CybozuLabs

携帯Flashでバイナリ処理

サイボウズ・ラボ株式会社　鴨志田良和

kamoshida@labs.cybozu.co.jp

きっかけ

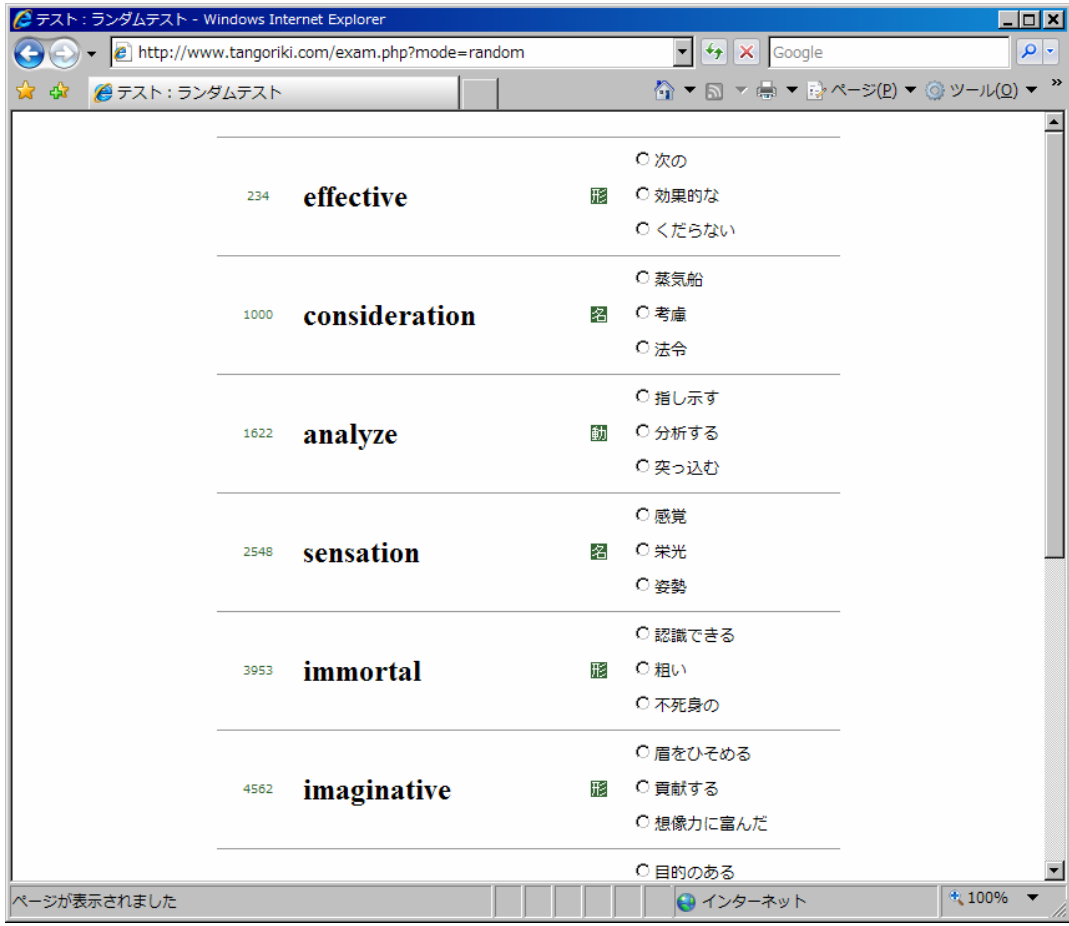
- 最近MNPでケータイ変えた
- Flash Liteがつかえる
- 何か作ってみよう

Flash Lite で携帯アプリ

- 限られたサイズ(100KB)
  - 505i/506iは20KB(今回は考慮せず)
- 1ビットを節約する価値があるプラットフォーム
- 限られたCPU速度
- 今後は改善されていくかもしれない
- 今が旬

英単語テストアプリ

- Tangoriki.com
  - 会員数10万の英単語テストWebサイト
  - 新聞・雑誌等に出てくる頻出単語8000語
    - 日本で広くカタカナ語として使われているものは除外 (economy, summer など)



携帯Flash版 単語力(タンゴリキ)

- Web版のデータはフリーで手に入るので、このデータを使ってケータイ上でtangorikiを再現する
  - 単語データはローカルに持つ
  - 動いてる電車の中で使える←これがウリ
  - 元データは約185KB

しかし悪いニュースが。。。

Flash Lite で携帯アプリ

- 限られたサイズ(100KB)
- 元データは185KB

- あとから通信をして読み込んでくるデータも含めて100KBまで

- orz...

