godk.datum vitsord

bedömare

Automatisk typsättning

Johan Brunberg

Helsingfors den 9 maj 2005

HELSINGFORS UNIVERSITET
Institutionen för datavetenskap

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Matematisk-naturvetenskapliga fakulteten	Institutionen för datavetenskap

Tekijä — Författare — Author
Johan Brunberg

Työn nimi — Arbetets titel — Title

Automatisk typsättning

Oppiaine — Läroämne — Subject
datavetenskap, övningsarbete för kursen Att skriva vetenskaplig text

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
avhandling	maj 2005	15 sidor

Tiivistelmä — Referat — Abstract

Utvecklingen kring World Wide Web är sorglig med tanke på typografin av det enorma antalet webbsidor i dagens Internet. Både det traditionella typografiska kunnandet, vars mål är att göra text så läsbar som möjligt, och metoderna som har skapats för automatisk typsättning verkar ha blivit bortglömda. Denna avhandling försöker i kompakt form presentera de centrala algoritmerna och principerna bakom det utsökta typsättningssystemet TEX, som utvecklades av matematikern Donald Knuth på 1980-talet. Vi betraktar ritning av bokstavsbilder och sätt att beskriva typsnitt i digital format och studerar detaljerna av TEXs rad- och ordbrytningsalgoritmer.

ACM Computing Classification System (CCS):
I.7.2 [Document Preparation]

Avainsanat — Nyckelord — Keywords
typsättning, radbrytning, avstavning, TEX, METAFONT

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Innehåll

1 Inledning

> En av de viktigaste arbetsfaserna när material av text förbereds för tryckning eller framvisning är uppgiften att bryta långa stycken i enskilda rader. När arbetet utförs väl, märker läsaren inte att orden har egenmäktigt brutits isär och placerats i en något stel och onaturlig rektangelram, men om det utförs dåligt, blir läsaren distraherad av dåliga brytningar som avbryter tankegången.
>
> – *ur inledningen till* Breaking Paragraphs into Lines *[KP81]*

Typsättning är en av de otacksamma arbeten i vilka närmast misslyckanden observeras. Målet är att göra texten så lätt att läsa som möjligt. God sättning märks inte, dels för att ett kriterium för den är att den inte skall distrahera, dels för att vi är vana med att majoriteten av alla texter i tryckt form som vi ännu idag omges av är välsatta. Före World Wide Web och hypertexttiden klarade dåligt satta texter bara undantagsvis nå en bredare publik.

Att göra en omärklig layout kan vara överraskande arbetsdrygt. I denna avhandling studerar vi hur några delområden i typsättning kan automatiseras. Avhandlingen förankrar sig till typsättningssystemet TEX, som matematikern, professor i datavetenskap vid Stanford universitet Donald Knuth utvecklade under 1980-talet. Speciellt för typsättning av matematisk text är TEX fortfarande det mest kompletta programmet som finns.

Det första problemet inom automatisk typsättning är utformingen av bokstäverna. En dator behandlar bokstäver i form av binär digital information och förvandlingen till igenkännbara bokstavsfigurer bestående av de små punkter eller pixlar som skrivare och skärmar arbetar med är inte trivial. När det gäller typsnitt är målet att ge professionella mänskliga typografer möjligheten och redskaperna till att beskriva egenskaperna av typsnitt så exakt som möjligt i en form som datorer kan använda [Knu86]. Det här är ett område där goda lösningar är så komplicerade att de lätt blir inkompatibla med varandra, vilket, som vi kommer att se, tyvärr är fallet med TEXs METAFONT och det nyare PDF-dokumentformatet och dess Type 1-fonter.

Utvecklandet av teknikerna bakom typsnitt leds idag av kommersiella företag och vi behandlar ämnet ytligt. Tyngdpunkten för avhandlingen ligger i algoritmerna med vilka TEX väljer ställena för rad- och ordbrytning.

Vi har vant oss vid att läsa text med jämna kanter, men att få orden att fylla ut raderna vackert är inte alltid lätt. När man sätter en tegelstensroman kunde man önska sig att åtminstone flertalet av styckena kunde brytas automatiskt. Avstavning försämrar läsbarheten men är ofta nödvändig. Vi kommer att se att en god radbrytningsalgoritm kan minska behovet att avstava ord och kan därför till och med ge bättre resultat än brytning för hand.

Det är inte möjligt att täcka alla delar av automatisk typsättning inom ramarna för denna avhandling. Vi kommer till exempel inte att behandla placering av figurer och tabeller, sammanställning av förteckningar och dylika funktioner vilka makropaketet LaTeX [Lam04] automatiserar.

Typografiska begrepp som avhandlingen använder följer dem använda i Typografisk handbok [Hel94]. I kapitel 2 använder vi ordet *font* i betydelsen information om utformningen av tecken i en teckenuppsättning.

