



アルゴリズムとデータ構造

Group Work

スケジュール概要

- **第1回(11/29)**

- 準備 & 開発内容の説明 & グループで作業
 - コースナビ「データ等」のdata.tar.bz2をDL.
- 計画書を提出

- 第2回～3回(12/13, 12/20)：特別講義
- 中間計測(プログラム提出 ✕ 切: 12/22)
- 第4回(2020/1/10): グループで作業
- 最終成果物(プログラム)提出 ✕ 切(2020/1/10)
- 発表資料提出 ✕ 切(2020/1/13)
- 第5回(2020/1/17): 成果発表会
- 報告書提出 ✕ 切(2020/1/24)

評価方法

- グループ全体の評価

- 成果物の性能

- 成果発表の質

- 個別の評価

- グループワーク中の各人の貢献度

評価方法

- グループ全体の評価

- 成果物の性能

- 教員が計測 → 成果発表会の日に公開

- 成果発表の質

- 皆さんが互いに評価, 教員の評価

- 個別の評価

- グループワーク中の各人の貢献度

- 皆さんが互いに評価
 - 計画書, 報告書

評価方法

● グループ全体の評価

○ 成果物の性能

● 教員が計測 → 成果物の性能

○ 成果発表の質

● 皆さんが互いに評価, 教員の評価

評価のポイント:

- 発表はわかりやすかったか？
- 提案手法は妥当だったか？
- 提案手法を正しく実装できていたか？
- 結果の分析は妥当か？

- 各人は、コースナビの“発表評価”から**自分が所属していないグループの中で特に良かった発表に投票する**。(8人のグループと7人のグループで1票の重みに差がありますが、評価の際には正規化します。) 自グループへの投票は無効。
 - また、なぜ投票したのかコメントも書く。
 - コメントは集計後、皆さんにシェアします。

評価方法

● グループ全体の評価

○ 成果

● 教

○ 成果

● 皆

- 各人は、コースナビの“作業評価”からメンバーの貢献度を評価します。
- 貢献度が高いと感じた上位4人を順位をつけて選んでください。
- 自分を選んでもOKです。

● 個別の評価

○ グループワーク中の各人の貢献度

- 皆さんが互いに評価
- 計画書, 報告書



コースナビ利用の練習

- 発表評価(練習)
- 作業評価(練習)

計画書の提出(本日×切)

- 以下の内容を含む文書を作成して、グループの代表者がコースナビ「計画書提出」に提出。
 - グループ名
 - メンバー
 - 作業の目的
 - 作業計画
 - 例)スケジュール表, 作業項目の列挙など.
 - メンバーの役割分担
 - 例)〇〇実装, データ分析, 発表資料作成, 議論, 記録係, 等...
- 分量:A4サイズで1~2枚程度でOK(必要に応じて増量して構わない.)

成果物提出のルール

- 提出するプログラムはC言語で記述すること.
- 提出時に利用してよいコンパイルオプションは
-lm, -O1, -O2, -Wall (wojと同ルール)
- 使用メモリの上限は32GByte.
- 実行ファイル名は「grpwk」とすること.
- Makefileも作ること.
- 計測用のテンプレートを使用すること. (template.c)
 - main_prgは別ファイルに移動してもOK. template.c以外のファイルを用意してもOK.
- 提出:
 - コンパイルに必要なファイルを一つのフォルダにまとめ, tarでアーカイブしてコースナビの「成果物提出」にグループの代表者が提出.
 - フォルダ名はグループIDとする事.
 - 例: グループIDが0の場合は「0.tar」を提出する.

性能評価

- 実行速度 (CPU時間) と精度 (後述) を評価指標とする.
- 実行速度と精度の順位の合計 s に対して, $\text{abs}(s-32)$ を各グループの基本スコアとする.
- 実行速度で順位が1位となったグループのスコアには「8」、2位のグループには「3」を加算. 精度の順位が1位, 2位の場合も同様.
- ただし, 「精度」に関しては, 最下位から数えて3番目までのグループにそれぞれ -10, -7, -2 のペナルティが与えられる.
- 中間計測に参加するグループのスコアには「1」を加算.
(コンパイルの不具合などを確認するためにも, 参加をお勧めします.)
- 例えば, 実行速度で5位, 精度で1位, 中間計測に参加した場合のスコアは, $\text{abs}(5+1-32)+8+1=35$

中間計測

- 参加の是非は自由.
- 2019/12/22, 23:59までに途中結果を提出したグループに関しては, 本番と同様の方法で計測を行って, 結果を公表します.
 - グループIDを知られたくない場合は, コードネームを使用可能なので, 提出時に要望を記して下さい.
- 提出方法: 本番の時と同じフォーマットでコースナビの「中間計測用提出」からグループの代表者が提出.

