CEITEC

Central European Institute of Technology

BRNO | CZECH REPUBLIC

Introduction to Bioinformatics (LF:DSIB01)

Week 5 : Dynamic Programming

Sequence Alignment – Hamming Distance (1950)

- Given two equally sized sequences, what is the distance between the sequences?

```
Hamming Distance = number of mismatches
ATCAGGTCATGCAG
ATCAGGACATGCAC

Hamming Distance fails with insertion
ATCAGGTCATGCAG
AtTCAGGACATGCA
```

Sequence Alignment – Levenshtein Distance (1965)

Given two potentially related sequences (of any length), what is the distance between the sequences?

Distance = minimum number of edits needed to get from A to B
Edits = Substitution of letter, addition of letter, deletion of letter

A = “ACGTCATCA”
B = “TAGTGTCA”

Scoring Function

Score(AB) = Total cost of editing A to B
Cost of substitution (mutation) (-1)
Cost of indel (-1)
Reward for match (+1)

What is the best alignment (highest score)?
How do you calculate it?

Sequence Alignment – Levenshtein Distance (1965)

```
Score = -5
A C G T C A T C A
T A G T G T C A _
```

Sequence Alignment – Levenshtein Distance (1965)

Score = -5

A C G T C A T C A

T A G T G T C A _

Score = +1

A C G T C A T C A

T A G T G _ T C A

Sequence Alignment – Levenshtein Distance (1965)

```
Score = -5
A C G T C A T C A
T A G T G T C A _

Score = +1
A C G T C A T C A
T A G T G _ T C A

Score = +2
_ A C G T C A T C A
T A _ G T G _ T C A
```

Sequence Alignment

Way too many sequences for Brute Force enumeration

$$\binom{n+m}{m} \quad \frac{(m+n)!}{(m!)^2} \approx \frac{2^{m+n}}{\sqrt{\pi m}}$$

n	Enumeration
10	184,756
20	1.40E+11
100	9.00E+58

Coin Change (Euro)

Convert an amount of money into the fewest number of coins

Input: Amount of money (M)
Output: the smallest number of 50c (a), 20c (b), 10c (c), 5c (d), 2c (e) and 1c (f) such that 50a+20b+10c+5d+2e+1f = M

```
1  while M > 0
2      c ← Largest coin that is smaller than (or equal to) M
3      Give coin with denomination c to customer
4      M ← M − c
```

Try: M=60c; M=55c; M=40c

Coin Change (Generalised)

Input: An amount of money M, and an array of d denominations $\mathbf{c} = (c_1, c_2, \ldots, c_d)$, in decreasing order of value $(c_1 > c_2 > \cdots > c_d)$.

Output: A list of d integers $i_1, i_2, \ldots, i_d$ such that $c_1 i_1 + c_2 i_2 + \cdots + c_d i_d = M$, and $i_1 + i_2 + \cdots + i_d$ is as small as possible.

Coin Change (US)

Input: An amount of money M, and an array of d denominations $\mathbf{c} = (c_1, c_2, \ldots, c_d)$, in decreasing order of value ($c_1 > c_2 > \cdots > c_d$).

Output: A list of d integers $i_1, i_2, \ldots, i_d$ such that $c_1 i_1 + c_2 i_2 + \cdots + c_d i_d = M$, and $i_1 + i_2 + \cdots + i_d$ is as small as possible.

```
BETTERCHANGE(M, c, d)
1  r ← M
2  for k ← 1 to d
3        i_k ← r/c_k
4        r ← r − c_k · i_k
5  return (i_1, i_2, ..., i_d)
```

NB: Division = “floor”

Try

M = 40; c1=25, c2=10, c3=5, c4=1

Coin Change (US)

Input: An amount of money M, and an array of d denominations $\mathbf{c} = (c_1, c_2, \ldots, c_d)$, in decreasing order of value $(c_1 > c_2 > \cdots > c_d)$.

Output: A list of d integers $i_1, i_2, \ldots, i_d$ such that $c_1 i_1 + c_2 i_2 + \cdots + c_d i_d = M$, and $i_1 + i_2 + \cdots + i_d$ is as small as possible.

```
BETTERCHANGE(M, c, d)
1  r ← M
2  for k ← 1 to d
3        i_k ← r/c_k
4        r ← r − c_k · i_k
5  return (i_1, i_2, ..., i_d)
```

NB: Division = “floor”

M = 40; c1=25, **c2=20**, c3=10, c4=5, c5=1

Discontinued 1875 for being too confusing

Pseudocode Exercise: Coin Change (US coins)

Input: An amount of money M, and an array of d denominations $\mathbf{c} = (c_1, c_2, \ldots, c_d)$, in decreasing order of value $(c_1 > c_2 > \cdots > c_d)$.