2 Bokstäverna

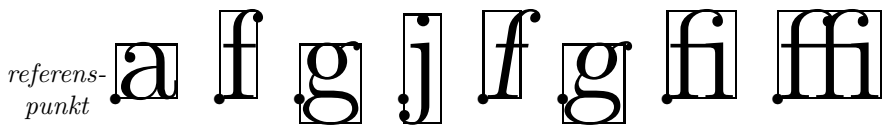
2.1 Box- och limmodellen

Vi börjar vår studie i typsättning genom att bekanta oss med komponenterna som TeX bygger de typsatta sidorna av. Två viktiga begrepp som TeX arbetar med är *boxar* och *lim* [Knu84].

En *box* är en platshållare för färdigt satt material. Den har formen av en rektangel, vars dimensioner är kända. De minsta boxarna reserverar utrymme för enskilda bokstäver eller symboler. Av de mindre boxarna och andra element såsom lim bygger TeX större boxar som till exempel innehåller hela rader av text. Figur 1 visualiserar TeXs syn på en rad typsatt text: TeXs uppgift är räkna ut de rätta platserna för boxarna. Bokstäverna och symbolerna som boxarna håller plats för ritas ut senare av en skild utskriftsrutin.

En rad typsatt text.

Figur 1: En rad typsatt text



Figur 2: Boxar med bokstäver

Lim (glue) används i motsats till vad namnet antyder till att hålla boxar isär. Lim sätts mellan ord, rader och stycken av text och limenheter beskrivs av tre egenskaper: den naturliga vidden, krympbarheten och töjbarheten. Genom att skilja åt orden på raderna med töjbart lim kan TEX fylla ut raderna till en jämn bredd så att man får en rak högerkant.

Innehållet av en box behöver paradoxalt nog inte hålla sig innanför boxens ramar. Det är tvärtom helt vanligt att bokstäver sticker ut ur sina boxar. De kursiverade bokstäverna *f* och *g* i figur 2 gör det tydligt. De gör det för att kompensera för att boxarna inte kan luta. Om boxarna vore större, skulle TEX ge

En rad typsatt text i falsk kursiv.

i stället för

En rad typsatt text i falsk kursiv.

Speciellt bokstäverna *r* och *a*, *y* och *p* samt *f* och *a* kommer här på den övre raden för långt från varandra.

Som figur 2 antyder har varje box en *referenspunkt* (reference point). Bokstavsboxar radas normalt tätt intill varandra så att deras referenspunkter kommer på samma baslinje. Vissa bokstavskombinationer kräver speciell uppmärksamhet. De breda boxarna för bokstäverna A och V ger till exempel i de flesta fallen rätt bokstavsmellanrum, men när V följer A föredrar vi AV framför AV. Det rätta bokstavsmellanrummet fås här genom att flytta boxen för bokstaven V lite ovanpå boxen för A. Förutom att boxarnas innehåll inte behöver hålla sig innanför boxen kan boxarna alltså också överlappa varandra.

Informationen om bokstavsboxarnas dimensioner och rätta placering i förhållande till varandra inom ett typsnitt får TEX från en metrikfil (*font metric file*) för typsnittet [Knu86]. Metrikfilerna instruerar TEX också till exempel att byta ut den

besvärliga bokstaven *f* och en påföljande *i*, vars prick *f*:t riskerar kollidera med, mot *ligaturen fi*. En ligatur ritas ut som ett enda tecken, så ordet *fiffig* sätts ihop av endast tre tecken: *fi*, *ffi* och *g*. Ligaturer kan naturligtvis ersättas bara för de bokstavskombinationer som designarn för typsnittet har definierat sådana. En användning för ligaturer i TEXs standardtypsnitt är ersättningen av två enkla diviser (--) med ett tankstreck (–) [Knu84]. Tre diviser ger ett längre streck (—), som används i engelsk typografi, men fyra läser TEX som det längre strecket och en ensam divis (—-).

Faktumet att vissa bokstäver sticker ut ur sina boxar kan ställa till med problem när man byter från ett typsnitt till ett annat, eftersom en fontmetrikfil inte kan innehålla information om hur tecken i typsnittet den beskriver skall placeras i förhållande till tecken i det andra typsnittet som den inte vet någonting om. För en kursiverad bokstav '*f*' i enkla citationstecken ger TEX '*f* om man inte för hand sätter in en korrigerande *kerningsinstruktion* (explicit kern) [Knu84]. Kerningsinstruktioner kan betraktas som lim utan krympbarhet eller töjbarhet, men medan lim mellan bokstavsboxar och rader anger tillåtna radbrytnings- och sidbrytninsställen gör kerningsinstruktioner det inte.

För att man inte skall behöva pröva sig fram till den rätta kerningen berättar fontmetrikfilerna för varje tecken i typsnittet hur mycket det sticker ut ur sin box och TEX har ett enkelt kommando som sätter in den motsvarande kerningsinstruktionen efter tecknet. Makropaketet LATEX har också kursiveringskommandon som försöker enligt behov sätta in den korrigerande kerningen automatiskt [Lam04]. Tumregeln är att en kerning behövs när man byter från en kursiverad stil till en okursiverad, om det följande tecknet inte är en punkt eller ett kommatecken [Knu84].

