Назив проблема: CMYK

Аутор: Петар Величковић
Анализа: Никола Јовановић
Тагови: динамичко програмирање

[Draft 1 - Neke stvari ce moci biti popunjene posle kraja kvalifikacija i kad se dogovorimo oko biltena. Treba srediti pseudokod.]

Извор и идеја:

Проблем **CMYK** је постављен као најтежи проблем на овогодишњим квалификацијама. Статистика [link ka statistici, gde god bila (u ovom resenju, u biltenu negde)] показује [valjda] да је овај проблем заиста највише намучио такмичаре (само X такмичара има максималан број поена).

Теоријску основу овог задатка представља проблем **парсирања контекстно-слободних граматика** (eng. context-free grammars). Конкретно, граматика дата у задатку је представљена у **Чомски нормалној форми** (eng. Chomsky normal form) па је парсирање могуће (уз мале модификације) извршити алгоритмом **Cocke–Younger–Kasami (CYK)** , чије име се крије у наслову задатка. Иако чине теоријску основу овог задатка, увођење поменутих појмова из области формалних језика није неопходно, будући да је за разумевање CYK алгоритма са потребним модификацијама и комплетно решавање овог задатка потребно само познавање динамичког програмирања. Стога, тај део ћемо заобићи, и како би решење било јасније и разумљиво такмичарима који нису упућени у поменуту теорију у наставку анализе ћемо одступати од формалне терминологије.

Оптимално решење:

Као што је већ поменуто, оптимално решење за овај задатак (и једино које је примећено међу решењима са максималним бројем поена [valjda]) се базира на **CYK** алгоритму. Иако овај алгоритам "има име", очекује се да би такмичари који имају искуства са динамичким програмирањем требало да буду у стању да независно дођу до решења, без обзира на то што се нису раније сусрели са конкретним алгоритмом. CYK алгоритам у својој основној форми као улаз узима низ правила налик онима у задатку (тј. контексно-слободну граматику у Чомски нормалној форми), стринг `S`, и стартни симбол (у нашем случају то је симбол `'W'`) и проверава да ли је могуће коначном применом правила доћи од стринга `S` до стартног симбола (тј. да ли `S` припада језику који граматика дефинише). Сложеност овог алгоритма је $\mathcal{O}(n^3 * m)$, где је `n` дужина стринга `S` а `m` број правила. У нашем случају, алгоритам може да одреди да ли се унети низ боја може свести на **једну** белу боју, што није довољно, будући да треба проверити да ли се улазни низ боја може свести на низ белих боја произвољне дужине. У наставку ћемо овај проблем, који онемогућава директно коришћење CYK алгоритма, звати **проблем низа** и поменућемо два приступа која га успешно решавају. Пре тога, објаснимо основну верзију CYK алгоритма примењену на овај задатак.

Најважније својство које CYK користи је да су сва правила облика `XY` $\rightarrow$ `Z`. Знајући ово, главна идеја алгоритма је да истражи све могуће начине да се стринг `S` подели на два дела и покуша да примени једно од правила на симболе добијене трансформацијом тих делова. Међурезултати се чувају у тродимензионалном низу *boolean* вредности `DP`, где `DP[i][j][k]` има вредност *true* акко је могуће подстринг дужине `i` који почиње на индексу `j` коначном применом правила свести на симбол `k`. У првом кораку се поставе вредности за све поднизове дужине 1 (тј. `DP[1][j][k]` има вредност *true* акко је `s[j] = k`). Након тога, алгоритам обрађује подстрингове редом, растуће по њиховим дужинама, и покушава да сваки подстринг подели на два дисјунктна подстринга `L` и `R`, након чега проверава да ли постоји правило типа `AB` $\rightarrow$ `C` такво да је се `L` може трансформисати у симбол `A` и да се `R` може трансформисати у симбол `B` (ове вредности се могу наћи у табели). Ако такво правило

постоји, у табелу се уписује да је тренутни подстринг могуће трансформисати у симбол `C`. Када алгоритам обради све подстрингове, укључујући и комплетан стринг `S`, информацију о томе да ли се се **S** може трансформисати у симбол `W` читамо из `DP[n][1][W]` (ако је стринг индексиран од позиције 1).

