Genome 540: Discussion Section Class - 12

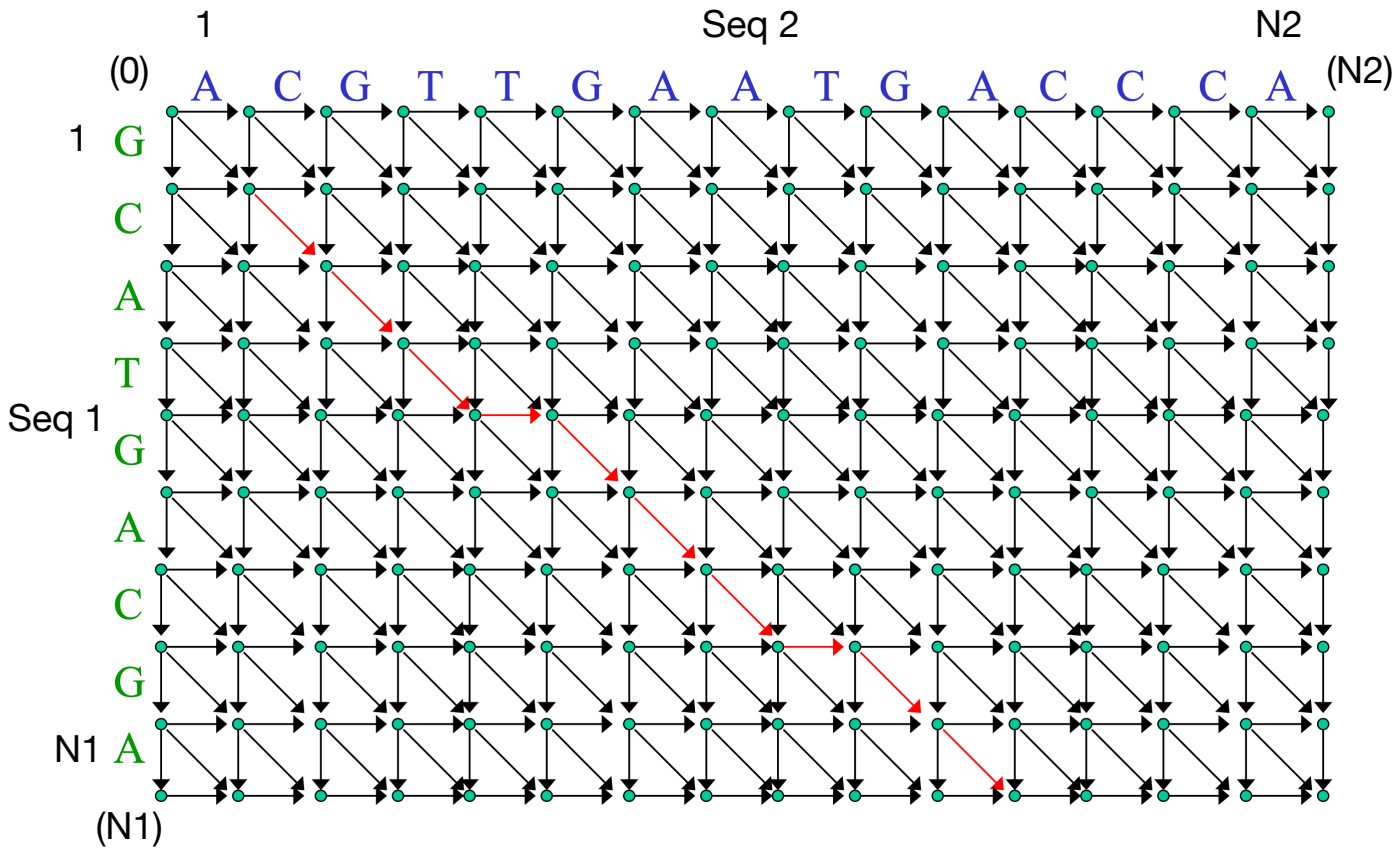
Chengxiang Qiu

HW 5: find multiple alignment for three sequences

- Create an edit graph for 3 sequences using the BLOSUM62 score matrix
- Run HW4 WDAG program on the edit graph to find the highest scoring path (local alignment)
- Report *in the specified format*:
  - Maximum path score for the multiple alignment
  - List of all edge weights (alphabetically sorted)
  - List of all edge counts (alphabetically sorted)
  - Highest scoring alignment