2.2 TEX och typsnitt

För TEX är en bokstav bara en box med en viss storlek. Den radar boxarna bredvid och ovanpå varandra enligt uppgifter och instruktioner den har fått, men den vet inte hur innehållet i en box skall se ut. Förr eller senare vill vi dock skriva ut dokumentet på papper eller dataskärmen och behöver en konkret visuell avbildning av bokstäverna i dokumentet i alla typsnittsvarianter som används. Här kommer METAFONT in i bilden.

En METAFONT-font är ett program som i ett kartesiskt koordinatsystem med hjälp av geometriska ekvationer och pennor med olika ritegenskaper beskriver egenskaperna av tecknen i en teckenuppsättning. Genom att variera parametrar kan man med

ett enda program beskriva tecknen i flera storlekar, varierande lutning och tjocklek med mera [Knu86]. När TEX sätter ett dokument behöver den veta dimensionerna på symbolerna, vilka den får färdigt uträknade av METAFONT. TEX placerar det satta dokumentet i en såkallad DVI- eller *device independent*-fil. DVI-filen innehåller informationen om vilka typsnitt skall användas var, men fonterna själva inkluderas inte i filen.

För att tecken skall kunna visas på skärm eller skrivas ut på papper måste fonten som används *rasteriseras*. Vid rasteriseringen anpassas tecknens former till skärmens eller skrivarens pixel- eller punktavstånd och beroende av resolutionen och rasteriseringsalgoritmen är resultatet en mera eller mindre god representation av fonten i formen av bitkartor. Tanken med DVI-filer är att utskriftsrutinen för skärmen eller skrivaren vid behov ber METAFONT att räkna ut bitkartorna av fonterna som behövs, i rätt resolution för utrustningen i fråga.

Numera är styrningsspråket för så gott som alla högklassiga printrar PostScript och det faktiska standardformatet för distribuering av dokument i elektronisk format PDF, som baserar sig på PostScript. PostScript är ett lågnivåspråk och möjliggör, liksom METAFONT, beskrivning av fonter i vektorformat. Dessvärre har det visat sig mycket svårt att konvertera METAFONT-program till Type 1, som är det prefererade sättet att koda fonterna i PostScript [Sza01, JNS01]. Type 1-fonter är kompakta PostScriptprogram som följer ett disciplinerat programmeringsmönster, vilken skall göra dem lätta och effektiva att rasterisera. De är typiskt designade med något grafiskt program. METAFONT-fonter kodas vanligen för hand, och programmeringsspråket ger designaren sådana kraftiga verktyg som möjligheten att lösa ekvationssystem, vilka Type 1 saknar.

Det enklaste och vanligaste sättet att använda METAFONT-fonter med PostScript är att inkludera fonterna i PostScriptfilen som bitkartor i Type 3-format. Liksom Type 1-fonterna är Type 3-fonter också PostScript men de behöver inte följa de stränga reglerna som Type 1-fonterna uppfyller. Det fruktbaraste försöket att konvertera METAFONT-program till Type 1 har hittills varit att gå via bitkartor uträknade av METAFONT i en speciellt hög resolution och återvektorisera dem med ett lämpligt program [Sza01]. Det finns också redskap som konverterar METAFONT-program till PostScript i vektorformat direkt, men dessa sätter förtretliga begränsningar på de tillåtna METAFONT-instruktionerna, vilket gör det omöjligt att på det här sättet automatiskt konvertera TEXs originalfonter till Type 1 [JNS01].

Ett alternativ till att använda METAFONT med TEX är att använda PostScript-fonter. Det enda TEX behöver är metrikfilerna, som är lätta att räkna. Konversionen från Type 1 till METAFONT är också enkel, men i allmänhet onödig [Sza01]. Ett problem med PostScriptfonter är att de ofta inte innehåller alla matematiska symboler som behövs.

3 Brytning

3.1 En översikt

Det vanliga sättet att bryta textstycken i rader är att helt enkelt sätta ord efter ord och låta det första ordet eller den första stavelsen som inte ryms börja en ny rad. Figur 3 på sidan 8 visar att den här enkla metoden inte alltid producerar bästa möjliga layout, utan kan i text med jämn högerkant ge onödigt stora gluggar mellan orden. Gluggar gör läsandet hackigt genom att de tvingar läsaren att flytta sin blick onödigt långt eller flera gånger. De gör det också svårare att hålla blicken på rätt rad [Hel94]. Dessutom bidrar stora ordmellanrum till att olika mönster uppstår av tomrummen på raderna och fångar läsarens uppmärksamhet [KP81].