成果発表

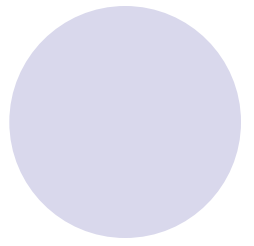
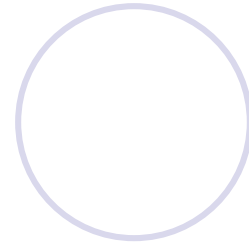
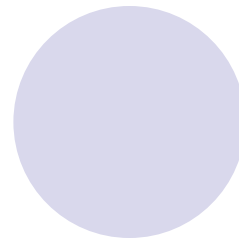
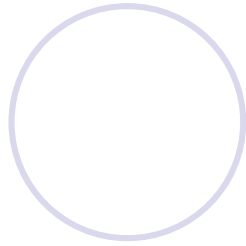
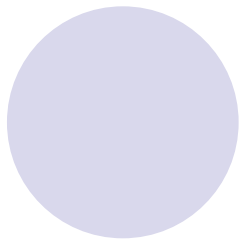
- 成果物に関する発表をする。
 - どのような方針で取り組んだのか？
 - 方針を実現するためにどのような方法論を用いたのか？
その方法論を用いた根拠は？
 - 実際にそれはうまくいったのか？
 - うまくいった(若しくはうまくいかなかった)要因の分析など
- グループの代表者(複数人でも可)が2分で発表
- グループの全員がポスター発表(60～90分を予定)
- 発表資料は1/13までにコースナビの「成果発表会用資料提出」にグループの代表者が提出。(通常のスライドで発表されることを想定していますが、動画などを利用するのもOKですので、その場合は事前にご相談ください。)

報告書の提出

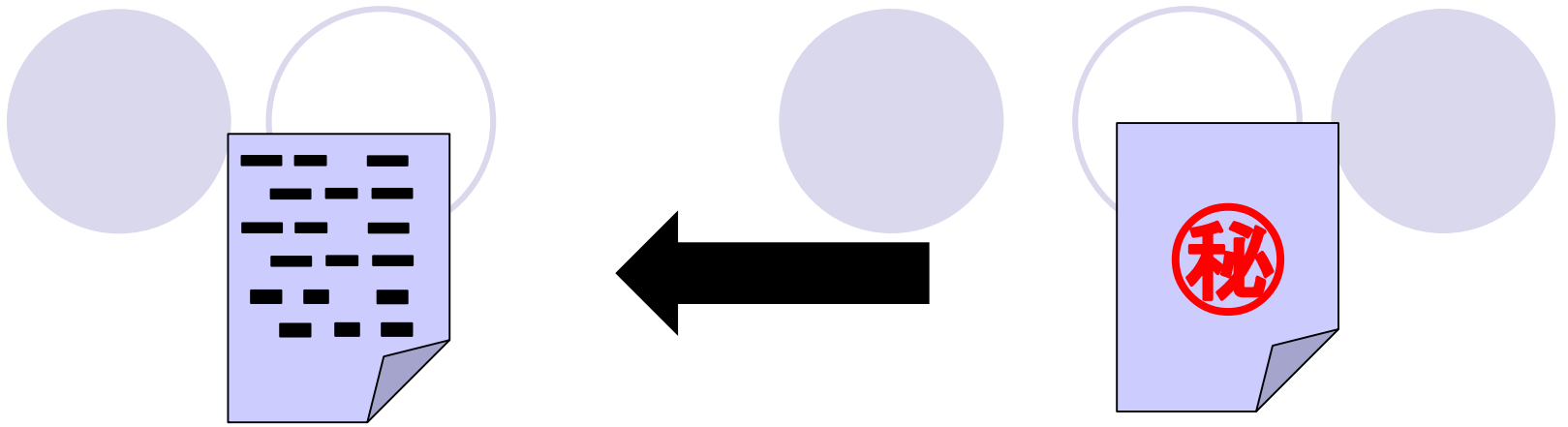
- 以下の内容を含む文書を作成し、2020/1/24までにグループの代表者がコースナビの「報告書提出」に提出。
 - グループ名
 - メンバー
 - 作業の目的と結果
 - 実際に行った作業（作業計画と対比させてもよい）
 - メンバーが実際に分担した役割
 - ディスカッションの議事録要約
 - 考察
 - 作業を進める上で難しかったこと、またそれをどうやって解決したか。
 - 今後同様の取り組みをする際には、どのような改善が望めるか。 など。
- 分量：A4サイズで1～2枚程度でOK（必要に応じて増量して構わない。）