そこで圧縮です

- ためしにgzip --bestしてみた
  - 品詞　　2.5KB
  - 英単語 35.5KB
  - 日本語 30.5KB

- 圧縮すればいけるかもしれない。

でもgzipじゃだめじゃん

- gzip: Lampel-Zif Encodingを使用
  - 辞書が動的に構築される
  - 途中から展開できない
  - 起動時に全部展開するのはおそすぎる
- 必要なところだけ展開したい

Flash Liteでバイナリ操作

- ActionScriptで圧縮データを展開
- Flash Lite1.1のActionScriptはFlash 4ベース
  - (Flash Lite2はFlash7ベース)
  - 現在のFlash最新版は8とか9
  - 先史的な機能が盛りだくさん♪
  - デバッガもない
    - Cソースに#include “hoge.as” で読み込んでGDBでデバッグ

さて突然ですが、

CybozuLabs

簡単ActionScript入門の時間です

Flash Lite 1.x ActionScript入門 1

- 配列アクセス
  - 配列はありません。
  - eval関数を使ってemulateしてください。
    - と、マニュアルに書いてある。
  - 例
  - **eval(“ary” add i) = 123;**
  - “add” は文字列連結命令

Flash Lite 1.x ActionScript入門 2

- 関数呼び出し
  - 関数はありません。
  - 再生しないフレームに書いたアクションスクリプトを呼び出してください。
  - 引数はグローバル変数として渡します。
  - 例

```
arg1 = 123;
call(“dosomething”);
x = result1;
```

Flash Lite 1.x ActionScript入門 3

- ビット演算: ありません(以下略)
- ほかにもいろいろな制約がありそう。

- ここに洗練された圧縮アルゴリズムを持ってきて融合させるのが今回のねらい

CybozuLabs

ここからアルゴリズムの話

データ・処理の概要

- 頻度順に番号付けされた英単語8000語
- 品詞(名詞、動詞、形容詞、副詞)と日本語訳

- 英単語の意味を当てる3択問題を出題
- はずれの2つは同じ品詞の他の単語の中からランダムに選択

実装のおおまかな方針

- 圧縮したままデータを参照
  - 好きな場所から復元できる
- 配列について
  - 変更しない配列は文字列で持ち、substringで参照する

途中から展開できる圧縮方法

- 静的ハフマン木
  - 50年前の技術

ブリッスルコーンパイン:
世界最古の木

- Succinct Data Structure(SDS) 15年前～
  - 整数配列とほぼ同じ領域でsetを実現
  - 最近実践的な実装方法が提案されてきた(検索エンジンなどに応用)
- 古くて新しい技術

温故知新

圧縮データのサイズ

- 試しにハフマン符号化
  - 品詞: 2KB(データ)
  - 英語: 34KB(データ)
  - 日本語: 35KB(データ) 8KB(ハフマン木)
  - 索引: 8K語 x 2B = 16KB
  - 計95KB
  - プログラム入れる隙間がほとんどない
  - orz...

ここで

さらに悪いニュースが。。。

ActionScriptの文字列 1

- ¥0は使えない(swfの中でデリミタとして使用)
- ValidなShift-JISの並び以外はFlash開発環境(Flash8 Pro)が受け付けない
  - 英語版を使ってもだめだった。OSの文字コードに依存?