Output: A list of d integers $i_1, i_2, \ldots, i_d$ such that $c_1 i_1 + c_2 i_2 + \cdots + c_d i_d = M$, and $i_1 + i_2 + \cdots + i_d$ is as small as possible.

```
BETTERCHANGE(M, c, d)
1  r ← M
2  for k ← 1 to d
3        i_k ← r / c_k
4        r ← r − c_k · i_k
5  return (i_1, i_2, ..., i_d)
```

NB: Division = “floor”

BetterChange
40=
1x25 + 1x10 + 1x5=
3 coins

Incorrect!
40 = 2x20=
2 coins

M = 40; c1=25, **c2=20**, c3=10, c4=5, c5=1

Discontinued
1875
for being too
confusing

Pseudocode Exercise: Coin Change

Input: An amount of money M, and an array of d denominations $\mathbf{c} = (c_1, c_2, \ldots, c_d)$, in decreasing order of value ($c_1 > c_2 > \cdots > c_d$).

Output: A list of d integers $i_1, i_2, \ldots, i_d$ such that $c_1 i_1 + c_2 i_2 + \cdots + c_d i_d = M$, and $i_1 + i_2 + \cdots + i_d$ is as small as possible.

Tries every combination
Guaranteed to find optimal
Slow

Brute Force Algorithm

Input: An amount of money M, and an array of d denominations $\mathbf{c} = (c_1, c_2, \ldots, c_d)$, in decreasing order of value $(c_1 > c_2 > \cdots > c_d)$.

Output: A list of d integers $i_1, i_2, \ldots, i_d$ such that $c_1 i_1 + c_2 i_2 + \cdots + c_d i_d = M$, and $i_1 + i_2 + \cdots + i_d$ is as small as possible.

```
BRUTEFORCECHANGE(M, c, d)
1  smallestNumberOfCoins ← ∞
2  for each (i_1, ..., i_d) from (0, ..., 0) to (M/c_1, ..., M/c_d)
3      valueOfCoins ← Σ_{k=1}^{d} i_k c_k
4      if valueOfCoins = M
5          numberOfCoins ← Σ_{k=1}^{d} i_k
6          if numberOfCoins < smallestNumberOfCoins
7              smallestNumberOfCoins ← numberOfCoins
8              bestChange ← (i_1, i_2, ..., i_d)
9  return (bestChange)
```

Tries every combination
Guaranteed to find optimal
Slow

Dynamic Programming (Bellman, 1954)

1 cent, 5 cents, 10 cents

1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	1	2	3	4	5	1	2

12	13	14	15	16	17	18	19	20	21	22
3	4	5	2	?						

Value of n

Pick the lowest number from:

1 + Val(n-1)
1 + Val(n-5)
1 + Val(n-10)

Val(16)

Min()

1+Val(15) = 1+2 = 3
1+Val(11) = 1+2 = 3
1+Val(6) = 1+2 = 3

Val(16) = 3

Val(17) = ?

Dynamic Programming

Min
V(39)
V(35)
V(30)
V(20)
V(15)
+1

BetterChange
40=
1x25 + 1x10 + 1x5=
3 coins

Incorrect!
40 = 2x20=
2 coins

M = 40; c1=25, **c2=20**, c3=10, c4=5, c5=1

Discontinued
1875
for being too
confusing

1	2	3	4	5	6	7	8	9	10
1	2	3	4	1	2	3	4	5	1

11	12	13	14	15	16	17	18	19	20
2	3	4	5	**2**	3	4	5	6	**1**

21	22	23	24	25	26	27	28	29	30
2	3	4	5	1	2	3	4	5	**2**

31	32	33	34	35	36	37	38	39	40
3	4	5	6	**2**	3	4	5	**6**	?

Backpropagation

What is the sequence that brought you here?