Flera sätt finns att handskas med gluggarna. Ett dåligt sätt är spärrning, det vill säga ökning av avståndet mellan tecknen i orden, eftersom det söndrar ordbilden [Hel94]. I exempeltexten löser en brytning av ordet glassgubbar problemet, men texten kan också brytas nöjaktigt utan spärrning eller avstavning genom att sätta början av stycket glesare, såsom på den högra spalten i figur 3. I följande kapitel lär vi oss hur TEX bryter stycken genom att räkna ut den bästa kombinationen av radbrytningar och hur den vid behov finner de rätta ställena för avstavning när avstavning inte kan undvikas.

En uppgift besläktad med radbrytning är sidbrytning. En sidbrytning får helst inte inträffa vid ett avstavat ord eller före styckets utgångsrad. En ensam utgånsrad som hamnar överst på en sida kallas för en dubbel horunge och är desto fulare ju kortare den är. En enkel horunge är en ensam förstarad som hamnar längst ner på en sida. Enkla horungar är inte lika fula som dubbla eftersom de fyller hela raden, och de har därför sedan länge allmänt accepterats i svensk typografi [Hel94].

Att välja goda ställen för sidbrytningar kan vara en hel del svårare än att bryta rader väl [Knu84]. Om radmängden per sida tillåts variera, endera genom att variera på styckeavståndet eller sidlängden, kan en dålig sidbrytning ofta korrigeras genom att flytta en enstaka rad till föregående eller nästa sida. Annars kan man be radbrytningsalgoritmen att försöka bryta ett eller två stycken glesare eller tätare så att radmängden blir större eller mindre.

TEX fyller ut sidorna oberoende av radbrytningen, en sida i taget, så användaren kan bli tvungen att göra korrigeringar genom att sätta in tillägsinstruktioner för brytningen i källtexten. Ännu tjugo år efter skapandet av TEX är program för automatisk optimering av sidlayout fortfarande på prototypnivå [BKKW03]. Det finns också försök att optimera sidbrytningarna med hjälp av makropaket byggda ovanpå TEX [Fin00].

3.2 Optimal radbrytning

TEXs radbrytningsalgoritm beskrivs i detalj i Donald Knuths och Michael Plass' Artikel *Breaking Paragraphs into Lines* [KP81]. Här angriper vi algoritmen genom att studera hur den bryter ett exempelstycke lånat från början av Gösta Knutssons Pelle Svanslös i Amerika [Knu41]. Vi kommer att se att den optimala brytningen visad i den högra spalten av figur 3 utgörs av den kortaste vägen genom nätet i figur 4.

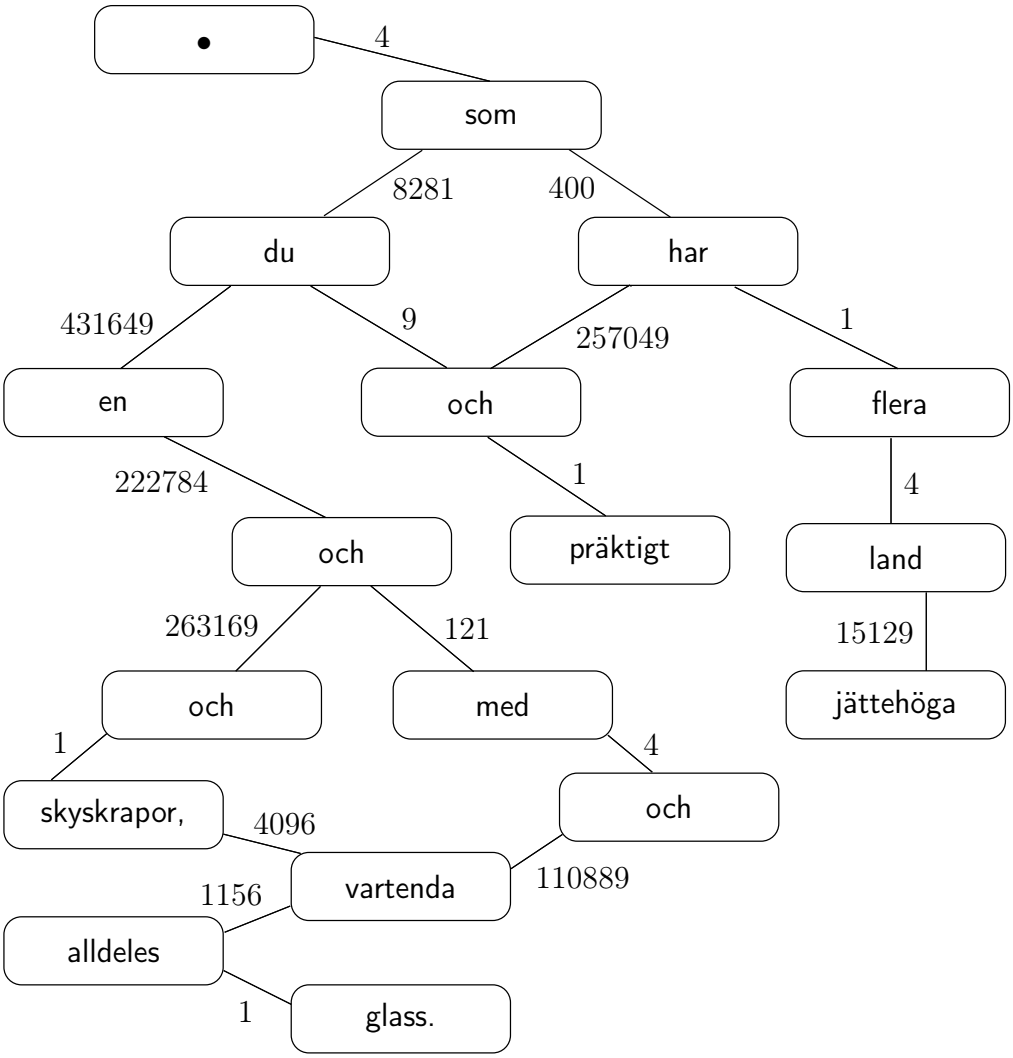
Innan radbrytningsalgoritmen kallas antas råtexten för stycket som skall brytas ha konverterats till en följd av boxar, lim, *straffar* och *diskretionära brytpunkter* [Knu84]. Boxar och lim har vi definierat i kapitel 2.1. Ett straff (penalty) är ett heltal som anger hur önskvärd eller oönskvärd en radbrytning vid en viss punkt är. Värden $\leq -10\ 000$ motsvarar en tvingad radbrytning; 0 anger att en brytning tillåts och att den varken är bra eller dålig; straffar $\geq 10\ 000$ förbjuder brytning på stället ifråga. Diskretionära brytpunkter (discretionary breaks) anger möjliga avstavningsställen och har förmodligen blivit insatta av avstavningsalgoritmen. Den vanligaste diskretionära brytpunkten motsvarar en mjuk divis och ett på förhand bestämt straff på till exempel 50.

