アルゴリズムとデータ構造

グループワーク

スケジュール概要

- ガイダンス 2020/12/4 14:45 – 16:15
  - 開発内容の説明，メンバー顔合わせ等
  - オンラインにより実施
- 開発・発表準備期間 2020/12/4～2021/1/14
- 発表会準備 2021/1/8 14:45 – 16:15
  - オンラインにより実施
- 成果発表会 2021/1/15 14:45 – 17:45
  - オンラインにより実施
- 報告書等の提出〆切 2021/1/22

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
    - 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
  - 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
  -

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，Moodle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．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
- 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
    - 報告書

- 各人は，Moodle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.

キックオフメモの提出（本日〆切）

- 以下の内容を含む文書を作成して，グループの代表者がMoodle 「キックオフメモ」に提出.
  - グループ番号
  - 代表者（提出係）
  - メンバー
  - 作業計画
    - 例）スケジュール表，作業項目の列挙など.
- PDF形式で作成.
- 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

- プログラムのソースを二つと設定ファイルを提出.
- 提出するプログラムはC言語で記述すること.
- 指定環境（後述）で動作確認すること.
- プログラムは指定の仕様（後述）に従うこと.
- 使用メモリの上限は2Gbyte.
- 提出：
  - 符号化のプログラムはgen_グループ番号.c，復号化用のプログラムはdec_グループ番号.c，設定ファイルをconf _グループ番号.txtとして，Moodleの「成果物提出」に**グループの代表者が提出**.
    - 例： グループ番号が0の場合は gen_0.c, dec_0.c, conf_0.txt を提出する.

性能評価

- 実行速度（CPU時間），容量（後述），費用（後述），精度（後述）を評価指標とする．
- 各グループの得点は以下により求める．
  - 各指標の順位の総和を加算．
  - 各指標の 1 ～ 4 位にはそれぞれ，-10，-5，-2，-1を加算．
  - また，「精度」に関してのみ，最下位から数えて3番目までのグループにそれぞれ 15, 10, 7を加算．
  - 中間計測に参加するグループには，-1を加算．
    （不具合確認のためにも，参加をお勧めします．）
- 例えば，実行速度で1位，容量で5位，費用で6位，精度で20位で，中間計測に参加した場合の得点は，
  1 - 10 + 5 + 6 + 20 + 15 = **37**
- 得点は低いほど良い．

中間計測

- 参加の是非は自由.
- 2020/12/20, 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.
  - グループ番号を知られたくない場合は，コードネームを使用可能．提出時に要望してください.
- 提出方法：本番の時と同じフォーマットでMoodle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
- グループの全員がスライドを用いて，オンライン会議室にて発表（120分程度を予定．）
- 発表資料は2021/1/12までにMoodleの「成果発表会用資料提出」にグループの代表者が提出．（PDFだけでなく，動画でもOK．）
- 成果発表の詳細は後日案内します．

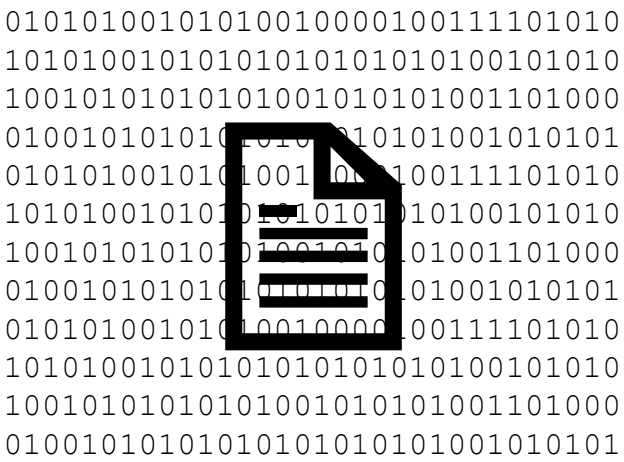
報告書の提出

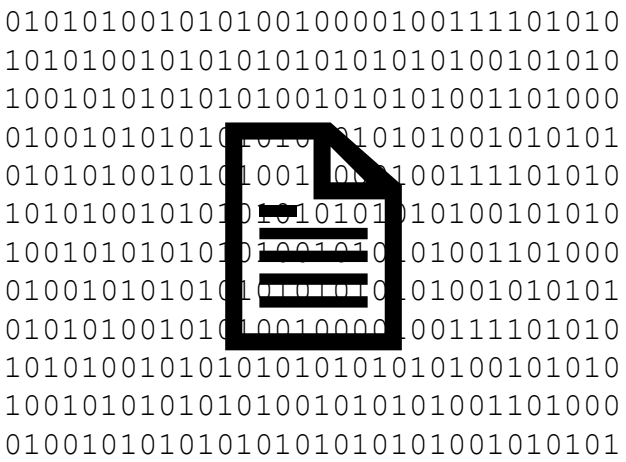
- 以下の内容を含む文書（pdf）を作成し，2021/1/22までに各自がMoodleの「報告書提出」に提出.
  - 提案手法の説明
  - 提案手法の評価
  - 自分貢献（どんな役割を果たしたかを具体的に説明.）
  - 発表会での質疑応答
    - 自分のチームの発表のみならず，他のチームの発表に参加した際の質疑応答についてもまとめる.
  - 考察
    - 作業を進める上で難しかったこと，またそれをどうやって解決したか.
    - 提案手法について，どのような改善が望めるか.　　など.