Min
V(39) = cent
V(35) = five
V(30) = ten
V(20) = twenty
V(15) = twefive
+1

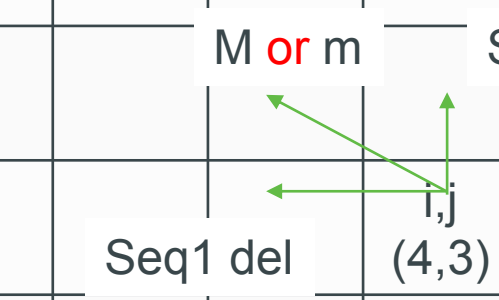
1	2	3	4	5	6	7	8	9	10
1 (cent)	2 (cent)	3	4	1 (five)	2 (cent)	3	4	5	1 (ten)

11	12	13	14	15	16	17	18	19	20
2	3	4	5	**2 (ten)**	3	4	5	6	**1 twenty**
21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
2	3	4	5	1 twefive	2	3	4	5	**2 (ten)**
31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
3	4	5	6	**2 (ten)**	3	4	5	**6 (cent)**	?

Needleman–Wunsch algorithm (1970)

Seq1 = “ACGTCATCA”
Seq2 = “TAGTGTCA”
Scoring Function
Mut / In / Del (-1)
Match (+1)

i,j	Seq1	A (i=1)	C (i=2)	G	T	C	A	T	C	A
Seq2	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
T (j=1)	-1									
A (j=2)	-2									
G	-3									
T	-4									
G	-5									
T	-6									
C	-7									
A	-8									



Seq1 = “ACGTCATCA”
Seq2 = “TAGTGTCA”
Scoring Function
Mut / In / Del (-1)
Match (+1)

i,j	Seq1	A (i=1)	C (i=2)	G	T	C	A	T	C	A
Seq2	0	-1 (L)	-2 (L)	-3	-4	-5	-6	-7	-8	-9
T (j=1)	-1 (U)	-1 (d)	-2 (d)	-3 (d)	-2 (d)					
A (j=2)	-2 (U)									
G	-3									
T	-4									
G	-5									
T	-6									
C	-7									
A	-8									

Seq1 = “ACGTCATCA”
Seq2 = “TAGTGTCA”
Scoring Function
Mut / In / Del (-1)
Match (+1)

i,j	Seq1	A (i=1)	C (i=2)	G	T	C	A	T	C	A
Seq2	0	-1 (L)	-2 (L)	-3	-4	-5	-6	-7	-8	-9
T (j=1)	-1 (U)	-1 (d)	-2 (d)	-3 (d)	-2 (d)	-3	-4	-5	-6	-7
A (j=2)	-2 (U)	0 (d)	-1 (L)	-2	-3	-4	-2	-3	-4	-3
G	-3	-1	-1	0 (d)	-1	-2	-3	-3	-4	-4
T	-4	-2								
G	-5	-3								
T	-6	-4								
C	-7	-5								
A	-8	-6								

Seq1 = “ACGTCATCA”
Seq2 = “TAGTGTCA”
Scoring Function
Mut / In / Del (-1)
Match (+1)

i,j	Seq1	A (i=1)	C (i=2)	G	T	C	A	T	C	A
Seq2	0	-1 (L)	-2 (L)	-3	-4	-5	-6	-7	-8	-9
T (j=1)	-1 (U)	-1 (d)	-2 (d)	-3 (d)	-2 (d)	-3	-4	-5	-6	-7
A (j=2)	-2 (U)	0 (d)	-1 (L)	-2	-3	-4	-2	-3	-4	-3
G	-3	-1	-1	0 (d)	-1	-2	-3	-3	-4	-4
T	-4	-2	-2	-1	+1	0	-1	-2	-3	-4
G	-5	-3	-3	-1	0	0	-1	-2	-3	-4
T	-6	-4	-4	-2	0	-1	-1	0	-1	-2
C	-7	-5	-3	-3	-1	1	-1	-1	1	0
A	-8	-6	-4	-4	-2	0	2	1	0	2