ActionSctiptの文字列 2

- 例:
  - ¥x82¥xA0 → OK.(あ)
  - ¥x82¥x9D → NG.
    - コンパイル時に、無効なSJISシーケンス以降のすべての文字がカットされる
    - length("abc**¥x82¥x9d**def") == 3
- swfに直接埋め込むことはできる
  - でもプレビューができなくなる＋開発時間の都合で今回は見送り

ActionScriptの文字列でバイナリ操作

- 使える文字:
  - ¥x01-¥x7E
  - ¥xA1-¥xDF(ﾊﾝｶｸｶﾀｶﾅ!!)
  - の189文字
  - 189*189 > 32768
  - 2Byteで15bits表現できる!(実際は182文字でOK)

2バイトで15ビット

- 2桁の182進数でデータを格納
- それ1Byteで7bitでも。。。
  - 1bitを笑うものは1bitに泣く
- ちなみに70KBのデータの場合、1バイトで7ビット使う方法に比べて5.3KBもお得!

友情出演

1円を笑う鳥

がんばって圧縮！！

- ハフマン木はcanonical表現を使ってツリーサイズを圧縮
  - 8KB → 4.3KB
- 単語データ(日本語+英語)
  - ation => “A” など細かい最適化
    - 2.5%程度圧縮率向上
  - 69KB*16/15*100/102=71.5KB

単語索引の圧縮

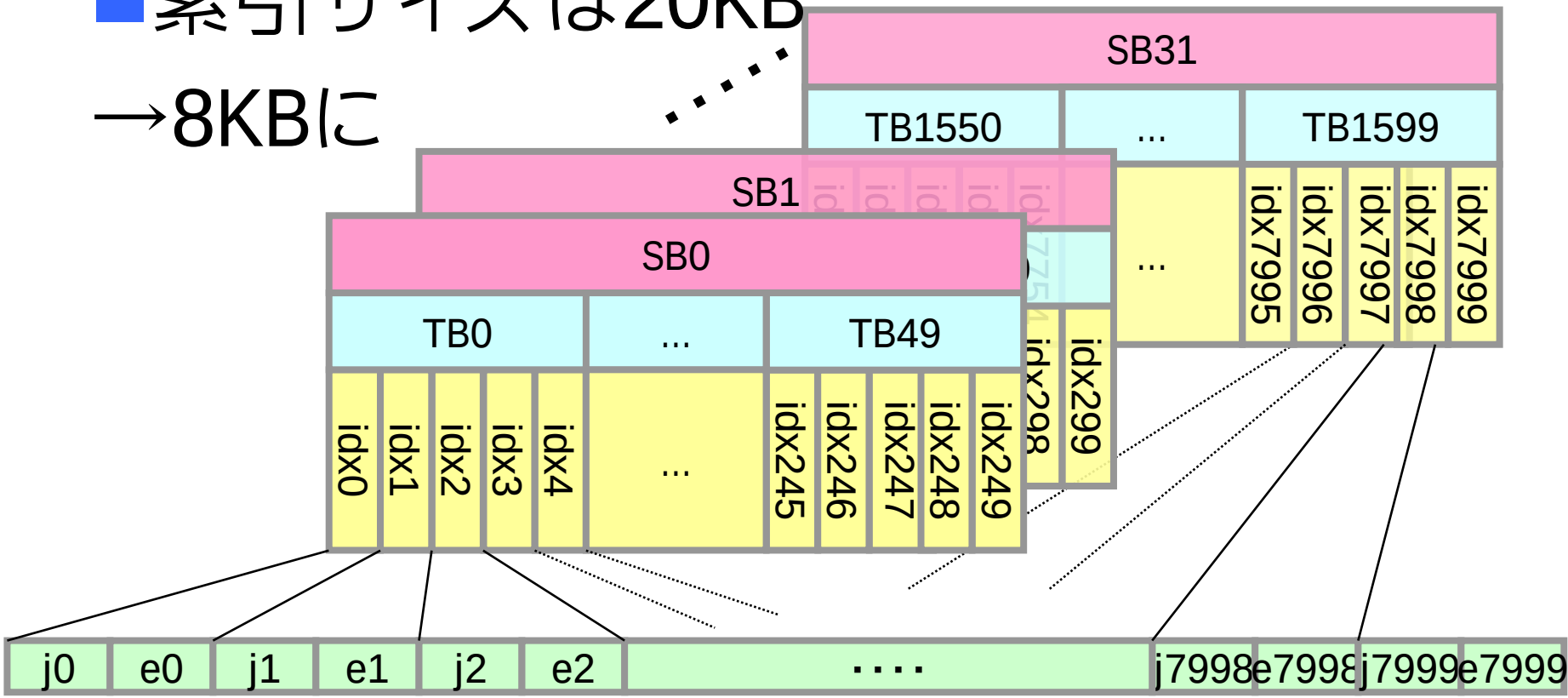
- 符号化後の日本語・英語データ: 71KB
- 単語索引
  - それぞれの単語がどのビットから始まるか記録(各単語へのポインタ)
  - 高速に(できればO(1)で)アクセスできる必要あり
  - log(71KB*8bits) x 8000words = 20KB
  - これも圧縮!