I det enkla fallet som vi här skall betrakta består stycket som skall sättas bara av boxar och lim. Det här betyder bland annat att vi inte avstavar ord. För radbrytningsalgoritmen är lim som direkt påföljer en box en tillåten brytpunkt. Om raden

Långt borta i världen ligger ett land, som heter Amerika, och jag tror nog, att du har hört talas om det landet både en och flera gånger. Det är ett stort och präktigt land med massor av bilar och med jättehöga hus, som kallas skyskrapor, och glassgubbar finns det i nästan vartenda gathörn, för folket i Amerika tycker alldeles förskräckligt mycket om glass.

Långt borta i världen ligger ett land, som heter Amerika, och jag tror nog, att du har hört talas om det landet både en och flera gånger. Det är ett stort och präktigt land med massor av bilar och med jättehöga hus, som kallas skyskrapor, och glassgubbar finns det i nästan vartenda gathörn, för folket i Amerika tycker alldeles förskräckligt mycket om glass.

Figur 3: Det första stycket ur Pelle Svanslös i Amerika [Knu41] brutet på två sätt. Den vänstra spalten är bruten rad för rad, en rad i taget; den högra har TEX brutit på ett optimalt sätt med avseende på helheten.



Figur 4: En visualisering av ett nät som bildas när TEX väljer radbrytningarna för texten i figur 3

bryts, försvinner limmet vid brytpunkten. Om limmet mellan de ordbildande bokstavsboxarna på raden är töjbart, vilket vi antar att det är, töjs det så mycket att raden blir full. Det här ger raden en *dålighet* (badness). Dåligheten är ett jämförelsetal som är beroende av det använda typsnittets egenskaper och räknas utgående från hur mycket limmet på raden har töjt eller krympt i förhållande till radens totala töjbarhet eller krympbarhet.

Med den spaltbredd och det typsnitt som används i figur 3 kan den första raden av exempeltexten brytas bara på ett acceptabelt sätt, det vill säga efter ordet som. Att brytpunkten är acceptabel betyder här att den brutna radens dålighet är mindre än 1000. Raden har dåligheten $b = 1$ och brytpunkten får en *dålighetsdiskreditering* (demerits) $d = 4$ enligt formeln

$$d = \begin{cases} (1+b)^2 + p^2, & \text{om } 0 \leq p < 10\ 000; \\ (1+b)^2 - p^2, & \text{om } -10\ 000 < p < 0; \\ (1+b)^2, & \text{om } p \leq -10\ 000, \end{cases}$$

där p är storleken på ett eventuellt straff och är $= 0$ när raden bryts vid lim. Formeln lämnar diskrediteringen för punkter med $p \geq 10\ 000$ odefinierad, eftersom en radbrytning vid en sådan punkt inte är möjlig. För punkter med $p \leq -10\ 000$ är en radbrytning tvingad och strafftermen för raden ifråga kan förkortas bort eftersom straffet ingår som brytpunkt i alla tillåtna sätt att bryta stycket och kan således inte ge någon av sättena någon fördel över de andra [Knu84].