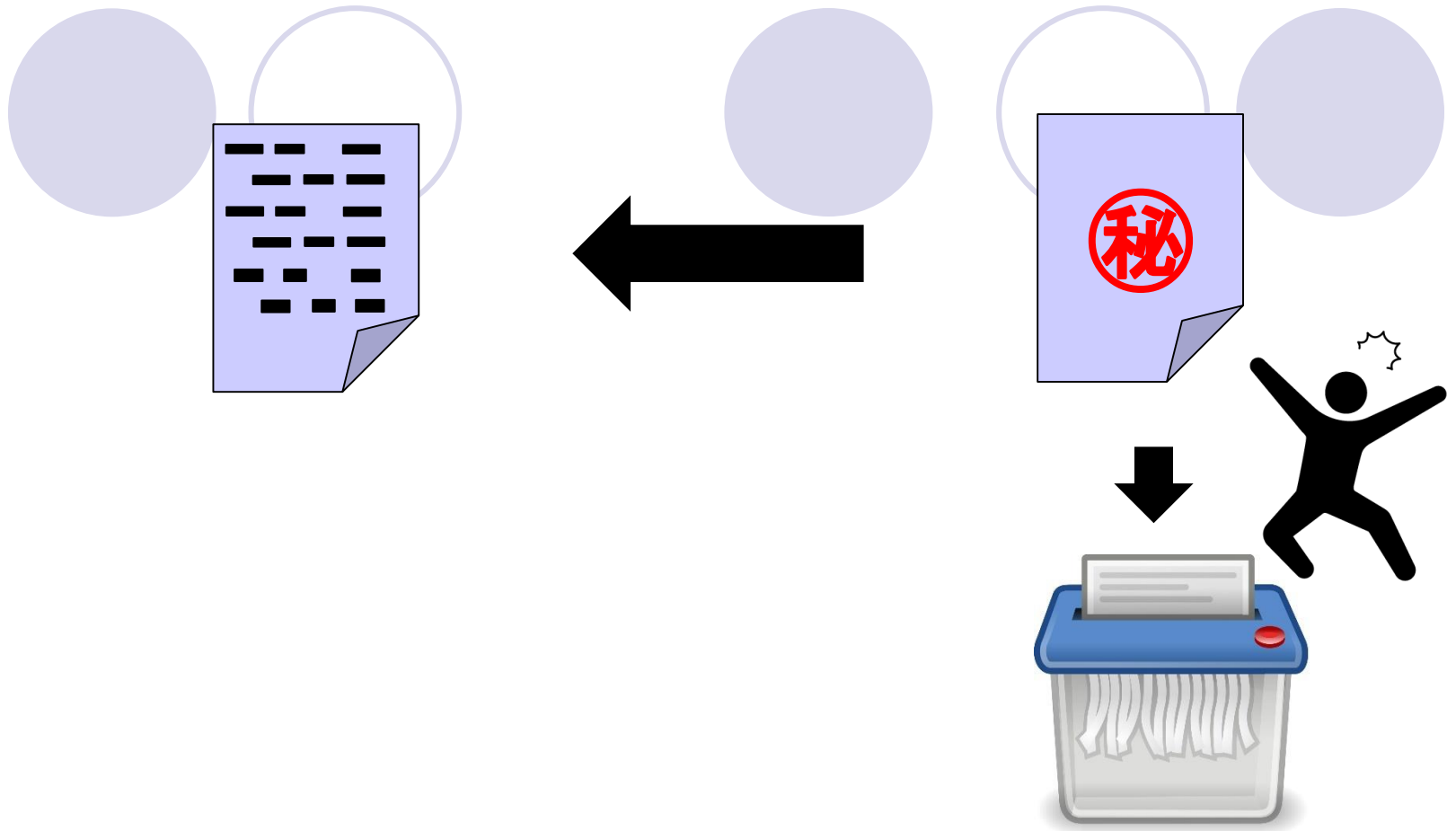
提出〆切 & 作業スケジュール

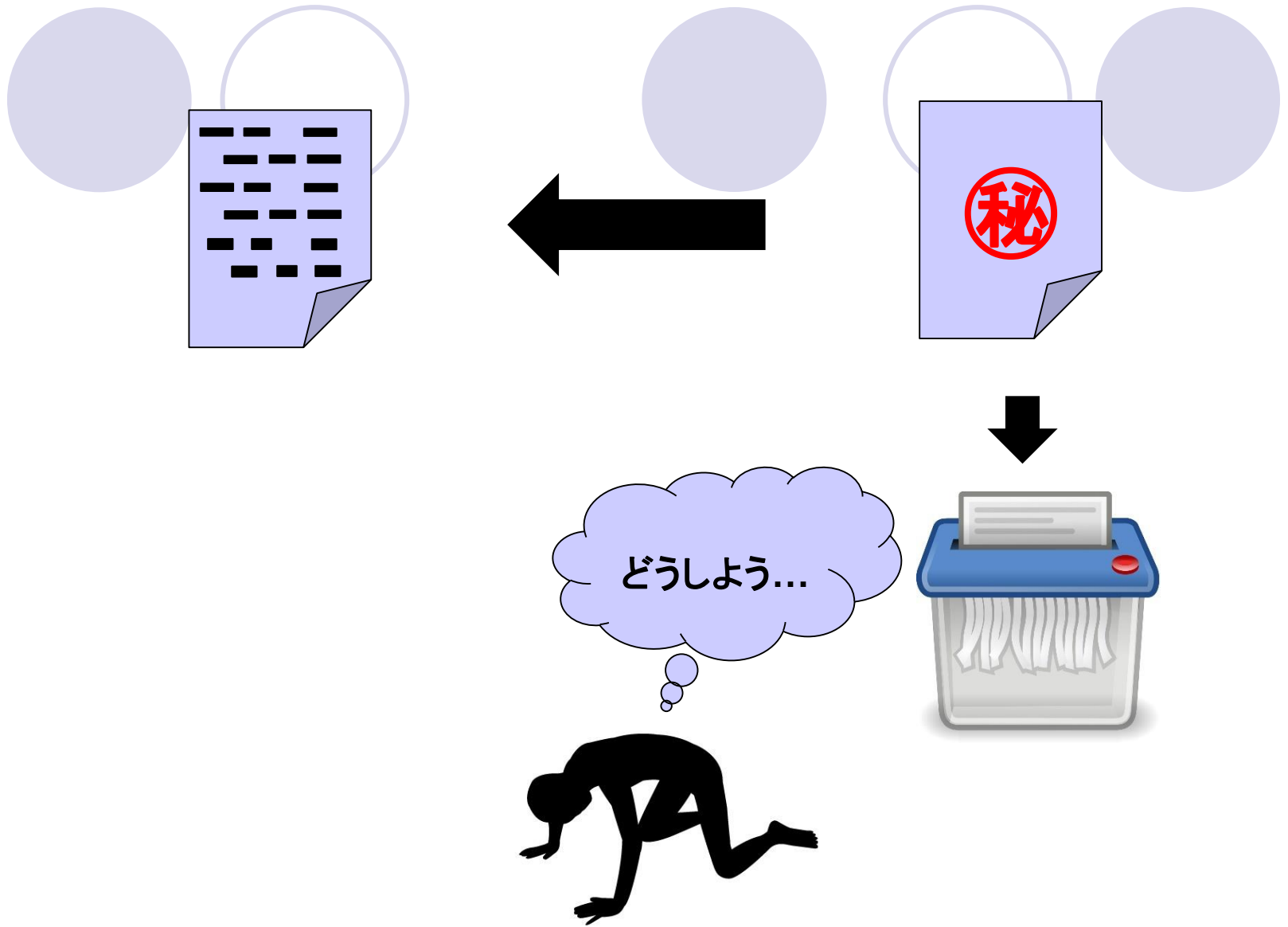
- 2019/11/29, 23:59
 - 提出物: 計画書
 - 提出先: コースナビ「計画書提出」
- 2019/12/22, 23:59 (オプション)
 - 提出物: 中間計測用のプログラム群(tarファイル)
 - 提出先: コースナビ「中間計測用提出」
- 2020/1/10, 23:59
 - 提出物: 最終評価用のプログラム群(tarファイル)
 - 提出先: コースナビ「成果物提出」
- 2020/1/13, 23:59
 - 提出物: 発表資料
 - 提出先: コースナビ「成果発表会用資料提出」
- 2020/1/17, 23:59