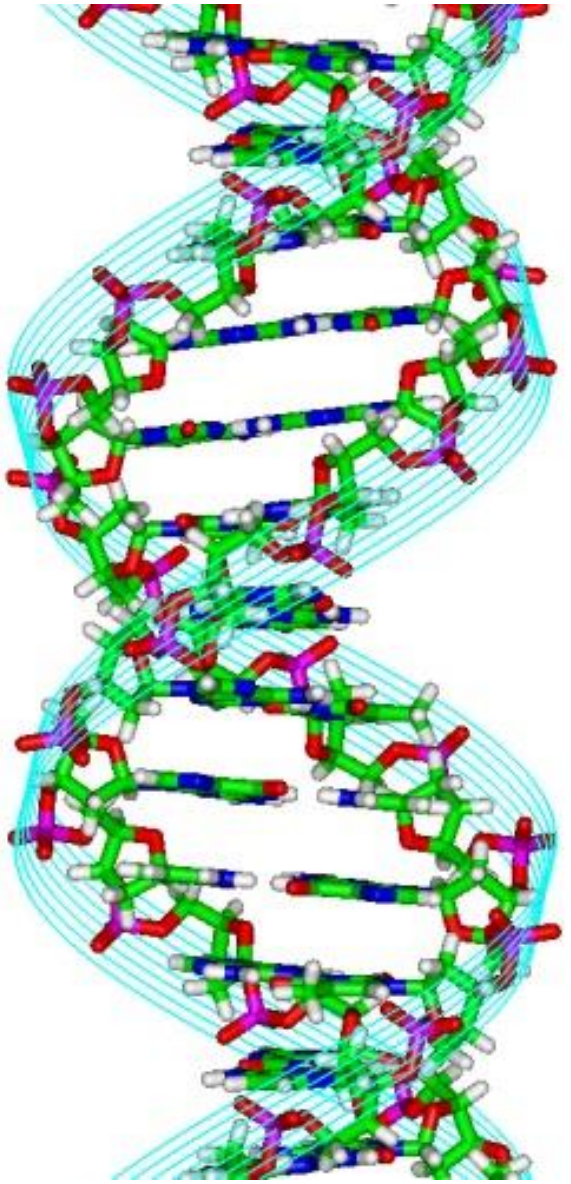
提出〆切＆作業スケジュール

- 2020/12/4, 23:59
  - 提出物：　キックオフメモ
  - 提出先：　Moodle 「キックオフメモ」
- 2020/12/20, 23:59 （オプション）
  - 提出物：　中間計測用のプログラム群（cファイル2つとテキストファイル）
  - 提出先：　Moodle 「中間計測用提出」
- 2021/1/8, 23:59
  - 提出物：　最終評価用のプログラム群（ cファイル2つとテキストファイル）
  - 提出先：　Moodle 「成果物提出」
- 2021/1/12, 23:59
  - 提出物：　発表資料
  - 提出先：　Moodle 「成果発表会用資料提出」
- 2021/1/15, 23:59
  - 作業：　発表評価（ Moodle 「発表評価」）
    ※ ただし，評価したいグループの選択は授業時間中に行うこと.
- 2021/1/22, 23:59
  - 提出物：　報告書
  - 提出先：　Moodle 「報告書提出」
  - 作業：　作業評価（ Moodle 「作業評価」）

課題：DNAストレージ



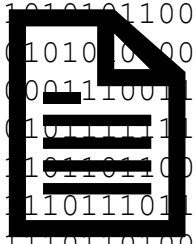




DNA人工
合成装置

DNA塩基配列決定装置
（シークエンサー）

書き込み

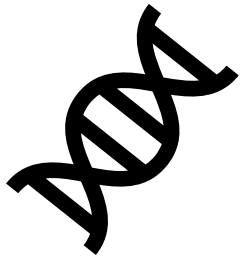


```
1101001111111101110
1010111010110010000
0010110101011000000
1111101010100001101
1111000111001110110
1101001011111111011
1111111011011000011
1000111101110111001
0100111101101000010
1100110101100101100
0100101000
```

符号化

```
TCATTTTCTC
CCTCCGCAAA
GTCCCGAACT
TCCCAATCTT
AATGCGTCGG
CCTTTGTTTT
GTCGACTACT
GTGTGCCATT
CGGACCGCGG
TAGTACAGGA
```