If we only align two sequences

Sequence 1: from 1 to N1
Sequence 2: from 1 to N2



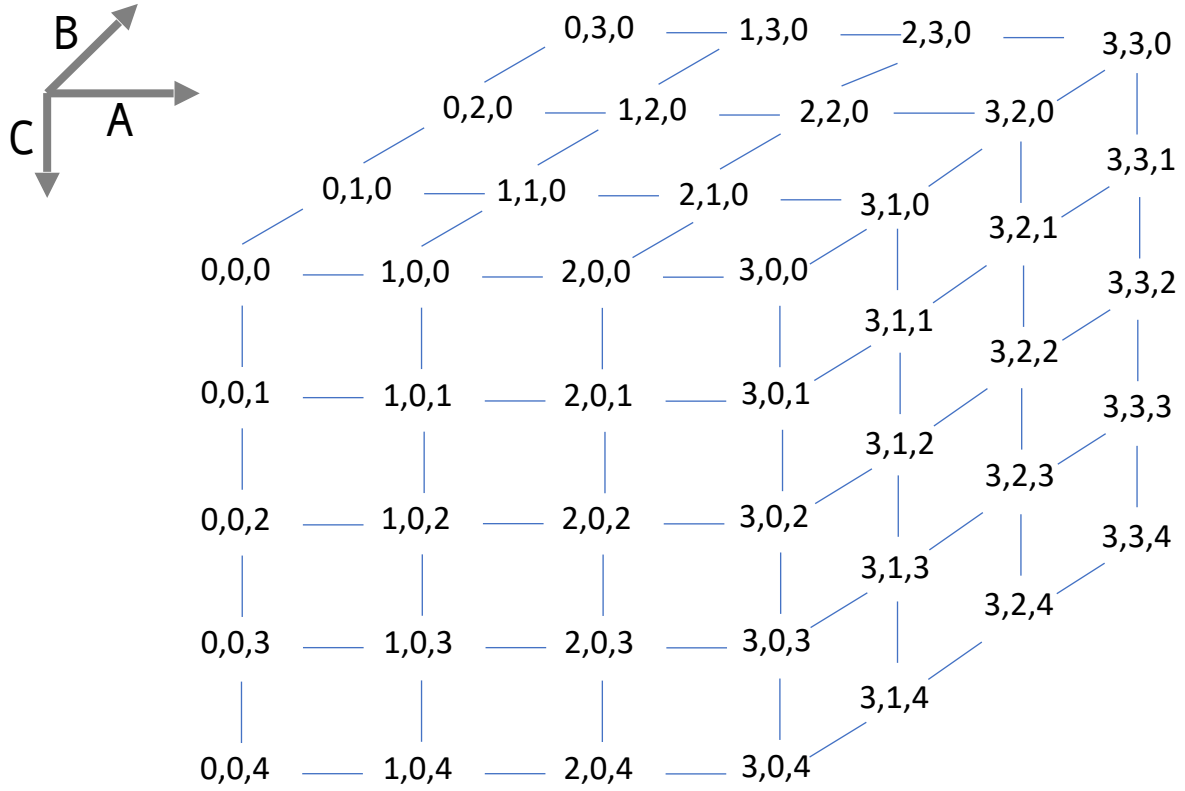
Vertice: (0,0) (0,1) (0,2) ⋯ (0,N2)
(1,0) (1,1)
(2,0) ⋱
⋯
(N1,0) (N1,N2)

Vertice: two *for* loops

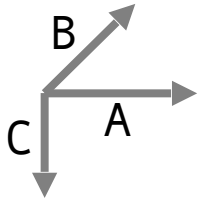
Edges: (0,0) (0,1) weight (_A)
(0,0) (1,0) weight (G_)
(0,0) (1,1) weight (GA)
…