 - 授業時間中の作業: 発表評価(コースナビの「発表評価」)
- 2020/1/24, 23:59
 - 提出物: 報告書
 - 提出先: コースナビ「報告書提出」
 - 作業: 作業評価(コースナビの「作業評価」)

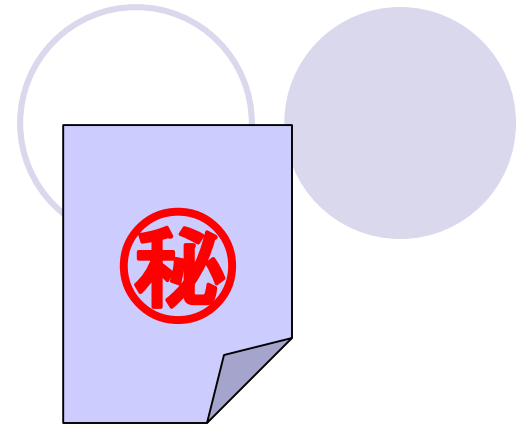
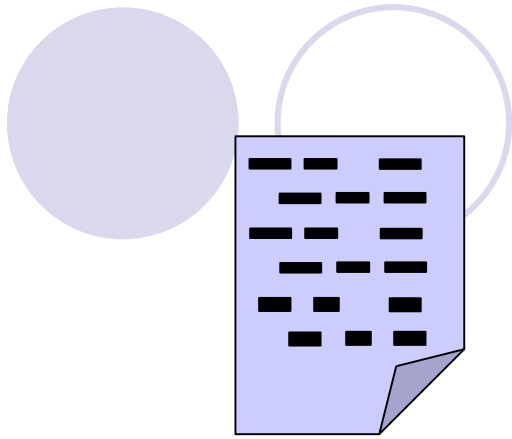


課題：データの復元





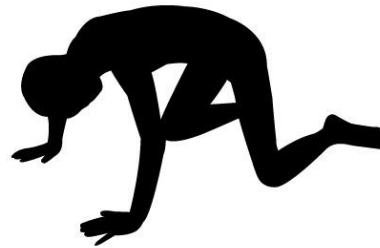




我々におまかせを！



どうしよう...



<http://01.gatag.net/>

- 文字列 $T = t_1, t_2, \dots, t_N$, ($t_i \in \{a, b, c\}$)に以下の加工を加えたデータを想定する.

- 虫食い: 部分的に t_i が 'x' に置換.