合成



データを4種の
文字で表現

文字列から
データを復元

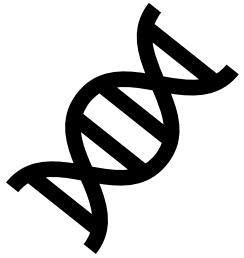
読み出し

```
1101001111111101110
1010111010110010000
0010110101011000000
1111101010100001101
1111000111001110110
1101001011111111011
1111111011011000011
1000111101110111001
0100111101101000010
1100110101100101100
0100101000
```

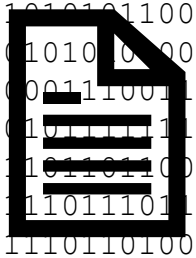
復号化

```
TCATTTTCTC
CCTCCGCAAA
GTCCCGAACT
TCCCAATCTT
AATGCGTCGG
CCTTTGTTTT
GTCGACTACT
GTGTGCCATT
CGGACCGCGG
TAGTACAGGA
```

読み取り



書き込み

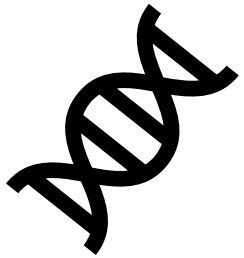


```
1101001111111101110
1010111010110010000
0010110101011000000
1111101010100001101
1111000111001 10110
1101001 1111111011
1111111011011000011
1000111101110111001
0100111101101000010
1100110101100101100
0100101000
```

符号化

```
TCATTTTCTC
CCTCCGCAAA
GTCCCGAACT
TCCCAATCTT
AATGCGTCGG
CCTTTGTTTT
GTCGACTACT
GTGTGCCATT
CGGACCGCGG
TAGTACAGGA
```

合成



データを4種の
文字で表現

文字列から
データを復元

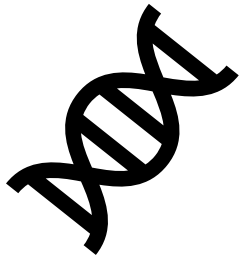
読み出し

```
1101001111111101110
1010111010110010000
0010110101011000000
1111101010100001101
1111000011100110110
1101001011111111011
1111111011011000011
1000111101110111001
0100111101101000010
1100110101100101100
0100101000
```

復号化

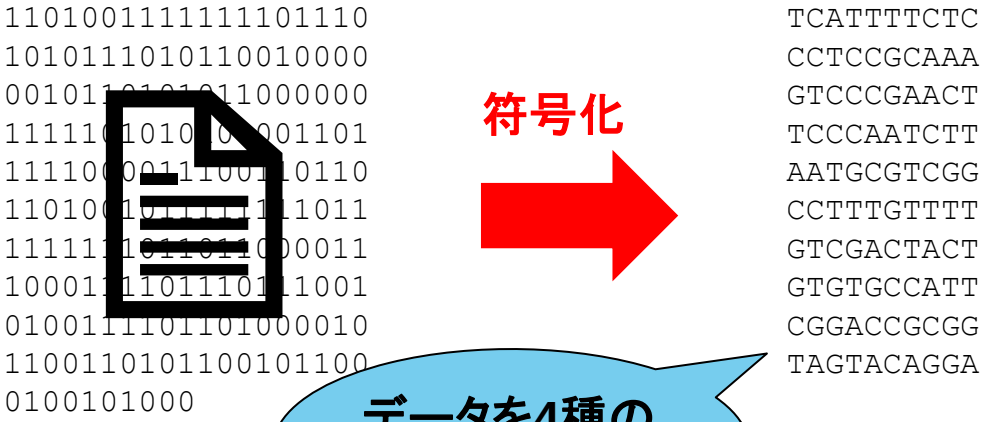
```
TCATTTTCTC
CCTCCGCAAA
GTCCCGAACT
TCCCAATCTT
AATGCGTCGG
CCTTTGTTTT
GTCGACTACT
GTGTGCCATT
CGGACCGCGG
TAGTACAGGA
```

読み取り



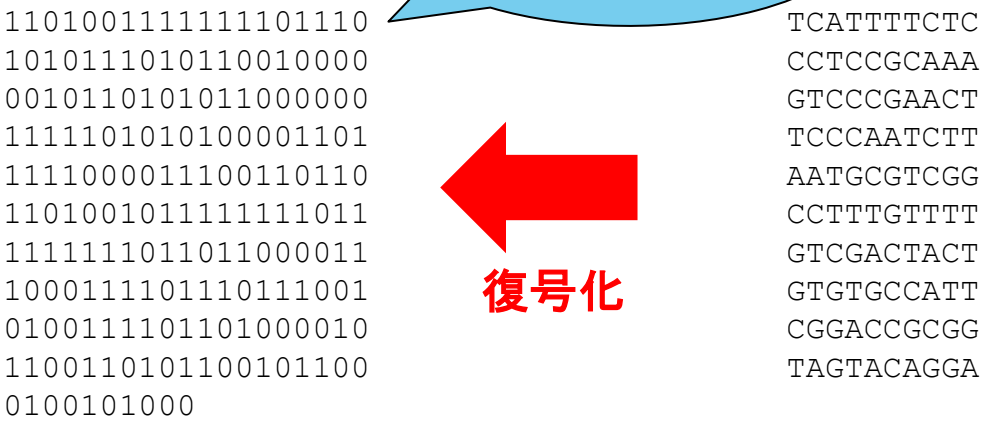
<u>性能の良い</u>符号化・復号化の手法を考案してほしい.

