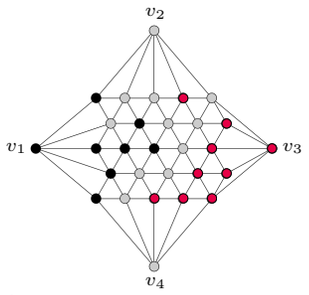
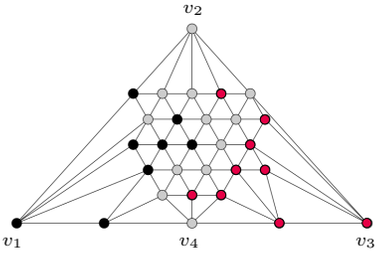


Rys. 9



Rys. 10

Pokolorujmy każdy wierzchołek v według następującej zasady (rys. 9):
- na czarno, jeśli v należy do Aldony i istnieje ścieżka od v_1 do v należąca całkowicie do Aldony,
- na szaro, jeśli v należy do Bogumiła i istnieje ścieżka od v_2 do v należąca całkowicie do Bogumiła,
- na pomarańczowo w przeciwnym wypadku.

Wydawałoby się, że nie da się tu użyć lematu Spernera (w końcu graf na rysunku 9 w żadnym stopniu nie przypomina trójkąta). Nie jest to jednak problem, bo możemy go zdeformować, aby uzyskać trójkąt, co widać na rysunku 10. Sprawdźmy, czy powstała triangulacja spełnia warunki lematu Spernera:
- v_1, v_2 i v_3 są wierzchołkami powstałego dużego trójkąta i są odpowiednio czarne, szare i pomarańczowe (gdyby v_3 było czarne, to istniałaby ścieżka od v_1 do v_3 należąca do Aldony – sprzeczność z założeniem o remisie).
- Na lewym boku dużego trójkąta znajdują się v_1 i v_2, które są odpowiednio czarne i szare. Jest też lewy górny róg planszy. Jest on połączony z v_1 i v_2, zatem jeśli należy do Aldony, to jest czarny, a jeśli należy do Bogumiła, to jest szary. Nie ma więc koloru pomarańczowego.
- Na prawym boku dużego trójkąta znajdują się v_2 i v_3, które są odpowiednio szare i pomarańczowe. Jest też prawy górny róg planszy. Gdyby był czarny, to istniałaby ścieżka od v_1 do v_3 w całości należąca do Aldony – sprzeczność. Nie ma więc koloru czarnego.
- Na dolnym boku dużego trójkąta znajdują się: v_1, v_3, v_4 oraz rogi planszy. Wierzchołek v_1 jest czarny, a v_3 pomarańczowy. Gdyby v_4 lub któryś z rogów planszy był szary, to istniałaby ścieżka od v_2 do dolnej krawędzi planszy należąca tylko do Bogumiła – sprzeczność. Nie ma więc koloru szarego.

Widzimy zatem, że ta triangulacja spełnia warunki lematu i istnieje różnokolorowy mały trójkąt. Oznaczmy przez a, b, c jego wierzchołki, które są odpowiednio czarne, szare i pomarańczowe. Jeśli c należy do Aldony, to ponieważ istnieje ścieżka od v_1 do a należąca do Aldony, to istnieje także ścieżka od v_1 do c należąca do Aldony. Zatem c powinno być czarne – sprzeczność. Podobnie sprzeczność uzyskujemy, gdy c należy do Bogumiła, co kończy dowód. □

Pół szklanki mocnego kodu

Ghost speaker

*Piotr KRZYŻANOWSKI**

* Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

Jest wiele sposobów zrobienia wrażenia na ludziach, ale jednym z najbardziej popularnych jest wciąż

wygłoszenie przemówienia.

W życiu każdego mówcy może przyjść taki moment, gdy wcześniej przygotowaną płomienną przemowę chciałoby się przedłużyć, by jeszcze wyżej wznieść się na emocjach entuzjastycznego tłumu, doskonale rezonującego z głosem z megafonów... Łatwo tu jednak wpaść w pułapkę, bo przecież nie można zbyt długo mówić samych mądrych, samych porywających, samych głęboko przemyślanych zdań. Trzeba więc mieć w zanadrzu pewną „watę słowną", którą w razie konieczności można by *nadmuchać* długość przemówienia.

Jak działa Textinflator? Na przykład wstawia do tekstu niewiele znaczące ozdobniki, zastępuje krótkie wyrazy długimi itp.

Oczywiście problem nie jest nowy – i jest od dawna rozwiązany, zresztą na kilka sposobów. Na przykład na stronie `textinflator.com` możemy skorzystać z automatu, który *rozwadnia* podany tekst – nawet dwukrotnie zwiększając jego objętość! Niestety w trakcie wydarzenia na żywo nie da się z niego skorzystać, bo przecież to, co mieliśmy przeczytać z kartki, już przeczytaliśmy...