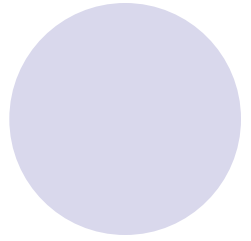
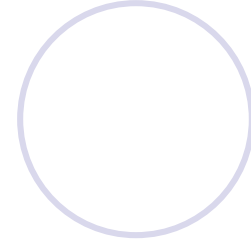
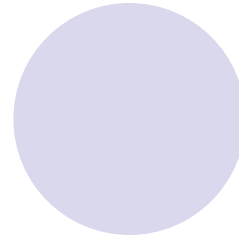
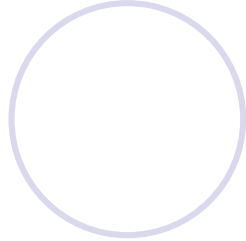
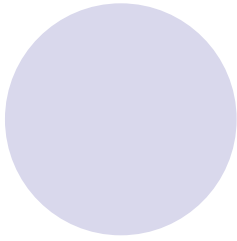
$$\text{i.e.: } T' = t'_1, t'_2, \dots, t'_N, \quad t'_i = \begin{cases} t_i & (t_i \in \{a, b, c, d\}) \\ \text{'x'} & (\text{otherwise}) \end{cases}$$

- 刻む: 部分文字列に分解.

$$\text{i.e.: } \begin{aligned} P &= \{p_1 = 1, p_2, \dots, p_{k-1}, p_k = N + 1\} \quad (p_1 < p_2 < \dots, p_{k-1} < p_k) \\ S &= \{s_i \mid s_i = t_{p_i}, \dots, t_{p_{i+1}-1}, 0 < i < k\} \end{aligned}$$

- 目的:
虫食いデータ(T')と刻んだデータ(S)が与えられたとき, できるだけ速く, そして精度よく元のデータ(T)を復元する.

例



T = bacbadcababcbabacbcacabacaaabdcbbcbbaccbcbaccbbacbc

T' = baxxadcxbxxxxxxxxcbcabxcxaxbdcxbxxbxxbxxcxxxxxxxx

**S = {
acaba ,
d ,
cbbaccbcbaccbbacbc ,
bacbaacaba ,
bcb ,
caaabbcb ,
abacb ,
}**

性能評価

● 実行速度

○ 計測環境(予定)

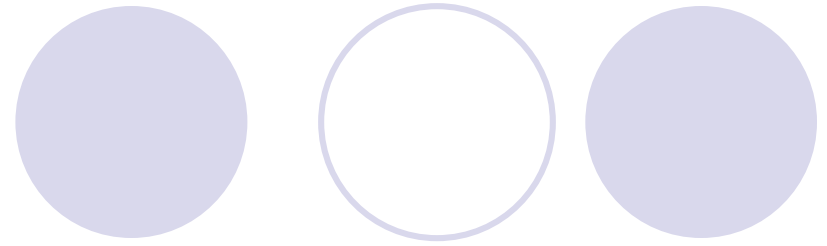
- OS: Ubuntu 18.04LTS
- gcc: v7.3.0
- CPU: i7-6800K CPU @ 3.40GHz

○ 15分で打ち切り

● 精度

○ T と提出物の出力 X の編集距離

データに関して



● サンプルデータ

○ コースナビから入手可能

- datx_in: 入力データ ($x=0, \dots, 4$)
- datx_ref: 正解データ

○ 文字列の長さ: 約 4×10^5

● 本番用のデータ

- サンプルデータを生成したのと同じモデルを用いて作成. (つまり, サンプルデータと同質のデータを使って評価. 文字列長は同じ.)

入力データフォーマット

- ファイル入力
- $T' \langle \text{改行} \rangle s_1 \langle \text{改行} \rangle s_2 \langle \text{改行} \rangle, \dots, s_k \langle \text{改行} \rangle$
- 例:

```
bacbaacdbabcbabacbcacabdcaaabbcbcabacbcbaaccabacbc  
cabacbcbac  
acabaddbab  
abacbcacab
```


出力データフォーマット

- ファイル出力
- 復元した文字列＜改行＞

- 例：

`bacbaacdbabcbacbcddacabacadbabbcbcbbbaccbcbaccbbacbc`

Template.c (1)

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <sys/resource.h>
```

```
int main_prg(int, char**);
```

```
int main(int argc, char** argv){
```

```
    struct rusage u;
    struct timeval start, end;
```

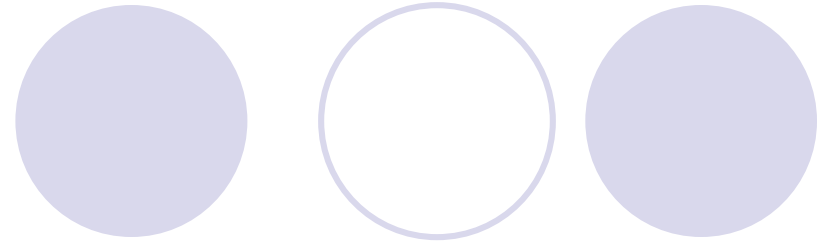
```
    getrusage(RUSAGE_SELF, &u);
    start = u.ru_utime;
```