Сада долазимо до поменутог "проблема низа". Како модификовати/надоградити овај алгоритам тако да тестира трансфомацију у низ симбола `W` уместо у појединачни симбол? Творци задатка су дошли до два решења за овај проблем:

- **CYK_start**: Додавањем новог симбола `^` на почетак стринга и правила `^W` → `W` у скуп правила омогућавамо да се низови белих боја на почетку стринга (након `^`), условно речено, "сажму" тј. да се низ симбола `W` (стринг облика `^W..W`) може трансформисати у један симбол `^`. Сада се наш услов мења, и ако CYK докаже да се модификовани стартни стринг може довести до симбола `^`, знамо да је одговор "**YES**".
- **CYK_tiles**: Приступ који је већина [verovatno] такмичара користила. Овога пута не мењамо скуп правила и стринг већ из CYK табеле тривијално извлачимо све подстрингове који се могу трансформисати у `W` и решавамо потпроблем: да ли је могуће тим подстринговима "поплочати" цео стринг `S` (тј. да ли је могуће одабрати скуп дисјунктних подстрингова чија је унија цео стринг `S` такав да се сваки од њих може трансформисати у `W`)? Јасно је да је решење потпроблема еквивалентно решењу задатка, а потпроблем се решава још једном применом динамичког програмирања, коришћењем рекурентне везе $T[i] = T[i-l_1]|\ldots|T[i-l_n]$, где је l_i дужина `i`-тог подстринга који се може трансформисати у `W`. Дакле, `T[i]` се односи на поплочавање првих `i` карактера стринга, па се решење налази у `T[n]`.

Псеудокод CYK алгоритма са "start" модификацијом [kako se pise pseudokod na srpskom, pomoc]:

Нека је улазни стринг `S` дужине `n`: $a_1 ... a_n$.
Нека је скуп дозвољених симбола $[A..Z, \hat{}]$.
Нека имамо `m` правила облика $A \to BC$.
Нека је $DP[n, n, r]$ матрица boolean вредности, где су све вредности на почетку false.

```
for each j = 1 to n:
    set DP[1, j, a_j] = true
for each i = 2 to n: – дужина подстринга
    for each j = 1 to n-i+1: – почетни индекс
        for each k = 1 to i-1: – дужина првог сегмента
            for each A → BC:
                if DP[k, j, B] i DP[i − k, j + k, C] then set DP[i, j, A] = true

if DP[n, 1, ^] is true
    одговор је "YES"
else
    одговор је "NO"
```

Погледајмо сада илустрацију рада CYK алгоритма са "start" модификацијом на примеру. Нека имамо правила `RG` → `B` и `BB` → `W` и нека је стринг `S = ''RGB''`.

Модификован скуп правила:
`RG` → `B`
`BB` → `W`
`^W` → `^`

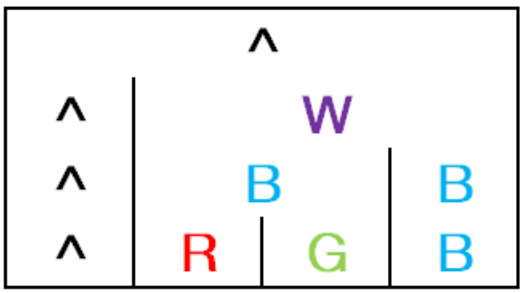
Модификован стринг S:
`^RGB`

Приказ табеле DP на овом примеру (како није згодно цртати тродимензионе табеле, у овој илустрацији се у `DP[i][j]` налазе сви симболи за које је `DP[i][j][k]` = *true*):

i \ j	1	2	3	4
1	^	R	G	B
2	/	B	/	
3	/	W		
4	^			

Симбол `^` се налази у `DP[4][1]` из чега закључујемо да је одговор **"YES"**.

Илустрација (са леве стране се налази стринг као и скуп правила, а са десне су приказане "поделе" које доводе до решења и које одговарају правилима која треба применити да би се до решења дошло:



Субоптимална решења:

Сада када знамо оптимално решење задатка, осврнимо се на нека субоптимална решења:

- **Brute-force приступ**, који у овом случају подразумева да покушамо да применимо све могуће секвенце правила за мешање боја. Стандардна имплементација овог приступа би рекурзивно на тренутни низ боја примењивала сва правила која је могуће применити. Сложеност овог приступа је $\mathcal{O}((nm)^n, valjda, nemamblage)$ и очекиван број поена који би овај приступ донео је $20 - 40$.
- **Greedy приступи**, који подразумевају да у сваком тренутку примењујемо правила на две најлевље тј. најдешње боје. Сложеност овог приступа је $\mathcal{O}(nm)$ и очекиван број поена који би овај приступ донео је $20 - 40$.
- **Недетерминистички приступи и хеуристике**, тј. примена насумичних правила како би се дошло до траженог низа белих боја и коришћење одређених претпоставки и алгоритама који не морају бити увек тачни али се претпоставља да покривају одређени број случајева. Сложеност и успех оваквих приступа варира. Међу такмичарским решењима није примећена успешна примена насумичних правила без претпоставки (ни имплементације твораца овог задатка не успевају да буде боље од *Brute-force* приступа), али је примећено пар делимично успешних хеуристичних приступа. [ovde popuniti] Један од њих се ослања на ЦИ и претпоставља ЦИ и освојио је ЦИ поена.

- **"Компресија"**: Решење које користи CYK алгоритам али као модификацију додаје правило `WW` $\rightarrow$ `W`. Ово решење, иако на први поглед делује као да решава проблем низа и ради за велики број насумичних тест примера, није исправно. Најпростији пример на коме се види неисправност овог приступа је:

  2
  AW $\rightarrow$ X
  XB $\rightarrow$ W
  1
  4
  AWWB

  Приметимо да овде свођење на низ белих боја није могуће иако би "компресија" то успешно урадила (`AWWB` $\rightarrow$ `AWB` $\rightarrow$ `XB` $\rightarrow$ `W`). Ово решење осваја 25 поена.

Тестирање:
Тест корпус овог проблема садржи 20 тест примера, где сваки доноси 5 поена. У свим тест примерима важи да је `t = 10`. Већина примера генерисана је аутоматски на следећи начин:

- Генерише се насумичан скуп правила за дате параметре
- Насумично се врши примена правила (тачније, инверзних правила `C` $\rightarrow$ `AB`) на низ белих боја како би се добили низови за које је решење "YES"
- Насумично се врши додавање и премештање боја на низ добијен у претходном кораку како би се добили низови за које је решење "NO"

Параметри коришћени у генератору тест примера:

- **grammar_size** - Природан број у опсегу [1..100], величина скупа правила (`m`)
- **max_strlen** - Природан број у опсегу [1..100], максимална дужина низа боја (`n`)
- **letters_used** - Природан број у опсегу [4..26], број различитих слова енглеског алфабета који се у том тест примеру појављује
- **force_strlen** - Карактер из скупа $['f', 'n']$, означава да ли ће сви низови дати у тест примеру бити тачно дужине **max_strlen**

Поред примера генерисаних генератором додато је и неколико ручно осмишљених примера, који би требало да покрију примећене граничне случајеве. Такође, како је генератор превише наклоњен неисправном решењу "компресија", било је неопходно понекад ручно додавати примере, како би то решење дало нетачан резултат. У наредној табели је дат преглед тест примера (**тип** = тип тест примера (аутоматски генерисан, мануелно написан), **compress** = на овом тест примеру пролази "компресија"). Пример под бројем 0 је дат у тексту задатка, а при тестирању су коришћени тест примери 1 - 20:

р. бр.	тип	grammar_size	max_strlen	letters_used	force_strlen	compress
0	manual	2	9	4	/	да
1	auto	11	10	5	n	не
2	auto	11	10	26	n	не
3	auto	100	10	26	n	не
4	auto	11	100	26	n	не
5	auto	41	40	26	n	не
6	auto	41	70	26	n	не
7	auto	74	40	26	n	не
8	auto	70	70	5	f	да
9	auto	71	70	26	f	не
10	auto	41	100	26	n	не
11	auto	100	40	26	n	не
12	auto	70	100	26	n	не
13	auto	100	70	26	n	не
14	auto	100	100	5	f	не
15	auto	100	100	26	f	не
16	auto	100	100	26	f	не
17	manual	1	100	4	/	да
18	manual	1	100	6	/	да
19	auto	2	100	3	n	да
20	auto	3	2	5	n	не

Статистика:

[kasnije] Максималан број поена на овом задатку има само X такмичара, док је укупно Y такмичара освојило бар неке поене. Просек и још неке статистике не знам баш које. То ће бити обрађено након краја квалификација тако да ћу тада да упечујем. Неки графици рецимо. Можда на једном месту за све задатке, на почетку билтена.