書き込み



データを4種の
文字で表現

読み出し



文字列から
データを復元

読み取りには
色々と問題がある.

<u>性能の良い</u>符号化・復号化の手法を考案してほしい.

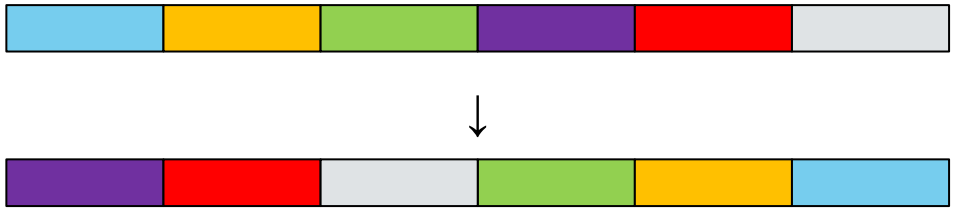
読み取りの際の問題

- NP方式
  - 挿入，欠損エラーが生じる．
  - 同じ文字が続いていると間違えやすくなる．

TCATTTTCTC
↓
TCATTTTTCTC

- BS方式
  - 置換エラーが生じる．
  - 続けて25文字しか読み取れない．
  - 25文字がシャッフルされて出力される．

TCATTTTCTC
↓
TCATTATCTC



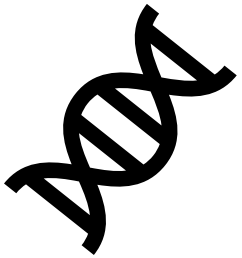
書き込み

```
1101001111111101110
1010111010110010000
0010110101011000000
1111101010100001101
1111000111001110110
1101001011111111011
1111111011011000011
1000111101110111001
0100111101101000010
1100110101100101100
0100101000
```

符号化

```
TCATTTTCTC
CCTCCGCAAA
GTCCCGAACT
TCCCAATCTT
AATGCGTCGG
CCTTTGTTTT
GTCGACTACT
GTGTGCCATT
CGGACCGCGG
TAGTACAGGA
```

合成



データを4種の文字で表現

エラーを含む文字列からデータを復元

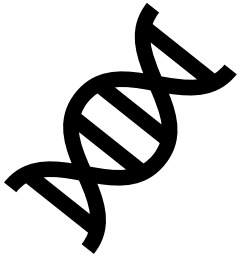
読み出し

```
1101001111111101110
1010111010110010000
0010110101011000000
1111101010100001101
1111000111001110110
1101001011111111011
1111111011011000011
1000111101110111001
0100111101101000010
1100110101100101100
0100101000
```

復号化

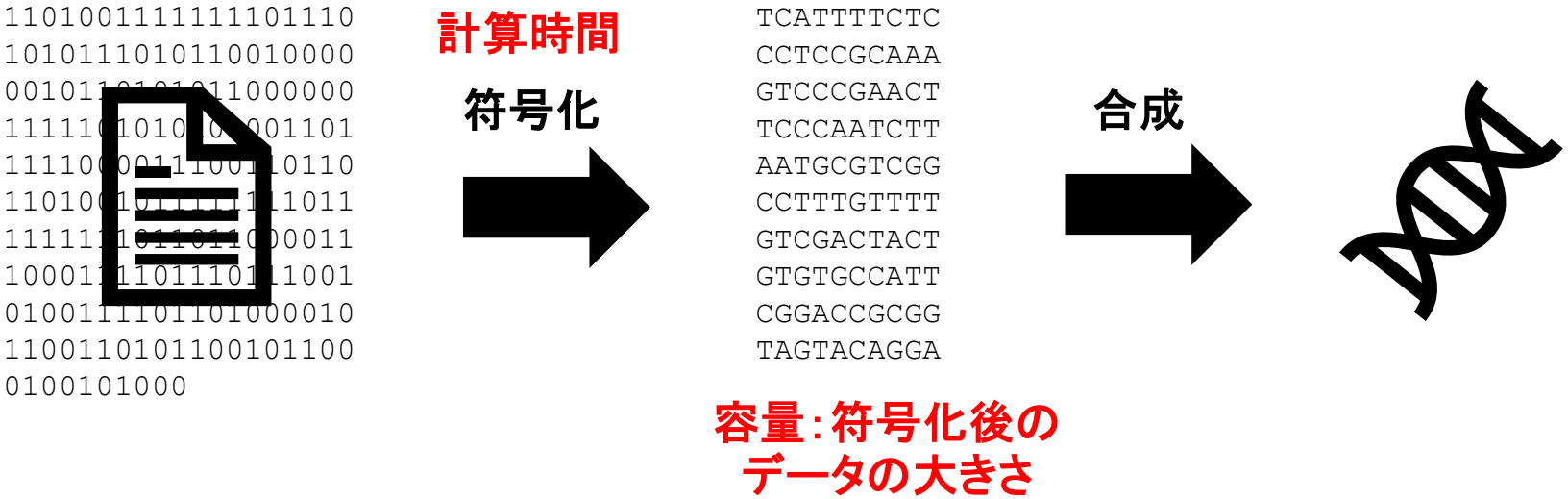
```
TCATTTTCTC
CCTCCGCAAA
GTCCCGAACT
TCCCAATCTT
AATGCGTCGG
CCTTTGTTTT
GTCGACTACT
GTGTGCCATT
CGGACCGCGG
TAGTACAGGA
```