Backpropagation

Seq1 = “ACGTCATCA”
Seq2 = “TAGTGTCA”
Scoring Function
Mut / In / Del (-1)
Match (+1)

i,j	Seq1	A (i=1)	C (i=2)	G	T	C	A	T	C	A (i=9)
Seq2	0	-1 (L)	-2 (L)	-3	-4	-5	-6	-7	-8	-9
T (j=1)	-1 (U)	-1 (d)	-2 (d)	-3 (d)	-2 (d)	-3	-4	-5	-6	-7
A (j=2)	-2 (U)	0 (d)	-1 (L)	-2	-3	-4	-2	-3	-4	-3
G	-3	-1	-1	0 (d)	-1	-2	-3	-3	-4	-4
T	-4	-2	-2	-1	+1	0	-1	-2	-3	-4
G	-5	-3	-3	-1	0	0	-1	-2	-3	-4
T	-6	-4	-4	-2	0	-1	-1	0	-1	-2
C	-7	-5	-3	-3	-1	1	-1	-1	**1**	0
A (j=8)	-8	-6	-4	-4	-2	0	2	1	0	**2 (d)**

[9,8] diag A A
[8,7] ?

Backpropagation

Seq1 = “ACGTCATCA”
Seq2 = “TAGTGTCA”
Scoring Function
Mut / In / Del (-1)
Match (+1)

i,j	Seq1	A (i=1)	C (i=2)	G	T	C	A	T	C	A (i=9)
Seq2	0	-1 (L)	-2 (L)	-3	-4	-5	-6	-7	-8	-9
T (j=1)	-1 (U)	-1 (d)	-2 (d)	-3 (d)	-2 (d)	-3	-4	-5	-6	-7
A (j=2)	-2 (U)	0 (d)	-1 (L)	-2	-3	-4	-2	-3	-4	-3
G	-3	-1	-1	0 (d)	-1	-2	-3	-3	-4	-4
T	-4	-2	-2	-1	+1	**0**	-1	-2	-3	-4
G	-5	-3	-3	-1	0	0	**-1**	-2	-3	-4
T	-6	-4	-4	-2	0	-1	-1	**0**	-1	-2
C	-7	-5	-3	-3	-1	1	-1	-1	**1**	0
A (j=8)	-8	-6	-4	-4	-2	0	2	1	0	**2 (d)**

[9,8] diag AA
[8,7] diag CC
[7,6] diag TT
[6,5] diag AG (m)

Backpropagation

Seq1 = “ACGTCATCA”
Seq2 = “TAGTGTCA”
Scoring Function
Mut / In / Del (-1)
Match (+1)

i,j	Seq1	A (i=1)	C (i=2)	G	T	C (i=5)	A	T	C	A (i=9)
Seq2	**0**	-1 (L)	-2 (L)	-3	-4	-5	-6	-7	-8	-9
T (j=1)	**-1** (U)	-1 (d)	-2 (d)	-3 (d)	-2 (d)	-3	-4	-5	-6	-7
A (j=2)	-2 (U)	**0** (d)	**-1** (L)	-2	-3	-4	-2	-3	-4	-3
G	-3	-1	-1	**0** (d)	-1	-2	-3	-3	-4	-4
T (4)	-4	-2	-2	-1	**+1**	**0**	-1	-2	-3	-4
G	-5	-3	-3	-1	0	0	**-1**	-2	-3	-4
T	-6	-4	-4	-2	0	-1	-1	**0**	-1	-2
C	-7	-5	-3	-3	-1	1	-1	-1	**1**	0
A (j=8)	-8	-6	-4	-4	-2	0	2	1	0	**2 (d)**

Score = 2

[9,8] diag AA
[8,7] diag CC
[7,6] diag TT
[6,5] diag AG (m)
[5,4] left C_
[4,4] diag TT
[3,3] diag GG
[2,2] left C_
[1,2] diag AA
[0,1] up _T

```
_ACGTCATCA
TA_GT_GTCA
-+-++--+++
```

Local Alignment

- Local Alignment

```
Score = 3
ACGTCATCA
TAGTG_TCA

Score = 4
AC__GTCATCA
TAGTGTCA___
```

Smith-Waterman algorithm (1981)

Seq1 = “ACGTCATCA”
Seq2 = “TAGTGTCA”
Scoring Function
Mut / In / Del (-1)
Match (+1)
or ZERO

i,j	Seq1	A (i=1)	C (i=2)	G	T	C	A	T	C	A
Seq2	0	0	0	0	0	0	0	0	0	0
T (j=1)	0									
A (j=2)	0			M or m		Seq2 del				
G	0		Seq1 del		i,j (4,3)					
T	0									
G	0									
T	0									
C	0									
A	0									

Smith-Waterman algorithm (1981)

