

Algoritmer, datastrukturer och komplexitet, hösten 2019

Lösningsförslag till övningsmästarprov 1

Kortaste texten

Den självklara giriga algoritmen placerar orden på raderna uppifrån och ned och byter rad först när ett ord inte får plats på samma rad.

```
TextPacking( $\{w_1, \dots, w_n\}, len$ ) =  
  if  $n = 0$  then return 0  
  line  $\leftarrow 1$   
  pos  $\leftarrow w_1$   
  for  $i \leftarrow 2$  to  $n$  do  
    if  $pos + 1 + w_i > len$  then  
      line  $\leftarrow line + 1$   
      pos  $\leftarrow w_i$   
    else pos  $\leftarrow pos + 1 + w_i$   
  return line
```

För att bevisa att en girig algoritm är optimal brukar man antingen visa att algoritmen i varje läge ligger före varje optimal algoritm (se beviset för att girig intervallschemaläggning är optimalt i avsnitt 4.1 i Kleinberg-Tardos) eller visa att varje optimal lösning kan transformeras till den giriga lösningen (se beviset för att Kruskals och Prims algoritmer ger optimala spännande träd i avsnitt 4.5 i Kleinberg-Tardos eller föreläsningsanteckningarna till föreläsning 12). Båda metoderna kan användas för att visa att girig textpackning är optimal. Låt oss genomföra den andra metoden.

Om $n = 0$ returnerar algoritmen 0 vilket är optimalt.

Anta att U_k är en optimal lösning (dvs utplacering av ord på optimalt antal rader) som överensstämmer med algoritmens utplacering av dom k första orden. Anta att ord $k + 1$ i U_k har placerats på rad r_U med början i position p_U medan algoritmen placerat ordet på rad r_A med början i position p_A . Eftersom föregående ord placerades på samma sätt så vet vi att $r_U \geq r_A$ (för om ord $k + 1$ hade fått plats på samma rad som ord k så hade algoritmen gjort det) och att om $r_U = r_A$ så är $p_U \geq p_A$ (för algoritmen har placerat ord $k + 1$ så långt till vänster som möjligt på raden).

Om $r_U = r_A$ och $p_U > p_A$ så kan vi skjuta ordet till vänster så att det börjar på p_A . Denna modifierade U_k är då fortfarande en optimal lösning eftersom antalet rader inte ändrats och inga ord överlappar på grund av förskjutningen.

Om $r_U > r_A$ så måste $r_U = r_A + 1$ (för det kan aldrig vara optimalt att ha en tom rad i texten). I så fall kan vi flytta upp ord $k + 1$ till rad r_A med start i position p_A , för ordet fick uppenbarligen plats där. Denna modifierade U_k är då fortfarande en optimal lösning eftersom antalet rader inte kan ha ökat och inga ord överlappar på grund av omplaceringen.

På detta sätt kan vi alltså alltid modifiera U_k till en lösning U_{k+1} , alltså en optimal lösning som överensstämmer med algoritmens utplacering av dom $k + 1$ första orden.

Med induktion över k från 0 till n så kommer vi fram till att algoritmens utplacering av orden måste vara optimal med avseende på antalet rader.

Tidskomplexiteten är $O(n)$ eftersom algoritmen bara innehåller en enda slinga och den går $n - 1$ varv där varje varv tar konstant tid med enhetskostnad.