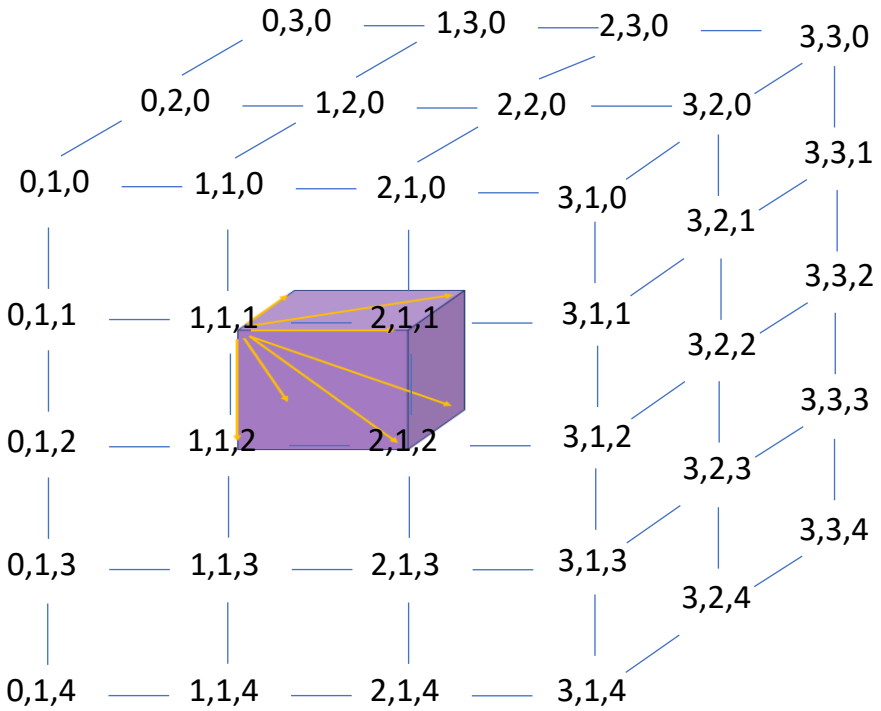
Edges: for any node (i, j)
(i, j) -> (i+1, j)
(i, j) -> (i, j+1)
(i, j) -> (i+1, j+1)

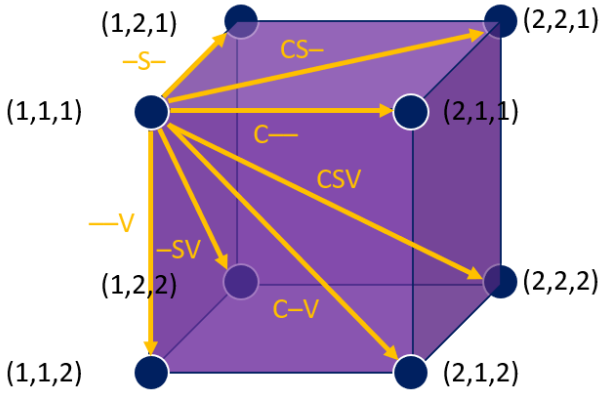
We could extend it to three dimensions (or even multiple dimensions)



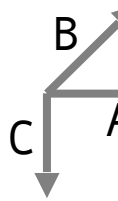

```
A = •M•C•D•
B = •M•S•D•
C = •M•V•D•R•
```