Den andra raden kan brytas efter ordet du med dåligheten $b = 90$ eller har med dåligheten $b = 19$. Om vi omräknar dåligheterna till diskrediteringar enligt formeln ovan och adderar dem till de tidigare, har vi ett val mellan 8285 och 404 diskrediteringar. Vi är dock inte ännu färdiga att välja mellan vägarna, utan fortsätter utforskningen av stycket längs båda av dem.

Om den andra raden bryts efter du, kan den tredje brytas efter en eller och, så att dåligheten är $b = 656$ eller $b = 9$. Om den andra raden bryts efter har, har vi möjligheterna och ($b = 506$) och flera ($b = 0$). För brytningen av resten av stycket är det detsamma hur vi har brutit början av stycket ifall den tredje raden avslutas efter och, så vi glömmer kombinationen som–har–och, som skulle ge sammanlagt 257 453 diskrediteringar och är sämre än kombinationen som–du–och. Kvar blir kombinationerna eller vägarna som–du–en, som–du–och och som–har–flera, vilka motsvaras av de kumulativa diskrediteringarna 439 934, 8294 och 405.

För den fjärde raden får vi kombinationerna som–du–en–och, som–du–och–präktigt och som–har–flera–land samt de kumulativa diskrediteringarna 662 718, 8295 och 409, och för den femte kombinationerna som–du–en–och–och, som–du–en–och–med och som–har–flera–land–jättehöga samt de kumulativa diskrediteringarna 925 887, 662 839 och 15 538. Det finns inget acceptabelt sätt att bryta den femte raden så att den fjärde raden skulle brytas efter präktigt!

Vi har nu kommit över hälften av stycket och har byggt den övre delen av radbrytningsnätet visualiserat i figur 4. Noderna i nätet motsvarar de möjliga brytpunkterna, och avstånden mellan dem utgörs av dålighetsdiskrediteringarna. Nätet är inte ännu färdigt, men redan nu vet vi att noderna och och präktigt i mitten av det är onödiga så vi stryker dem. Om stycket slutade efter ordet skyskrapor, skulle Knuths algoritm välja att bryta det som på den vänstra spalten i figur 3, men som det fortsätter är denna brytning inte acceptabel: ordet glassgubbar går inte att klämma in på den sjätte raden och som raden är bruten i den vänstra spalten har den den största möjliga dåligheten $b = 10\ 000$. Vägen –har–flera–land–jättehöga kan alltså också strykas.

Det visar sig att exempelstycket kan brytas på två acceptabla sätt. Båda bryter de fyra första och de två sista raderna lika. Nätet i figur 4 finns aldrig i sin helhet i TEXs minne: när TEX har läst och behandlat ordet vartenda består nätet faktiskt enbart av vägen –som–du–en–och–med–och–vartenda. Uträknandet av den kortaste vägen sker alltså samtidigt som byggandet av nätet.

TEXs radbrytningsalgoritm och box/lim/straff-modellen har många andra tillämpningar förutom brytning av enkla textstycken till rak högerkant [KP81]. Ojämn högerkant kunde till exempel åstadkommas genom att låta ordmellanrummen i texten stå för följden $l_1 s l_2$, där l_1 är en limenhet med vidden 0, töjbarheten $t > 0$ och krymbarheten 0, s är ett straff med $p = 0$, och l_2 är en limenhet med vidden av ett normalt ordavstånd, töjbarheten $-t$ och krymbarheten 0. Det här fungerar dels eftersom den negativa töjbarheten av l_2 upphäver töjbarheten av l_1, dels eftersom radbrytningar, som här i praktiken bara inträffar vid straffar, äter upp det påföljande limmet l_2. Exempel på andra användningar för algoritmen är sättning av programkod och komplicerade index. Figur 5 visar den optimala brytningen av exempeltexten med ojämn högerkant. Märk att brytställena påverkas av att ordmellanrummen inte är flexibla.

Långt borta i världen ligger ett land, som
heter Amerika, och jag tror nog, att du
har hört talas om det landet både en och
flera gånger. Det är ett stort och präktigt
land med massor av bilar och med
jättehöga hus, som kallas skyskrapor, och
glassgubbar finns det i nästan vartenda
gathörn, för folket i Amerika tycker
alldeles förskräckligt mycket om glass.

Figur 5: Texten ur figur 3 optimalt bruten till en ojämn högerkant

3.3 Avstavning av ord

Radbrytningsexemplet i figur 3 är förstås konstgjort; det långa ordet glassgubbar skulle inte ställa till med några problem om avstavning av ord hade tillåtits. Till exempel i matematisk text kan emellertid brytning av en lång formel på två rader vara oönskvärt, och i sådana fall är den optimerande radbrytningsalgoritmen behändig.