読み取り



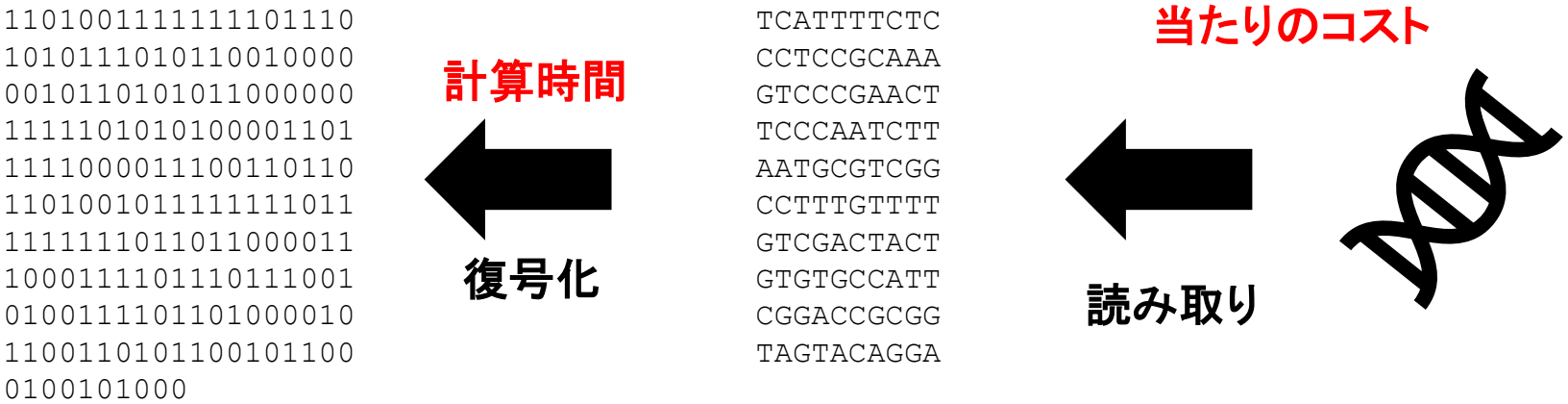
<u>性能の良い</u>符号化・復号化の手法を考案してほしい.

書き込み



精度:元データと復元データ
のハミング距離

読み出し



性能の良い符号化・復号化の手法を考案してほしい.

- 元のデータ
  ビット列 $B = b_1, b_2, \ldots, b_N,\ (b_i \in \{1,0\})$
- 符号化後のデータ（要作成）
  塩基配列 $X = x_1, x_2, \ldots, x_M,\ (x_i \in \{A, C, G, T\})$
- シークエンスデータ
  読み取り方を指定可能（要指定）
  塩基配列 $S = s_1, s_2, \ldots, s_L,\ (x_i \in \{A, C, G, T\})$
- 復号化後のデータ（要作成）
  ビット列 $V = v_1, v_2, \ldots, v_N,\ (v_i \in \{1,0\})$
- 評価指標
  - 計算時間： X とV の生成時間の合計，容量： M
  - 精度：B とV のハミング距離，費用：後述

- N ： 2×10^5, $M < 10^8$ とする.
- NP方式
  - エラー生起確率：同一塩基の連続： 1/16, それ以外：1/256, 挿入と欠損は同確率
  - 費用： 10^4文字ごとに1 （ 10^4未満でも1）
    - 15000文字読む費用は2
- BS方式
  - エラー生起確率：1/10, 置換のみ
  - $x_0, x_{25}, \ldots$ から始まる25文字がランダムにシャッフルされる. 25文字に満たない末尾の文字列にはランダムな配列が追加される.
  - 費用： 10^4文字ごとに2 （ 10^4未満でも2）
    - 15000文字読む費用は4

例（N=100の場合）

元データ

```
1101100110110101000011000011100110001000110000100101010110001001110010100110011101101000101101100111
```