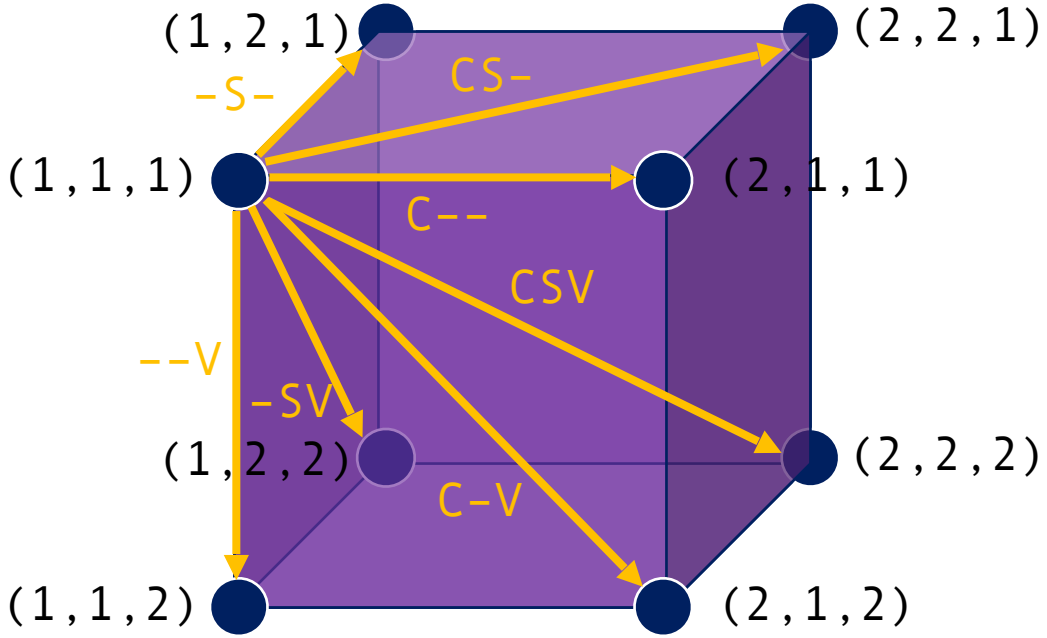






A = •M•C•D•
B = •M•S•D•
C = •M•V•D•R•





	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

A = •M•C•D•R•...
B = •M•S•D•E•...
C = •M•V•D•R•...

weight(CSV) = score(CS) + score(CV) + score(SV)

