets längd så måste båda spåren vara helt fyllda. Summan av talen som motsvarar

vagnarna på första spåret är då lika med summan av övriga tal (som ju motsvarar vagnarna på andra spåret). Därför finns det en mängdpartitionering.

Att Karpreduktionen av delmängdssumma till mängdpartitionering är polynomisk och korrekt visas i övningsmästarprov 2. En komposition av två polynomiska Karpreduktioner är fortfarande en polynomisk Karpreduktion.

Därmed är NP-fullständigheten bevisad.

2. **Spel med färgkort**
Betrakta beslutsproblemet SPEL definierat som i uppgiften. Vi visar att SPEL är NP-fullständigt genom att visa att SPEL ligger i NP och att SPEL är NP-svårt.

SPEL ligger i NP

Om vi får en lösning som anger hur ett parti bestående av s steg ska spelas (med information om spelat kort och nästa spelare för varje steg) så kan vi på polynomisk tid kontrollera om partiet är korrekt (att båda inblandade spelare har kort av aktuell färg vid varje steg) och om den totala poängen är minst p. Därmed har vi visat att SPEL ligger i NP.

SPEL är NP-svårt

För att visa att SPEL är NP-svårt så kombinerar vi två polynomiska Karpreduktioner som utgår från det känt NP-fullständiga problemet HAMILTONCYKEL:

HAMILTONCYKEL $\leq$ HAMILTONSTIG $\leq$ SPEL

där HAMILTONSTIG definieras på följande sätt:

Inmatning: En oriktad graf G = <V, E> och ett hörn S i grafen.
Fråga: Finns det en stig som startar i S och besöker varje hörn i G precis en gång?

HAMILTONCYKEL $\leq$ HAMILTONSTIG

Denna reduktion beskrivs i detalj i övning 9.

HAMILTONSTIG $\leq$ SPEL

Givet grafen G så konstruerar vi ett spel med n^2 färger och n spelare som har n kort var, där n är antalet hörn i G. Korten är fördelade så att två spelare har kort av samma färg om och endast om det finns en kant mellan motsvarande hörn i grafen. Vi låter antalet steg s vara $n-1$ och den totala poängen p vara $n(n-1)$.

Låt oss först visa att om det finns en hamiltonsk stig med start i S i G, så går det att uppnå poängen $n(n-1)$ i $n-1$ steg. Partiet spelas i samma ordning som hörnen i stigen och eftersom man hela tiden besöker nya spelare så får man poängen n i vart och ett av de $n-1$ stegen.

Nu återstår att visa att om det finns ett parti med $n-1$ steg som ger totalpoäng $n(n-1)$ så finns det också en hamiltonsk stig i G. Detta är klart eftersom den maximala poängen i varje steg är n och denna poäng kan endast uppnås när en spelare deltar för första gången. För att få $n(n-1)$ poäng i $n-1$ steg måste alltså samtliga spelare vara med. Spelordningen motsvarar därmed en hamiltonsk stig.

Reduktionen har tidskomplexiteten $O(n^2)$, vilket är polynomiskt i indatas storlek.

3. **Konstruktion av nattparkering av långt godståg**
Vi antar att det finns en algoritm NightParkingPossible$(d, v[1..n], m)$ som svarar på nattparkeringsproblemet.

Först tar vi fram en övre och undre gräns för parkeringsspårens längd och sedan gör vi en binärsökning mellan dessa med hjälp av anrop till NightParkingPossible för att hitta den minimala längden.

Som övre gräns för parkeringsspårens längd använder vi hela tågets längd och som undre gräns använder vi hela tågets längd delat med antalet spår.

```
MinNightParking(d, v[1..n]) =
    maxm ← Σ_{i=1}^{n} v[i]
    minm ← ⌈maxm/d⌉
    while min < max do
        inv lösningen finns i intervallet [min..max]
        m ← ⌊(minm + maxm)/2⌋
        if NightParkingPossible(d, v, m) then maxm ← m else minm ← m + 1
    return minm
```

När vi vet minimala längden för parkeringsspåren slår vi ihop vagnar två och två till vagnar som har summan av dom två vagnslängderna som längd. Detta gör vi om och om igen på ett sätt som gör att en nattparkering fortfarande är möjlig. När bara d (sammanslagna) vagnar återstår är vi klara. Beståndsdelarna av dessa sammanslagna vagnar utgör då en optimal nattparkering.

```
optm ← MinNightParking(d, v[1..n])
assert Optimala parkeringsspårslängden är optm
for i ← 1 to n do sol[i] ← 0
for t ← 1 to d do
   inv Det finns en optimal tågparkering som parkerar vagnarna på spår 1 till t − 1
       på det sätt som anges av sol där längden av vagnarna på spår k är used[k].
   used[t] ← 0
   for i ← 1 to n do
      inv Ovanstående invariant utökad med att vagnarna med index j < i där sol[j] = t
          kan parkeras på spår t i en optimal tågparkering och har längden used[t] tillsammans.
          Det finns dessutom ingen ytterligare vagn med index j < i som kan parkeras på spår t.
      if sol[i] = 0 and used[t] + v[i] ≤ optm then
         sol[i] ← t
         used[t] ← used[t] + v[i]
         current ← t
         // Alla vagnar som placerats ut (dvs har sol[j] ≠ 0) ingår nu i used[1..t].
         for j ← 1 to n do
            if sol[j] = 0 then
               current ← current + 1
               used[current] ← v[j]
         // Alla vagnar ingår nu antingen i used[1..t] eller är kopierade till used[t + 1..current].
         if not NightParkingPossible(d, used[1..current], optm) then
            sol[i] ← 0
            used[t] ← used[t] − v[i]
assert Det finns en optimal tågparkering som parkerar vagnarna på det sätt som anges av sol.
return sol[1..n]
```

Algoritmen tar polynomisk tid och gör $\log(\sigma) + dn$ anrop av NightParkingPossible, där σ är summan av alla vagnars längd. $\log(\sigma)$ är mindre än antalet bitar i indata och $d < n$, varför antalet anrop är polynomiskt i indatas längd.

För varje anrop av NightParkingPossible i binärsökningen görs ett konstant antal taloperationer och för varje anrop i konstruktionen görs en kopiering av olika tal i indata, vilket i båda fallen innebär linjär tid i indatas längd för varje anrop. Tidskomplexiteten utöver anropen är alltså polynomisk i indatas längd, närmare bestämt $O(n \log(\sigma)(\log(\sigma) + dn))$.

Korrektheten för algoritmen inses på följande sätt. Binärsökningen efter optimala värdet är en standardalgoritm som bevisas med en invariant som säger att det sökta värdet finns mellan dom aktuella gränserna i binärsökningen.

Efter binärsökningen vet vi att det finns en lösning med värdet *optm* men ingen lösning med värdet $optm - 1$. Att konstruktionen av $sol[1..n]$ är korrekt visas enkelt med hjälp av invarianterna i for-slingorna.