符号化したデータ

```
TCGCGTCCAATAATGCGAGATAAGCCCCGAGCTAGGCGCTCGGAGTCGCT
```

シークエンスデータ

```
TCGCGTCCAATAATGCGAGATAAGCCCCGAGCGAGGCGCTTGGAGTCGCT
```

復元したデータ

```
1101100110110101000011000011100110001000110000100101010110001001110010100110011101101000101101100111
```

容量:50
費用:2(bs)， 1(np)
元データと復元したデータのハミング距離:2

計測に関して

- 実行速度
  - 計測環境（予定）
    - OS: Ubuntu 16.04LTS
    - gcc: v5.4.0
    - CPU: Xeon E52643-v3 or Core i7-6700K
  - 20分で打ち切り

配布物

- プログラム
  - grpwk20.tar　　※ tar xvf grpwk20.tarで解凍
  - 一連の手順を実行できるプログラム群
- データ
  - Grpwk20_data_smpl.tar
  - 各データの例
  - ただし，これらは上記のプログラムで生成できる．

配布プログラム

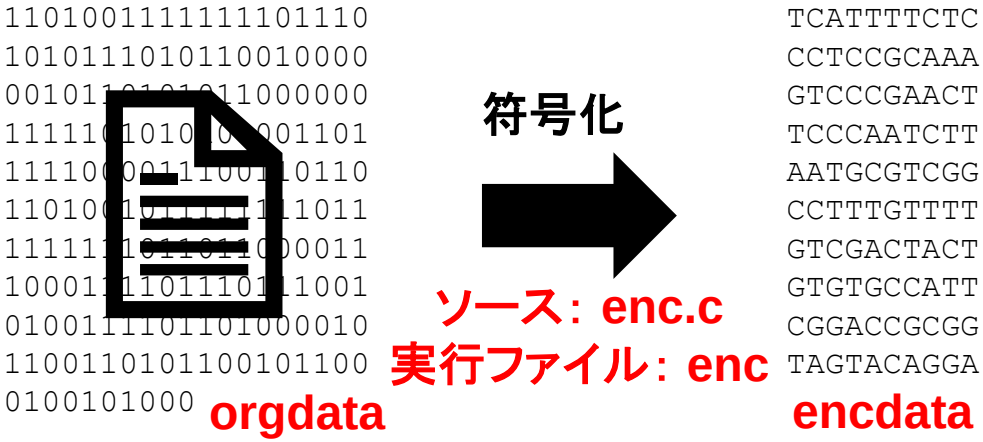
- gen.c
  - 入力ファイル：なし，出力ファイル：orgdata
- enc.c （要作成． 配布物は単純に2ビットを一塩基に変換しただけ．）
  - 入力ファイル：orgdata，出力ファイル：encdata
- syn.c
  - 入力ファイル：encdata，出力ファイル：syndna
- seq.c
  - 入力ファイル：syndna，出力ファイル：seqdata
- dec.c（要作成． 配布物はseqdataの一行目のみを復号しただけ． ）
  - 入力ファイル：seqdata，出力ファイル：decdata
- eval.c
  - 入力ファイル：orgdata, decdata

※ 入出力ファイルは全て，実行ファイルと同じ場所に置くこととする．

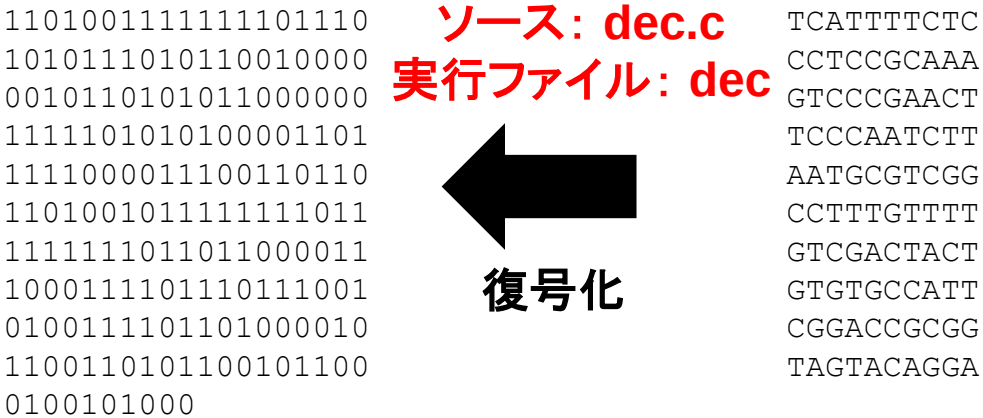
入出力ファイル仕様

- orgdata : ‘0’, ‘1’から成る文字列．末尾に改行.
- encdata : ’A’, ‘C’, ‘G’, ‘T’から成る文字列．末尾に改行.
- syndna : ’A’, ‘C’, ‘G’, ‘T’から成る文字列．末尾に改行.
- seqdata : A’, ‘C’, ‘G’, ‘T’から成る複数の文字列.
  - 500M Byte以下とすること.
- decdata : ‘0’, ‘1’から成る文字列．末尾に改行.