```
    main_prg(argc, argv);
```

```
    getrusage(RUSAGE_SELF, &u );
    end = u.ru_utime;
```

```
    fprintf(stderr, "%lf\n", (double)(end.tv_sec - start.tv_sec) + (double)(end.tv_usec -
start.tv_usec)*1e-6);
    return(0);
}
```

Template.c (2)



```
int main_prg(int argc, char** argv){
    assert( argc==3 );
    FILE *fp_in = fopen( argv[1], "r" );
    assert( fp_in != NULL );
    FILE *fp_out = fopen( argv[2], "w" );
    assert( fp_out != NULL );

    /** implement here
     *
     * read input values from fp_in
     * write output values to fp_out
     **/

    return 0 ;
}
```

どんな方法で解くか？

- コピー？

- 高速

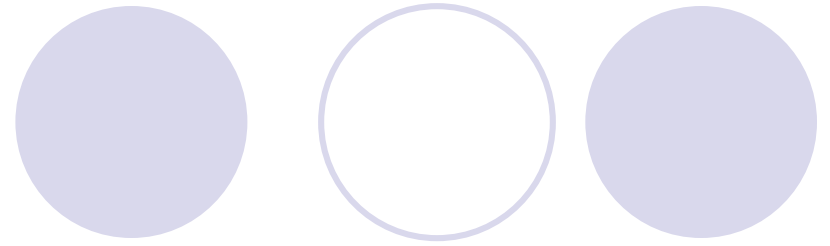
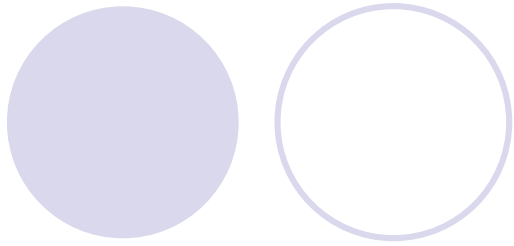
- 精度は悪い

- 全部アラインメントを計算？

- 精度はよい

- 低速

- 高精度で高速な方法は？



- ぜひ活発な議論を

- ソースの共有

- 理工Unixシステム

- (<http://www.waseda.jp/mse/web/pcroom/unix/>)

- `$ ssh USER@muse01.mse.waseda.ac.jp`

- github (<https://github.com/>)

- <https://education.github.com/pack>

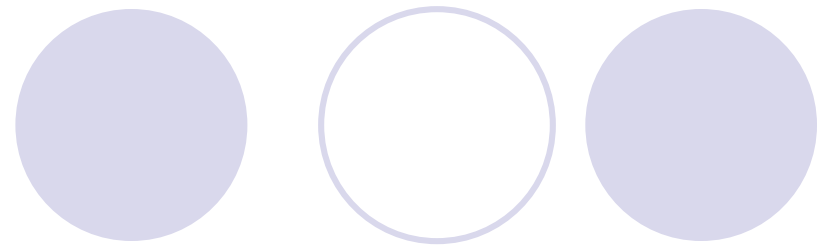
- Dropbox

- Google Drive

成果発表会スケジュール(仮)

- 開会の案内: 14:45～14:50
- ライトニングトーク: 14:50～15:25
- 各種案内: 15:25～15:40
- 休憩: 15:40～15:45
- ポスター発表: 15:45～17:15
- 着席 & 発表評価入力: 17:15～17:35
- 講評: 17:35～17:45
- 結果発表 & まとめ: 17:45～18:00

ライトニングトーク



- グループの代表者が壇上で口頭発表
- グループID順に発表
 - 教壇のPCに発表資料が保存されています.
- 持ち時間: **2分(厳守)**
- 各グループの発表者は、授業開始時には発表者待機エリアに集合していること.

ポスター発表

- 各グループを3つの班(A, B, C)に分けること
 - 各グループが自由な方針で班を決めてOK.
- 15:45～16:15: A班発表
- 16:15～16:45: B班発表
- 16:45～17:15: C班発表
- 発表当番以外の時間は, 別グループの発表を聞きに行くこと.
- 活発な議論を推奨.

※ 時間が限られているので, 一部の人同士で極端に長い時間話し込んでしまわないように. (色々な人と話しましょう.)

ポスター発表用資料の提出 (※切要注意)

- A0サイズ(縦)で白黒で印刷する発表資料をPDF形式でご用意ください。
(A4やA3等で提出してもOKですが、その場合は、拡大印刷します。)
- 印刷の都合上、必ず1/13 12:00 までに提出してください。(提出が遅れた場合はこちらでポスターを用意できません。)
- コースナビの「ポスター発表用資料提出」からご提出ください。

ポスター 作製の例

※ 例は英語ですが、日本語で作成してOKです。

Secure String Pattern Match Based on Wavelet Matrix

Hiroki Sudo¹, Masanobu Jimbo¹, Koji Nuida^{2,3}, Kana Shimizu^{1,2}

1. Department of Computer Science and Engineering, Faculty of Science and Engineering, Waseda University
2. National Institute of Advanced Industrial Science and Technology
3. Japan Science and Technology Agency (JST) PRESTO Researcher



Motivation: Privacy-protection of the biological/medical data is one of the emerging issues in the field of life science, and it is necessary to develop an efficient technology that enables a secure database search. Among the previous methods for this purpose, PBWT-sec is known to be efficient, which is a combined algorithm of a data structure such as BWT and a cryptographic technique called recursive oblivious transfer (ROT). Although the method works well for the string with small alphabet size, its computational cost is linear to alphabet size, and thus it cannot handle an important life science data of large alphabet size such as a protein sequence and a time series data.

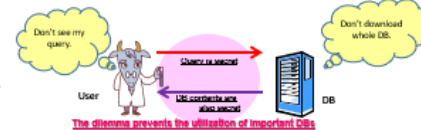
Approach: We propose a novel method combining a wavelet matrix and the ROT. The bottleneck of the previous method is calculation of rank, which requires a computational cost proportional to alphabet size. To reduce the total cost, the proposed algorithm generates several sub-queries for a query character $c \in \Sigma$, each of which is for computing a rank of corresponding bit of binary-encoded c . Since a computational complexity for each sub-query does not depend on alphabet size and only depends on the size of the original string length N , the total computational complexity for calculating rank of c becomes $O(N \log |\Sigma|)$.