階層的なブロック分割

- 8000語の集合を32個のSuper-block(SB)に分割
- SBをさらに50個のTiny-blockに分割
  - 所属ブロック内の相対アドレスを保持
  - 20bitsを32個に分割→5bits節約
  - TBは15bits、idxは9bitsで表現できる

単語へのポインタの圧縮

- TBを15ビット、idxを9ビットで表現
  - 索引サイズは20KB
    →8KBに

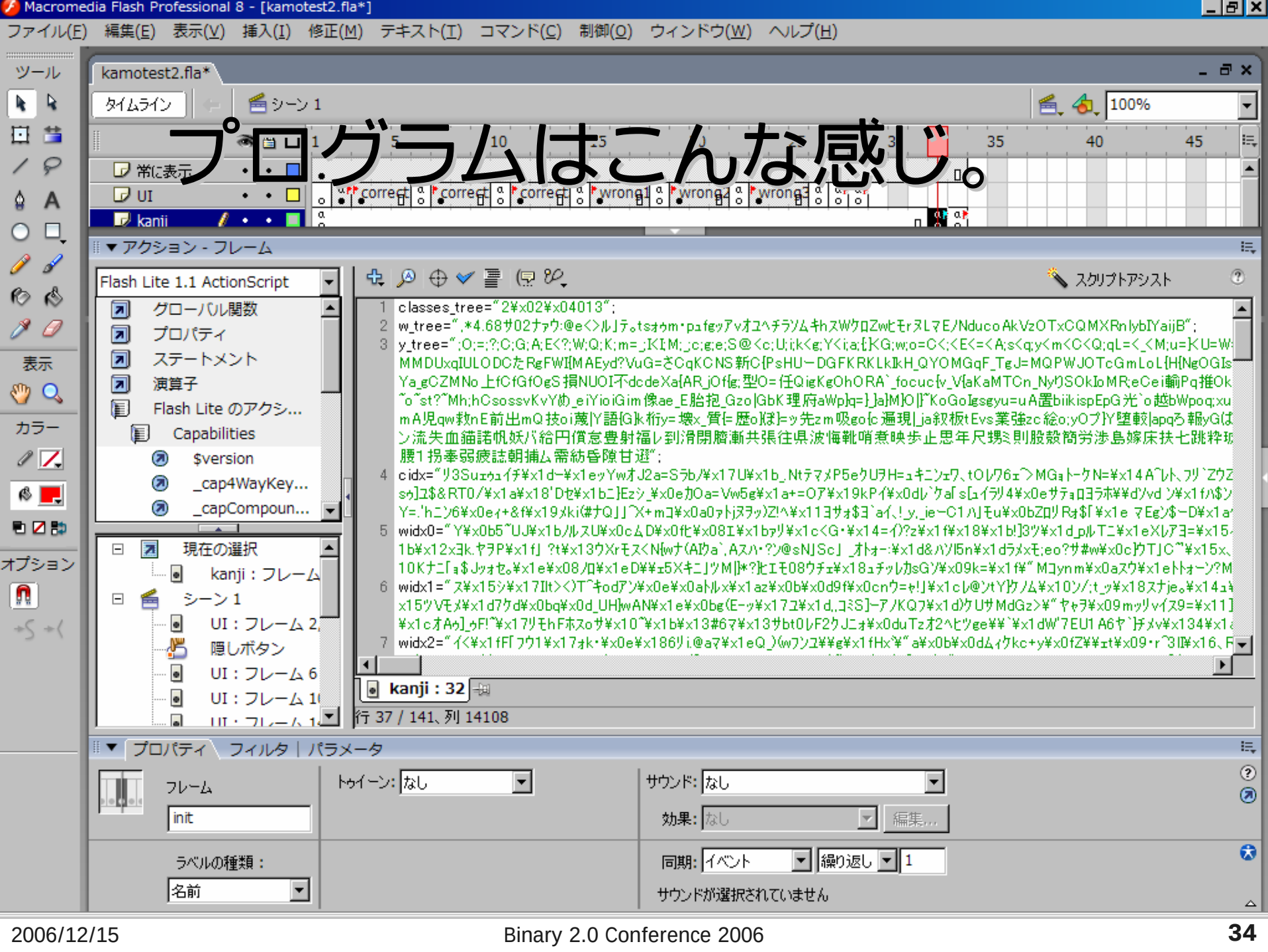


最終的なデータサイズ

- データ(品詞): 2KB→2.5KB 品詞毎のランダム選択を実現するため
- データ(英語・日本語): 69KB→71.5KB
- 符号表(英語・日本語): 4.8KB
- 索引: 8*16/15 = 10.5KB
- 合計 **89.3KB**
- 上記データ用の変数宣言、ビット演算用の表、展開処理など: 6.2KB

最終的なデータサイズ

- ただいま今95KB使用中です。
- 残りの5KBで、単語の三択問題を行うプログラムを作成
  - 2,5,8で選択
- 僕の初めてのActionScript
- 次ページ参照。