Avstavning är också svårare att automatisera än radbrytning. Klart är att text som innehåller långa ord eller är avsedd att sättas på en smal spalt inte kan brytas automatiskt utan en god avstavningsalgoritm, så problemet har undersökts en hel del. En god algoritm, skräddarsydd för TEX, beskrivs i Franklin Mark Liangs doktorsavhandling [Lia83]. Liang redogör också för några tidigare använda algoritmer och jämför den nya algoritmen med dessa.

TEX försöker med sina grundinställningar undvika avstavning och kan bryta många stycken väl helt utan att kalla på avstavningsalgoritmen. Om en bra brytning utan avstavning inte är möjlig, försöker TEX hitta mera möjliga brytningsställen genom att avstava alla orden i stycket. Detta ställer höga effektivitetskrav för avstavningsalgoritmen. Liangs algoritm är snabb, använder lite minne och hittar med få fel nästan alla tillåtna avstavningsställen. Den kan också anpassas för de flesta språk [Knu84].

En avstavningsalgoritm kan basera sig på regler eller ordlistor. Om algoritmen baserar sig på regler är problemet att formulera reglerna i en exakt form så att de är meningsfulla utan en allmän kännedom av språket ifråga. Om man kräver att algoritmen hittar majoriteten av de tillåtna avstavningsställena utan att samtidigt föreslå en massa otillåtna punkter är uppgiften beroende på språket endera svår eller omöjlig [Lia83]. En ordlistbaserad algoritm hittar avstavningsställena desto bättre ju större ordlista den har. Eftersom ordlistan måste innehålla både sammansatta och böjda former av ord blir storleken lätt ett problem; för att algoritmen skall vara snabb måste hela ordlistan rymmas i centralminne.

Långt borta i världen ligger ett land, som heter Amerika, och jag tror nog, att du har hört talas om det landet både en och flera gånger. Det är ett stort och präktigt land med massor av bilar och med jättehöga hus, som kallas skyskrapor, och glassgubbar finns det i nästan vartenda gathörn, för folket i Amerika tycker alldeles förskräckligt mycket om glass.

Figur 6: Texten ur figur 3 optimalt bruten till en rak högerkant med avstavning tillåten

TEXs algoritm är inte rent regel- eller ordlistbaserad. Algoritmen använder avstavningsmönster, som tillåter eller förbjuder avstavning mellan olika bokstavskombinationer. Råden som mönstren ger för avstavning har prioriteter, som anges med siffrorna 1–5. Det allmänna rådet ”bryt före och inte efter bokstaven b” ($_1$b$_2$) kan överskridas av de mera specifika ”bryt inte före bb” ($_4$bb) och ”bryt före ba” ($_3$ba). Siffrorna 1, 3 och 5 tillåter avstavning medan 2 och 4 förbjuder. Så här kan ett tillåtande och förbjudande avstavningsråd inte ha samma prioritet och inga konflikter uppstår; högre siffra väljs alltid framom lägre. På exempelordet glassgubbar passar mönstren

$_2$las, $_4$s$_3$g$_2$, $_1$g$_2$u, u$_2$b, $_1$b$_2$, $_4$bb och $_3$ba.

När dessa kombineras med ordet får vi g$_2$las$_4$s$_3$g$_2$u$_4$b$_3$b$_2$ar, det vill säga avstavningen glass-gub-bar. Figur 6 visar en optimal brytning av exempeltexten ur figur 3 när avstavning tillåts.

Om konstruerandet av avstavningsmönstren konstaterar Knuth: ”meningen är att mönstren utarbetas av experter som betalas väl för sin expertis” [Knu84]. Ifall en avstavningsordlista finns tillgänglig för språket ifråga kommer man långt med Liangs program `PATGEN` [Lia83], som bildar mönstren automatiskt med hjälp av ordlistan. Listan av avstavningsmönster kan ses som en komprimerad form av ordlistan, där all sådan information som för avstavningen är oväsentlig har lämnats bort.

Även om en avstavningsordlista finns tillgänglig kan problem uppstå från att den inte till exempel innehåller böjda former av ord, så språkteknologernas hjälp behövs för att fullända listan av mönster. TEX avstavar svensk text med hjälp av circa 4700 avstavningsmönster gjorda med `PATGEN` från en ordlista på drygt 100 000 ord och gör ibland fel med sammansatta ord. Finsk text avstavar TEX å andra sidan så gott som felfritt med knappa 300 mönster utarbetade för hand av ett antal språkexperter.

För att avstavningsmönstren som passar på ord skall hittas snabbt bygger TEXs initialiseringsprogram upp en speciell datastruktur av dem, kallad en *packad trie* [Lia83]. Innehållet av triet kan ändras endast genom att köra initaliseringsprogrammet på nytt. Om användaren märker att TEX avstavar något ord fel kan han dock tillsätta det med den rätta avstavningen markerad i en dynamisk undantagsordlista, som TEX uppehåller under kompileringen av dokumentet.