書き込み



読み出し　decdata



実行方法

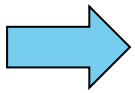
tar xvf grpwk20.tar
cd grpwk20
make
chmod 755 test.sh
./test.sh 1 0 0 0 0 0

※ test.shの引数は，seqの引数と同じとする．

提出に関して

- 符号化，復号化プログラムはそれぞれ，一つのC言語ファイルとしてください.
  プログラム内でgrpwk20.hは含めてOK.
- 設定ファイル
  - 1行目：ライブラリのオプション-lmの使用の有無（使用する場合1, 使用しない場合0）
  - 2行目：seqの引数
- 設定ファイルの例と実行例

```
1
1 0 0 1 0 0
```



```
gcc gen.c –o gen –lm
seq 1 0 0 1 0 0
```

seq（読み取り）の仕様

- 引数
  - 1番目：bsの読み取り回数
  - 2番目：bsの読み取り開始位置　※位置は0から数える.
  - 3番目：bsの読み取り長
  - 4番目：npの読み取り回数
  - 5番目：npの読み取り開始位置　※位置は0から数える.
  - 6番目：npの読み取り長

  ※読み取り長に0を指定した場合は，末尾まで読み取られる.
- 出力：
  - 出力はseqdataに書き込まれる
  - 一回当たりの読み取り結果は一行の文字列
  - bs，npの順に出力される

（例）0番目から25文字をbsで1回読みたい

- 実行コマンド： seq 1 0 25 0 0 0
- 入力：
  CGTTGCAATGCAATTCGCGCAATATAGTAGGGA
  AGGATTTTATCGAACTA
- 出力：
  CGTTGCAATGCCATTCGCGCAATAT

（例）0番目から25文字をbsで2回読みたい

- 実行コマンド： seq 2 0 25 0 0 0
- 入力：
  CGTTGCAATGCAATTCGCGCAATATAGTAGGGA
  AGGATTTTATCGAACTA
- 出力：
  CGTTGCAATGCCATTCGCGCAATAT
  CGTTGCAATGCCATTCGCGCAATAT

（例）0番目から25文字をnpで1回読みたい

- 実行コマンド： seq 0 0 0 1 0 25
- 入力：
  CGTTGCAATGCAATTCGCGCAATATAGTAGGGA
  AGGATTTTATCGAACTA
- 出力：
  AGTAGGGAAGGATTTATCGAACTA
  ※ 削除エラーがあり25文字より短くなっている.

（例）0番目から25文字をbsで1回読み，25番目から25文字をnpで1回読みたい

- 実行コマンド： seq 1 0 25 1 25 25
- 入力：
  GTAAAAGGTCACACTTCTTCCCGAGCGGGACGCAAATAACTGCACTTTAG
- 出力：
  GTAAAAGGTCACACTTCTTCCCGAG
  CGGGACGCAAATAACTGCACTTTAG

動作確認

- 提出物は，以下の環境で動作することを確かめてから提出すること．
- OS　：ubuntu-20.04.1-desktop-amd64
  開発：build-essential

- Linux環境をお持ちでない方は，**インストールして利用することを推奨**しますが，インストールせずに利用することもできます．（次を参照）

1. https://releases.ubuntu.com/20.04/ より ubuntu-20.04.1-desktop-amd64.iso を入手
2. VirtualBoxをインストール
3. 仮想マシンの作成

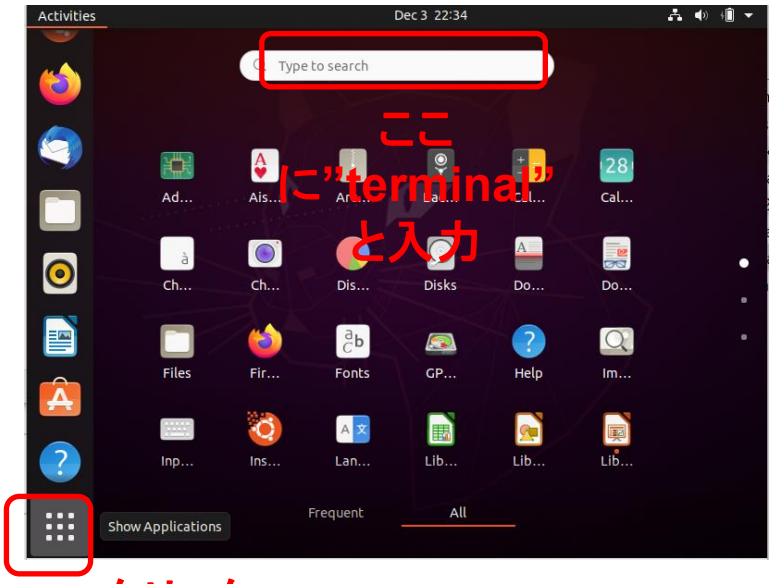
4. isoイメージを設定

「光学ドライブ」をクリックして
「ディスクファイルを選択」をクリックした後で
ubuntu-20.04.1-desktop-amd64.isoを設定

5. 仮想マシンの電源を入れる

クリック

6. Live CDを起動



クリック

クリック

7. terminalを立ち上げて以下を実行

sudo apt-get install build-essential

enc.c, dec.cの初期状態

- enc.c
  - 先頭から２ビットを一塩基に置換するだけ
- dec.c
  - enc.cと同じ規則で先頭から一塩基を2ビットに置換するだけ
- bsで読んだ場合
  - 50ビット（文字）ずつシャッフルされたまま出力. 置換エラーもそのまま出力.
- npで読んだ場合
  - 途中で削除，挿入があると，出力文字にずれが生じる．評価はハミング距離なので，先頭近くで一文字ずれるとほぼ全滅.