Macromedia Flash Professional 8 - [kamotest2.fla*]
ファイル(F) 編集(E) 表示(V) 挿入(I) 修正(M) テキスト(T) コマンド(C) 制御(O) ウィンドウ(W) ヘルプ(H)
ツール
kamotest2.fla*
タイムライン
シーン 1
100%
プログラムはこんな感じ。
常に表示
UI
kanji
correct
correct
correct
wrong1
wrong2
wrong3
アクション - フレーム
Flash Lite 1.1 ActionScript
グローバル関数
プロパティ
ステートメント
演算子
Flash Lite のアクシ...
Capabilities
$version
_cap4WayKey...
_capCompoun...
スクリプトアシスト
現在の選択
kanji : フレーム
シーン 1
UI : フレーム 2
隠しボタン
UI : フレーム 6
UI : フレーム 1
classes_tree="2¥x02¥x04013";
w_tree=",*4.68サ02ナァウ:@e<>ル」テ。tsォゥm・pュfgッアvオユへチラソムキhスWクロZwヒモrヌレマE/NducoAkVzOTxCQMXRnlybIYaijB";
y_tree=";O;=;?;C;G;A;E<?;W;Q;K;m=_;KI;M;_;c;g;e;S@<c;U;i;k<g;Y<i;a;{]KG;w;o=C<;<E<=<A;s<q;y<m<C<Q;qL=<_<M;u=]KU=W:
MMDUxqIULODCたRgFWI[MAEyd?VuG=さCqKCNS新C{PsHU−DGFKRKLkIkH_QYOMGqF_TgJ=MQPWJOTcGmLoL{H[NgOGIs
Ya_gCZMNo上fCfGfOgS損NUOI不dcdeXa[AR_jOf仮型O=任QigKgOhORA`_focuc{v_V[aKaMTCn_NyりSOkIoMR;eCei輸Pq推Ok
~o~st?~Mh;hCsossvKvYめ_eiYioiGim像ae_E胎抱_Gzo|GbK理府aWp]q=}_a]M}O]]~KoGoIgsgyu=uA置biikispEpG光`o越bWpoq;xu
mA児qw救nE前出mQ技oi蔵|Y語{G]k術y=壊x_質{=歴o|ぼ}=ッ先zm吸go{c遍現|_ja叙板tEvs業強zc給o;yOブ}Y堕較|apqろ報yGは
ン流失血猫諾帆妖バ給円償意豊射福レ到滑開膳漸共張往県波悔靴哨煮映歩止思年尺甥ミ則股殻簡労渉島嫁床扶七跳粋
腰1拐奉弱疲誌朝捕ム需紡昏隙甘遜";