Results: We implemented the method and confirmed that the CPU time for searching random string whose alphabet size is 4096 by our method is around 100 times better than that by the previous method, which is concordant with the theoretical complexity. We also show results for searching on Pfam database and a real time series dataset. Those results support the efficiency of our approach and we expected that the approach will be applied for wide range of life science data including clinical data.

Motivation & Aim

Our goal: Developing new string search technology to deal with important life science data of large alphabet size

- Sharing biomedical data, such as personalized genome, clinical information, and industrial secrets like leads compound, is one of the most promising approaches for accelerating life science, however, it also poses serious privacy risks.
- Among the previous methods for privacy-preserving database search, PBWT-sec is known to be efficient, however, it cannot handle large alphabet size data.
- To resolve this drawback, we propose a novel method that uses a **wavelet matrix** for implementing a secure rank dictionary.



Approach

FM-Index

- FM-index is a method for calculating a max match length between two strings.
- Match between two strings is reported as an interval $[f_0, g_0]$.
- An interval $[f_{k+1}, g_{k+1}]$ is computed by the form of:

$$f_{k+1} = \text{Rank}_c(T, f_k) + CF_c(T)$$

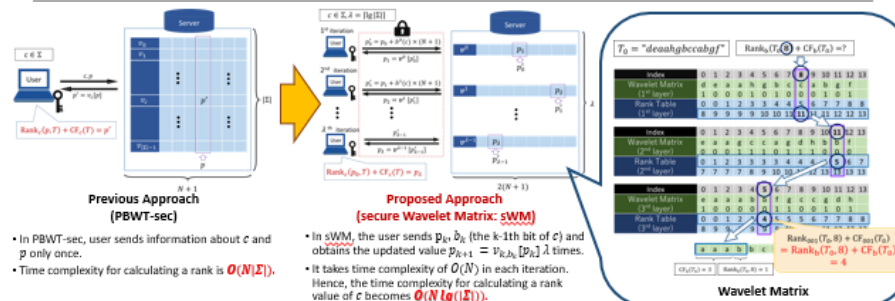
$$g_{k+1} = \text{Rank}_c(T, g_k) + CF_c(T)$$

Query = "ab", T = "abracadabra"



Recursive Oblivious Transfer

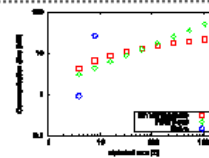
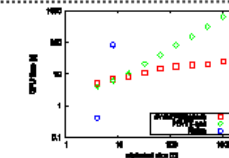
- sWM employs Recursive Oblivious Transfer (ROT) as a secure communication protocol using additive homomorphic encryption.
- Using ROT, when the user has an initial value τ and the server has a vector \mathbf{v} of length N , the user can recursively access \mathbf{v} and obtains only $\mathbf{v}[\tau] \dots \mathbf{v}[\tau] \dots$ without showing any query information to the server while all the intermediate results are concealed to the user.



Experiment

Experimental Setup




- We conducted experiments both on a simulated dataset and on two different real datasets.
- We used protein sequences selected from Pfam and all of the clinical study files stored in JAPIC Clinical Trials Information.
- Tested on Laptop (Intel Core(TM) i7 3.00GHz CPU; total 4 cores with HT; run with two threads) & a compute node (Intel Xeon 3.40GHz CPU; total of 24 cores with HT; run with eight threads)
- Implement Secure Wavelet Matrix by C++ using mcl (Open source library of EC Elasmol).



Complexity of sWM

	Time complexity	Communication size
sWM	$O(N \log \Sigma)$	$O(N \log \Sigma)$
PBWT-sec	$O(N \Sigma)$	$O(N \log \Sigma)$
Naïve approach	$O(N^2)$	$O(N^2)$

N : database string length $|\Sigma|$: character set ℓ : query length

Protein sequences			Clinical data (hiragana)			Clinical data (kanji)		
	Server	Client		Server	Client		Server	Client
	8.28 s	2.56 s		75.9 s	9.32 s		103.7 s	15.2 s
PBWT-sec	12.5 s	1.27 s	PBWT-sec	857 s	9.65 s	PBWT-sec	No measurement	

Simulation data
 $A = 10000$ $|\Sigma| = 4$, 1024 $1 \sim 10$
 Protein sequence data
 $A = 10000$ $|\Sigma| = 21$ $1 \sim 10$
 Clinical data (Hiragana)
 $A = 17152$ $|\Sigma| = 110$ $1 \sim 10$
 Clinical data (Kanji)
 $A = 10000$ $|\Sigma| = 21007$ $1 \sim 10$

成果発表のポイント

- ストーリーがあって、説得力のある発表を心がける事。

- 例えば...

「〇〇による予備調査の結果、〇〇の方針でやるのが良いと考え、それを実現するためには〇〇であることから、〇〇の方法と〇〇の方法を考案した。それらを実装して性能をテストしたところ、〇〇のような結果が得られた。結果を分析したところ〇〇の部分が〇〇であったが、それはデータの性質が〇〇で〇〇の方法がよく適合したためだと考えられる。」

コースナビから発表評価入力(時間厳守)

- 入力時間が限られているので、ポスター発表中にある程度決めておくこと.
- 時間内に必ず入力を
すませること！

評価のポイント：

- ・ 発表はわかりやすかったか？
- ・ 提案手法は妥当だったか？
- ・ 提案手法を正しく実装できていたか？
- ・ 結果の分析は妥当か？

- ・ 各人は、コースナビの“発表評価”から **自分が所属していないグループの中で特に良かった発表に投票する**。(8人のグループと7人のグループで1票の重みに差がありますが、評価の際には正規化します。) 自グループへの投票は無効.
- ・ また、なぜ投票したのかコメントも書く。
- ・ コメントは集計後、皆さんにシェアします。

補足資料: Makefile (1)