どんな方法で解くか？

- 多少冗長でも，エラーに強い符号化の方法はあるだろうか？
- encからdecに伝えたい情報は，すべて塩基配列（encdata）に埋め込まないといけない．何か有用な情報を埋め込めるだろうか？
- 同じ場所を複数回読みだせば精度は上がるだろうが，コストはかさむ．

- ぜひ活発な議論を
- ソースの共有
  - github (https://github.com/)
  - Dropbox
  - Google Drive

グループディスカッション

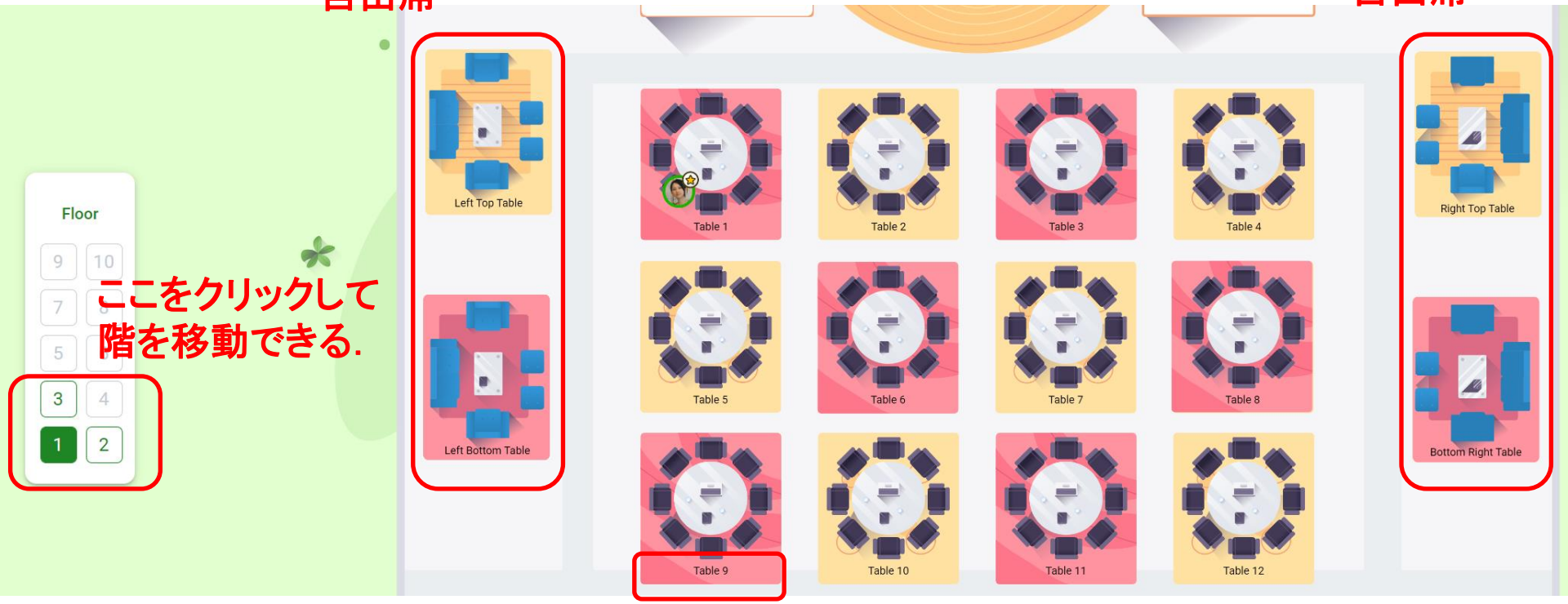
- 各人のグループ番号はgroup.pdfに記載.
- オンライン会議システムremoを使います。
- 着席の方法がややこしいので，ご協力お願いします！

着席の方法

- イベントに入ると，システムが勝手にテーブルを選んで着席させます.
- **自分のグループの番号のテーブルに移動してください.**
- ただし！各テーブルには８名までしか着席できないので，移動する場合は，**2〜3階の「free」のテーブルか，ソファ席を経由して，自分のテーブルの席が空いたら移動**してください.

自由席

自由席



ここをクリックして
階を移動できる.

グループ9
のテーブル

※ 1階と2階の一部はグループの席. 2階の一部と3階は自由席.