cidx="リ3Suェゥュイチ¥x1d−¥x1eッYwオJ2a=Sラb/¥x17U¥x1b_.Ntテマメ5eクUヲH=ュキニンェワ、tOレワ6ェ〉MGョトークN=¥x14A^レト、フリ`ZウZ
sゥ]ユ$&RTO/¥x1a¥x18'Dセ¥x1bこ]Ezシ_¥x0eカOa=Vw5g¥x1a+=Oア¥x19kPイ¥x0dレ`クa「s[ュイラリ4¥x0eサテョロヨラホ¥¥dソvd ン¥x1fハ$ン
Y=.'hニン6¥x0eィ+&f¥x19メki(#ナQ」」^X+mコ¥x0aoァトjヌヲッ)Z!へ¥x11ヨサョヨ`aイ、!_y,_ieーC1ハ」モu¥x0bZロリRョ$「¥x1e マEgン$ーD¥x1a
widx0="Y¥x0b5~UJ¥x1bノルスU¥x0cムD¥x0fヒ¥x08エ¥x1bァリ¥x1c<G・¥x14=イ)?z¥x1f¥x18¥x1b!]3ツ¥x1dルTニ¥x1eXレアヨ=¥x15
1b¥x12xヨk.ヤヲP¥x1f」?t¥x13ウXrモスくN{wナ(AひaA、Aスハ・?ン@sNJSc」_オトォー:¥x1d&ハソ5n¥x1dラメxモ;eo?サ#w¥x0c]ウTJC™¥x15x、
10Kナニ「ョ$Jッォセ。¥x1e¥x08ノロ¥x1eD¥¥ェ5Xキニ」ツM]*?北エモ08ウチェ¥x18ュチッルカsGンメ¥x09k=¥x1f¥"Mコynm¥x0aスウ¥x1eトトォーン?M
widx1="ス¥x15シ¥x17IIt><)T^キodアン¥x0e¥x0aトルx¥x1az¥x0b¥x0d9f¥x0cnウ=ャ!」¥x1cレ@ンtY]クノム¥x10ン/;t_ッ¥x18スナje。¥x14ュ¥
x15ツVモメ¥x1d7ケd¥x0bq¥x0d_UH]wAN¥x1e¥x0bg(Eーッ¥x17ユ¥x1d,.ユミS]ーアノKQフ¥x1d)クUサMdGz>¥"ヤャヲ¥x09mッリvイス9=¥x11]
¥x1cオAゥ]_ゥF!~¥x17リモhFホスoサ¥x10~¥x1b¥x13#6マ¥x13サbt0レF2クJニョ¥x0duTz才2へヒツge¥¥`¥x1dW'7EU1A6ヤ`}ヂメv¥x134¥x1
widx2="イ<¥x1fF「フウ1¥x17ォk・¥x0e¥x186リi.@aマ¥x1eQ_)(wフンユ¥¥g¥x1fHx¥"a¥x0b¥x0dムィクkc+y¥x0fZ¥¥ェt¥x09・r~3II¥x16、F
kanji : 32
行 37 / 141、列 14108
プロパティ
フィルタ
パラメータ
フレーム
init
トゥイーン:
なし
サウンド:
なし
効果:
なし
編集...
ラベルの種類:
名前
同期:
イベント
繰り返し
1
サウンドが選択されていません

今後

- さらに圧縮！
  - 出現頻度を考慮した変数名割り当てアルゴリズムなど
- 成績の記録
  - ローカルストレージには保存できない
  - HTTPで通信