W takim przypadku lepiej sprawdzi się rozwiązanie polegające na wykorzystaniu *mowy-trawy*, czyli spreparowaniu tekstu składającego się z okrągłych zdań, zawierających pewne rytualne, oklepane teksty – przez co przemowa będzie udawać, że niesie w sobie jakąś sensowną treść. Między innymi w latach osiemdziesiątych XX wieku tygodnik „Polityka" opublikował żartobliwą tabelkę – *uniwersalną ściągaczkę zebraniową* – wystarczającą na „czterdziestogodzinne przemówienie", tworzone na żywo przez przypadkowe złączenie fraz–wytrychów ówczesnej nowomowy. Przytaczamy ją w całości poniżej, łącznie z instrukcją użycia: *Dowolną frazę z kolumny 1. należy kolejno połączyć z dowolnymi frazami z kolumn 2., 3. i 4.*

Koleżanki i koledzy	realizacja nakreślonych zadań programowych	zmusza nas do przeanalizowania	istniejących warunków administracyjno-finansowych.
Z drugiej strony	zakres i miejsce szkolenia kadr	spełnia istotną rolę w kształtowaniu	dalszych kierunków rozwoju.
Podobnie	stały wzrost ilości i zakres naszej aktywności	wymaga sprecyzowania i określenia	systemu powszechnego uczestnictwa.
Nie zapominajmy jednak, że	aktualna struktura organizacji	pomaga w przygotowaniu i realizacji	postaw uczestników wobec zadań stawianych przez organizację.
W ten oto sposób	nowy model działalności organizacyjnej	zabezpiecza udział szerokiej grupie w kształtowaniu	nowych propozycji.
Praktyka dnia codziennego dowodzi, że	dalszy rozwój różnych form działalności	spełnia ważne zadania w wypracowaniu	kierunków postępowego wychowania.
Wagi i znaczenia tych problemów nie trzeba szerzej uzasadniać, ponieważ	stałe zabezpieczenie informacyjno-programowe naszej działalności	umożliwia w większym stopniu tworzenie	systemu szkolenia kadry odpowiadającego potrzebom.
Różnorakie i bogate doświadczenia	wzmacnianie i rozwijanie struktur	powoduje docenianie wagi	odpowiednich warunków aktywizacji.
Troska organizacji, a szczególnie	konsultacja z szerokim aktywem	przedstawia interesującą próbę sprawdzenia	modelu rozwoju.
Wyższe założenia ideowe, a także	rozpoczęcie powszechnej akcji kształtowania postaw	pociąga za sobą proces wdrażania i unowocześniania	form oddziaływania.

Dziś te frazesy nieco trącą myszką, choć wciąż zaskakująco dobrze się je czyta... Oczywiście *teraz* – z pomocą komputera – możemy bez najmniejszego trudu wygenerować nieskończenie długi tekst oparty na tej czy innej podobnej tabelce (w Internecie jest tego trochę):

```
from random import choice

przemowa = [ ['koleżanki␣i~koledzy,␣', 'z␣drugiej␣strony,␣', ...itd...],
['realizacja␣zadań', 'zakres␣szkolenia␣kadr', ...itd...], ...itp... ]
while True:
  for frazy in przemowa:
    print choice(frazy),
  print
```

Równie łatwo możemy tak wyprodukowane przemówienie automatycznie *przeczytać*, wykorzystując komputerową syntezę mowy – a potem godzinami wsłuchiwać się w coraz bardziej usypiające zdania: w kółko *o tym samym*, a przecież *nie te same*:

```
1  from gtts import gTTS
2  import pygame
3  from random import choice
4
5  pygame.init()
6  pygame.mixer.init()
7
8  while True:
9    for frazy in przemowa:
10     fraza = choice(frazy)
11     print fraza,
12     gTTS(fraza, lang='pl').save('fraza.mp3')
13     pygame.mixer.music.load('fraza.mp3')
14     pygame.mixer.music.play()
15     while pygame.mixer.music.get_busy():
16       continue
17   print
```

Moduł gTTS (*Google Text-to-Speech*) generuje z pomocą internetowego serwisu Google Translate (tak właśnie!) plik mp3 zawierający odczytaną kobiecym głosem frazę.

Zobacz też `cloud.google.com/translate/docs/reference/libraries`.

(Nie syntezujemy całego zdania, tylko pojedyncze fragmenty, ze względu na ograniczenie serwisu do tekstów o maksymalnej długości 100 znaków.) Odtworzenie pliku mp3 powierzamy sympatycznej bibliotece pygame. Co prawda ostateczny efekt dźwiękowy działania naszego skryptu jest daleki od tego, który uzyskalibyśmy, zapraszając do współpracy, powiedzmy, panią Krystynę Czubównę – ale łatwo go ulepszyć, nagrywając własne, pełne pasji interpretacje fraz.

A więc, złotouści, do dzieła! Zróbcie maszynkę opowiadającą młodszemu rodzeństwu *nieskończoną bajkę* na dobranoc! (I w razie sukcesu zapewne także na dzień dobry?) Najzabawniejsze tabelki/kody opublikujemy na stronie internetowej *Delty*, `deltami.edu.pl`.