```
PROG = grpwk
OBJS = template.o
CC = gcc
CFLAGS = -W -Wall -Wextra -Wconversion -Wshadow
LDFLAGS =

.SUFFIXES: .c

$(PROG): $(OBJS)
    $(CC) $(LDFLAGS) -o $(PROG) $^
.c.o:
    $(CC) $(CFLAGS) -c $<
clean:
    rm $(OBJS) $(PROG)
```

補足資料: Makefile (2)

● コンパイル

```
$ make
```

```
gcc -W -Wall -Wextra -Wconversion -Wshadow -c template.c
```

```
gcc -o grpwk template.o
```

```
$ make
```

make: 'grpwk' は更新済みです.

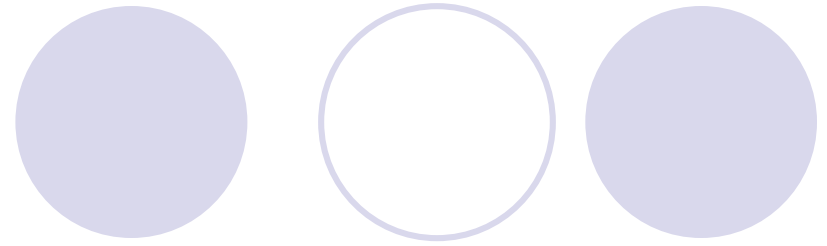
● 新しいファイルの追加

```
OBJS = template.o # ここに追加
```

○ 注意: 「ファイル名.o」で追加してください

```
OBJS = template.o foo.o
```

補足資料: git (1)



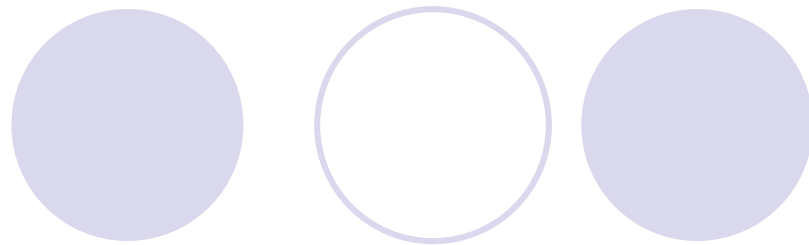
- gitとは？

- プログラムのソースコードなどの変更履歴を記録・追跡するための分散型バージョン管理システム

- 使い方

- レポジトリの作成: `$ git init`
- ファイルの追加: `$ git add foo.c`
- コミット: `$ git commit`
- サーバーにアップロード: `$ git push`
- サーバーから更新: `$ git pull`

補足資料: git (2)



```
$ mkdir my_project
mkdir: ディレクトリ 'my_project' を作成しました
$ cd my_project
$ vim foo.c
$ git add foo.c
$ git commit
[master (root-commit) cba65ce] コミットメッセージ
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 foo.c
$ git remote add origin git@github.com:ユーザー/レポ.git
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 209 bytes | 209.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:ユーザー/レポ.git
 * [new branch]      master -> master
```


補足資料: git (3)

- 使えるサーバー？ <https://github.com>
- 入門

○ <https://techacademy.jp/magazine/6235>

○ <https://qiita.com/nnaquito/items/565f8755e70c51532459>