4 Slutsatser

TEX är en viktig del av typsättningens och datavetenskapets historia. Den är ett exeptionellt program. Den är väldefinierad och väldokumenterad och innehåller inga kända fel. Vi kan säga att den är historia eftersom Knuth har bestämt att den inte längre skall vidareutvecklas [Knu90]. TEX har blivit färdig; nya egenskaper såsom stöd för Unicode implementeras i program härledda från den.

Som vi har konstaterat, automatiserar TEX ensam inte typsättning helt och hållet, utan till exempel för korrigering av dåliga sidbrytningar behöver den hjälp. Om raderna i ett textstycke inte kan brytas acceptabelt inom den givna toleransen, försöker heller TEX inte hitta den bästa kompromisslösningen utan markerar problemraden i utskriftsfilen så att användaren skall märka den och berätta TEX hur raden skall korrigeras. Den här problemlösningsmodellen passar bra för tryckning av böcker, då texten i varje fall måste korrekturläsas. Mänskliga typsättare blir också glada om de också får utnyttja sitt kunnande ibland.

Internet håller på att revolutionera våra sätt att leta efter och tillägna oss information. Det är möjligt att samtidigt som traditionell typsättning har automatiserats, har betydelsen och uppskattningen av god typografi minskat. Enligt sina grundstilinställningar sätter till exempel dagens webbläsare textraderna till fönstrets fulla bredd, vilket kan vara närmare 200 tecken per rad. De bekymrar sig inte för att så här långa rader betydligt försämrar läsbarheten, även om de minskar mängden dåliga radbrytningar.

När det gäller webbsidor finns det ingen möjlighet att korrekturläsa den slutliga layouten av sidan, eftersom den är beroende av fönsterstorleken, programmet och stilinställningarna som läsaren använder. Sättningen måste alltså göras helt automatiskt. Kunde radbrytnings- och avstavningsalgoritmerna vi har studerat ändå användas till att förbättra sättningen av webbsidor?

TEXs radbrytningsalgoritm är behändig vid typsättning av facktext som innehåller å ena sidan ord som är korta eller kan avstavas och ger flexibilitet och å andra sidan obrytbara formler eller dylika för vilkas placering flexibiliteten kan utnyttjas. En annan fördel med algoritmen är att den också för enklare texter minskar mängden ord som måste avstavas och gör således texten antagligen en aning smidigare att läsa. För visuella textbehandlingsprogram lämpar sig algoritmen dåligt; skrivaren tvingas välja om det är viktigare att sättningen är den bästa möjliga eller att den syns i realtid medan man skriver. För webbläsare passar algoritmen däremot ypperligt. Webbsidor är ofta facktext med långa ord och att kunna kringgå behovet av avstavning är värdefullt. Algoritmen är snabb och kan implementeras så att den har en vettig nödlösning också för stycken som saknar en acceptabel brytning [KP81].

Om man sätter webbsidorna med ojämn högerkant och väljer radbrytningspunkterna väl är avstavning av ord sällan önskvärt. Med tanke på arbetsmängden är det således obefogat att lära webbläsare avstavning. Till skillnad från radbrytning kan avstavning på webbsidor göras på förhand genom att sätta in diskretionära brytpunkter i texten och då kan skrivaren också korrigera eventuella fel som uppkommer om avstavningspunkterna hittas med hjälp av ett avstavningsprogram. Tyvärr finns det ännu inte något fungerande sätt att markera diskretionära brytpunkter i HTML.

Referenser

BKKW03 Anne Brüggemann-Klein, Rolf Klein och Stefan Wohlfeil. On the pagination of complex documents. *Lecture Notes in Computer Science*, 2598:49–68, 2003.

Fin00 Jonathan Fine. Line breaking and page breaking. *TUGboat*, 21(3):210–221, 2000.

Hel94 Christer Hellmark. *Typografisk handbok*. Ordfront & Ytterlids, 1994.

JNS01 Bogusław Jackowski, Janusz M. Nowacki och Piotr Strzelczyk. METATYPE1: a METAPOST-based engine for generating Type 1 fonts. *Proceedings of the EuroTEX 2001 conference*, Kerkrade, Nederländerna, 23–27 september 2001.

Knu41 Gösta Knutsson. *Pelle Svanslös i Amerika*. Bonniers, 1941.

Knu84 Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1984.

Knu86 Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, 1986.

Knu90 Donald E. Knuth. The future of TEX and METAFONT. *TUGboat*, 11(4):489, 1990.

KP81 Donald E. Knuth och Michael F. Plass. Breaking paragraphs into lines. *Software—Practice and Experience*, 11(11):1119–1184, 1981.

Lam04 Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, andra utgåva, 2004.

Lia83 Franklin Mark Liang. *Word Hy-phen-a-tion by Comp-uter*. Doktorsavhandling, Stanford universitet, 1983.

Sza01 Péter Szabó. Conversion of TEX fonts into Type 1 format. *Proceedings of the EuroTEX 2001 conference*, sidorna 192–206, Kerkrade, Nederländerna, 23–27 september 2001.