アルゴリズムとデータ構造

DNAストレージの符号化と復号

13

1W232012　安部　倫太朗
1W232098　金子　大河
1W232257　永田　健人
1W233013　浦沢　峻太郎
1W233019　大戸　暢丈

解法

私たちのグループ13では符号化したデータの各文字をn回連続で出力した.そのデータを読み取って連続した同じ文字による文字列の長さをnで割った値に一番近い整数個分のその文字を選択するというやり方で進めていった.

実験

n=5～34の各値について起こるエラーの数を計測した.1000回実行した際のエラーが起こった回数は以下のようになった.なお1000回の実行にあたり作成したコードを実行した様子も以下に掲載した.

考察

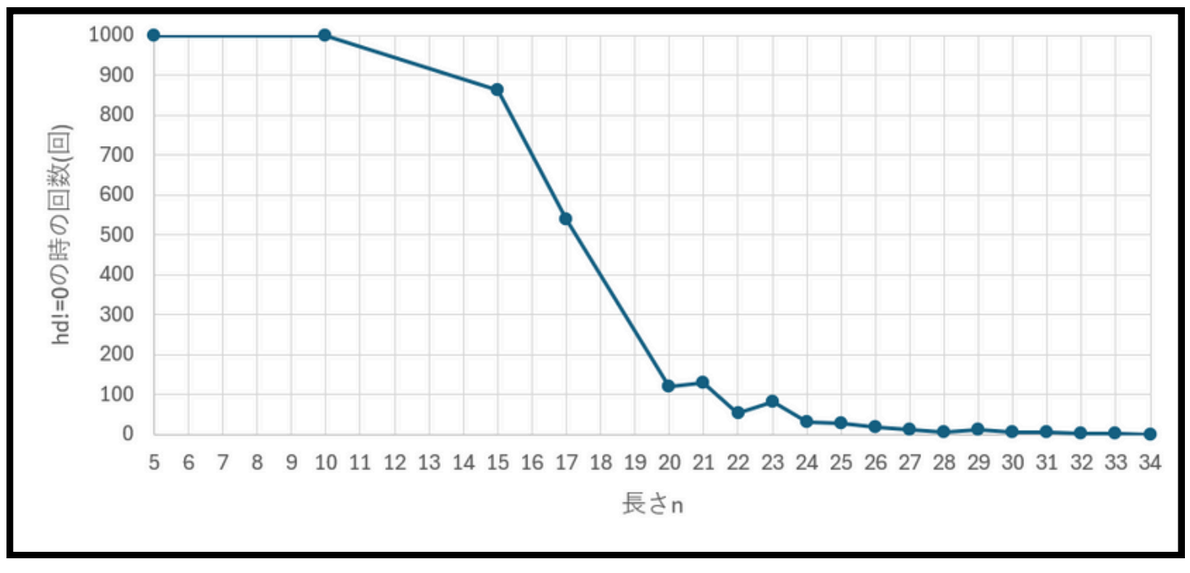
dec_13.c内のround_to_nearest関数を見ると一番近い整数を取る際に連続した文字の個数をnで割った値を四捨五入しているが言語の特性上,個数が奇数の時に(2N+1)/2=Nとなり個数が2Nの時と同じ振る舞いをする.よって,ある偶数とその次の奇数の振る舞いが同じようであると考えられる.

n=28にした理由

n=28程度からエラーの発生割合は1%を切っていて割と単純な戦略でもあり,n=30のコスト3000は他グループと被る可能性を考慮しそれに勝るために上記の考察と合わせ,n=29でなくn=28を採用した.

```
Running test(n=28): 987
Running test(n=28): 988
Running test(n=28): 989
Running test(n=28): 990
Running test(n=28): 991
Running test(n=28): 992
Running test(n=28): 993
Running test(n=28): 994
Running test(n=28): 995
Running test(n=28): 996
Running test(n=28): 997
Running test(n=28): 998
Running test(n=28): 999
Running test(n=28): 1000
The number of failure is: 5
```

長さn	hd!=0の時の回数
5	1000
10	1000
15	861
17	537
20	119
21	130
22	53
23	82
24	32
25	29
26	17
27	13
28	5
29	11
30	5
31	5
32	1
33	1
34	0



上記コードのGITHUB

アルゴリズムとデータ構造

DNAストレージの符号化と復号

13

棄却された方法

今回の課題であるエラー修復(ECC)に対し,私たち上に示した方法以外に様々な方法を実験した.BS方式を利用すると挿入・欠損エラーがないため,BSの25文字の置換修復だけでhd=0が得ることができる.これを元に以下の方法を考察した.

1.ハミング符号

ハミング符号とは、以下の図のような5行5列の行列を考え(1,1)をパリティビットとして用意する.A,T,G,Cを0～3の各数字に対応させる.(1,1)を除く緑色のマスであるパリティビットにはその行,列の総和に対する4の剰余を格納する.また(1,1)の紫色のマスには全体の総和の4に対する剰余を格納する.
全てをBS方式で実行しようとした際に得た25文字のブロックが全体のどの位置に存在するのかを保存するために0～3の4つの数字を水色の7ビット分確保すれば4^7-1分の位置を指定できる.パリティビットを除いた残りの黄色の9ビットに塩基情報を格納する.

1		1	0	2	0
	2	0	2	3	2
1	2	0	0	1	0
0	1	2	2	0	3
2	3	0	1	1	1
0	1	1	3	1	2

2	0	2	3	2	
2	1	0	1	0	
1	2	2	0	3	
3	0	1	3	1	
1	1	3	1	2	

このやり方と問題点を左の図をこのまま用いて解説する.と左の灰色の部分はその行,列の総和の4に対する剰余を表している.剰余が0になっていない行,列はエラーが起こっていると判断できる.

左の図では2行,4行と2行,4行がそれぞれ+1,+2の誤差ができていると判断できるため(2,2)成分が1で(4,4)成分が3であるというエラーが判定できる.
しかし右のような場合に(4,4)と(5,5)が-1で(4,5)と(5,4)が+1となると各行各列で総和が変わらなくなりエラーに気づくことができない.

1		0	0	0	0
	2	0	2	3	2
0	2	1	0	1	0
0	1	2	2	0	3
0	3	0	1	2	2
0	1	1	3	2	1

2.リードソロモン符号

リードソロモン符合とは優れた符号化率,最小距離のトレードオフ関係を持つ符号である.検査行列とシンドロームを用いて誤差ベクトルを求めていく手法である.データが小さい際の誤差ベクトルは手計算で求めることはできたが今回の問題に適用することができず断念した.

3.HEDGES

HEDGESとは元々2020年の7月に論文として発表されたECCの一種で,リードソロモンを含むアルゴリズムにより,np方式より厳しい条件(1/10で挿入と欠損)に耐える.HEDGESは元々C++でかつ外部ライブラリを大量に使用していたため,本課題のCでの実装を断念した.

HEDGES/hamming　code参考文献