Seq1 = “ACGTCATCA”
Seq2 = “TAGTGTCA”
Scoring Function
Mut / In / Del (-1)
Match (+1)
or ZERO

i,j	Seq1	A (i=1)	C (i=2)	G	T	C	A	T	C	A
Seq2	0	0	0	0	0	0	0	0	0	0
T (j=1)	0	0	0	0	1	0	0	1	0	0
A (j=2)	0	1	0	0	0	0	1	0	0	1
G	0	0	0	1	0	0	0	0	0	0
T	0	0	0	0	2	0	0	1	0	0
G	0	0	0	1	1	1	0	0	0	0
T	0	0	0	0	2	1	0	1	0	0
C	0	0	1	0	1	3	2	1	2	1
A	0	0	0	0	0	2	4	3	2	3

Score: ATCAGGT vs ATTCAGG

Global Alignment

		A	T	C	A	G	G	T
A								
T								
T								
C								
A								
G								
G								

Local Alignment

		A	T	C	A	G	G	T
A								
T								
T								
C								
A								
G								
G								

Score: ATCAGGT vs ATTCAGG

Global Alignment

		A	T	C	A	G	G	T
	0	-1	-2	-3	-4	-5	-6	-7
A	-1							
T	-2							
T	-3							
C	-4							
A	-5							
G	-6							
G	-7							

Local Alignment

		A	T	C	A	G	G	T
	0	0	0	0	0	0	0	0
A	0							
T	0							
T	0							
C	0							
A	0							
G	0							
G	0							

Score: ATCAGGT vs ATTCAGG

Global Alignment

A_TCAGGT
ATTCAGG_
OR
AT_CAGGT
ATTCAGG_

		A	T	C	A	G	G	T
	0	-1	-2	-3	-4	-5	-6	-7
A	-1	**1**	0	-1	-2	-3	-4	-5
T	-2	**0**	**2**	1	0	-1	-2	-3
T	-3	-1	**1**	0	-1	-2	-3	-1
C	-4	-2	0	**2**	1	0	-1	-2
A	-5	-3	-1	1	**3**	2	1	0
G	-6	-4	-2	0	2	**4**	3	2
G	-7	-5	-3	-1	1	3	**5**	**4**

Local Alignment

TCAGG
TCAGG
OR
AT_CAGG
ATTCAGG

		A	T	C	A	G	G	T
	0	0	0	0	0	0	0	0
A	0	1	0	0	1	0	0	0
T	0	0	2	1	0	0	0	1
T	0	0	1	1	0	0	0	1
C	0	0	0	2	1	0	0	0
A	0	1	0	1	3	2	1	0
G	0	0	0	0	2	4	3	2
G	0	0	0	0	1	3	5	4

Binding Matrix

- Binding Matrix instead of Match/Mismatch
- How do you change the algorithm?

a	water				
	A	C	G	U	T
ALA	0.2	0.9	1.0	0.4	0.3
ARG	-0.8	-1.1	-1.2	-0.5	0.3
ASN	-1.0	-0.7	-0.6	-0.7	-0.7
ASP	0.5	2.4	1.0	2.8	3.7
CYSH	-1.2	-0.2	-0.9	-0.8	-0.6
GLN	-2.0	-0.5	-2.2	-1.5	-1.9
GLU	2.0	1.6	-0.3	2.2	3.7
HISA	-1.8	-0.9	-1.8	-1.5	-1.9
HISB	-1.2	-0.2	-1.9	-0.6	-1.2
HISH	-0.7	0.0	-0.6	0.5	0.2
ILE	-2.8	-1.9	-2.6	-1.5	-1.6
LEU	-2.0	-1.9	-2.6	-1.8	-1.7
LYSH	0.9	1.4	-0.5	0.6	0.9
MET	-2.5	-1.4	-2.2	-2.1	-2.3
PHE	-3.3	-2.7	-3.6	-3.1	-3.9
SER	0.3	1.1	1.1	0.5	-0.4
THR	-0.5	0.6	0.2	-0.2	-0.7
TRP	-3.8	-3.6	-4.8	-4.8	-6.1
TYR	-3.9	-3.3	-5.1	-3.3	-4.3
VAL	-1.5	-1.2	-1.4	-1.0	-1.1

Extending Gap

- Creation of a gap is much more rare than extension of a gap
- Based on evolution of biological sequences

- Indel Penalty
- Indel Extension Penalty

- How do you change the implementation?

CEITEC

@CEITEC_Brno