weight(CS-) = score(CS) + gap_penalty + gap_penalty

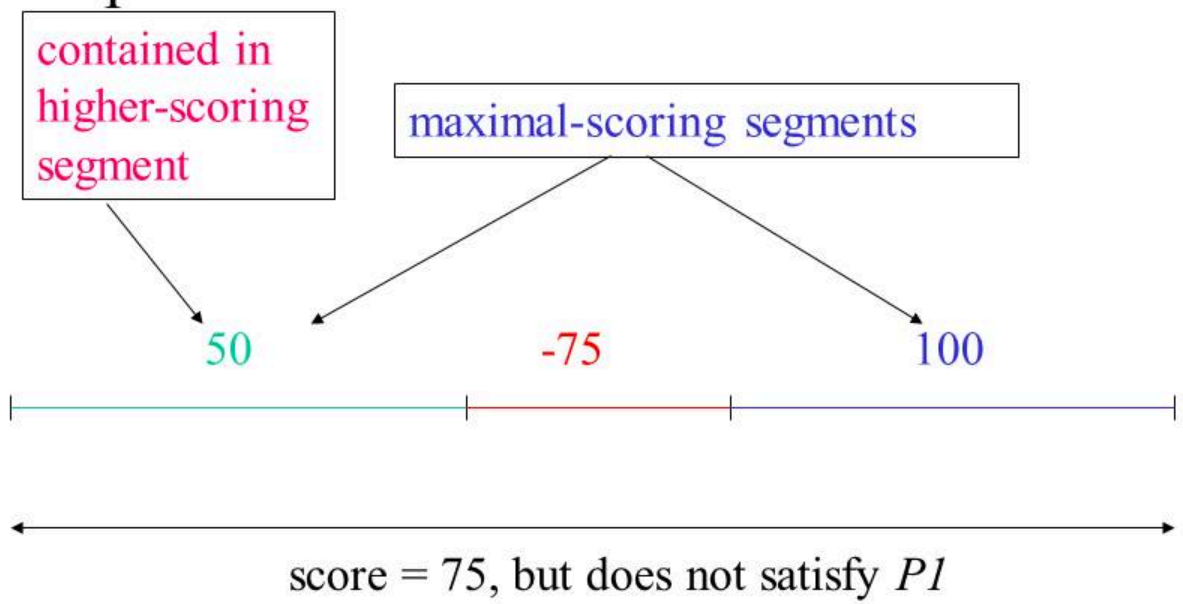
HW6

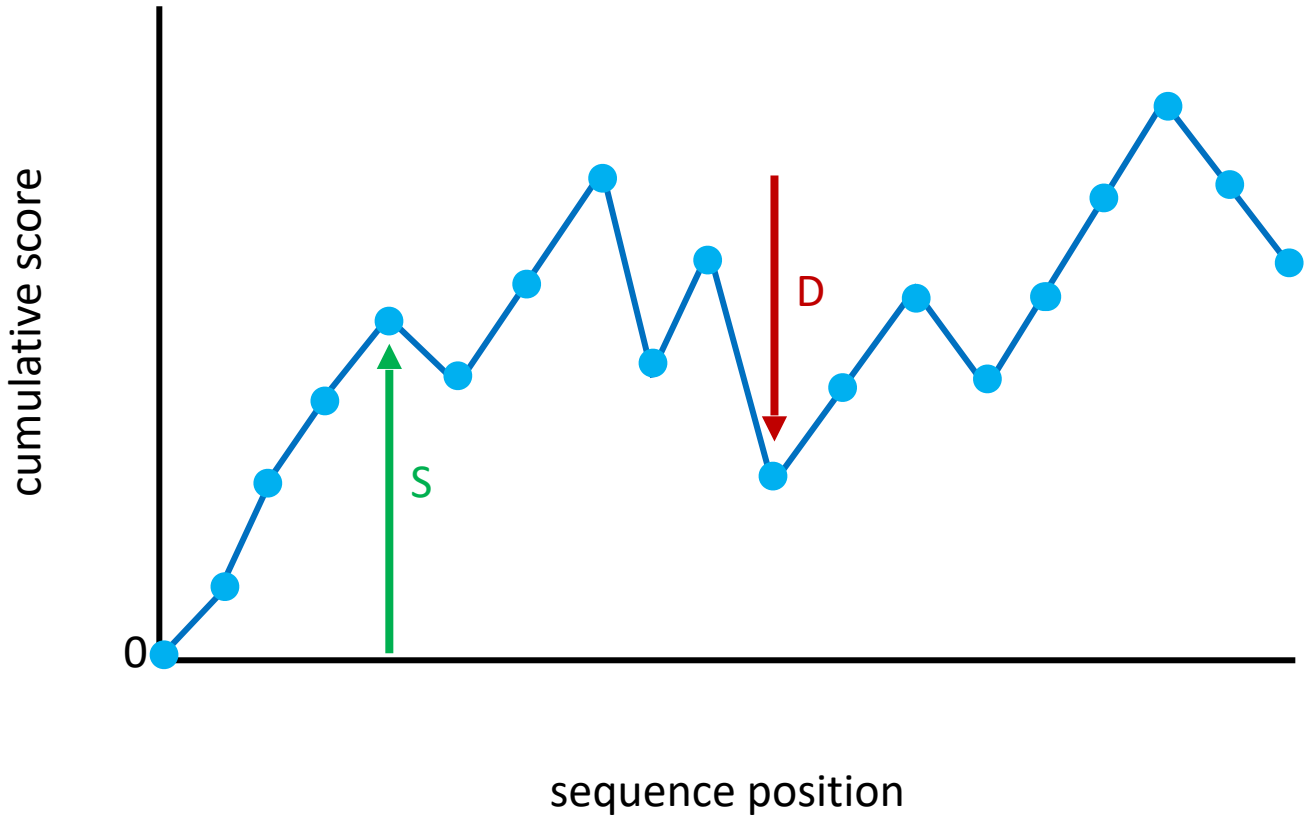
- Due 11:59pm on Sunday, Feb 20
- Assignment: use D-segment algorithm to identify sequence segments with high copy number.
  - Input:
    - File with read start counts at each position along a chromosome (Chromosome\tPosition\tScore)
    - Scoring scheme
  - Output:
    - Number of normal and elevated copy-number segments
    - List of elevated copy-number segments (start, end, score)
    - Annotations for the three segments with the highest scores (look up using UCSC genome browser)
    - Histograms of read-start counts (i.e. number of positions with 0, 1, 2, and >=3 read-starts) for non-elevated and elevated segments

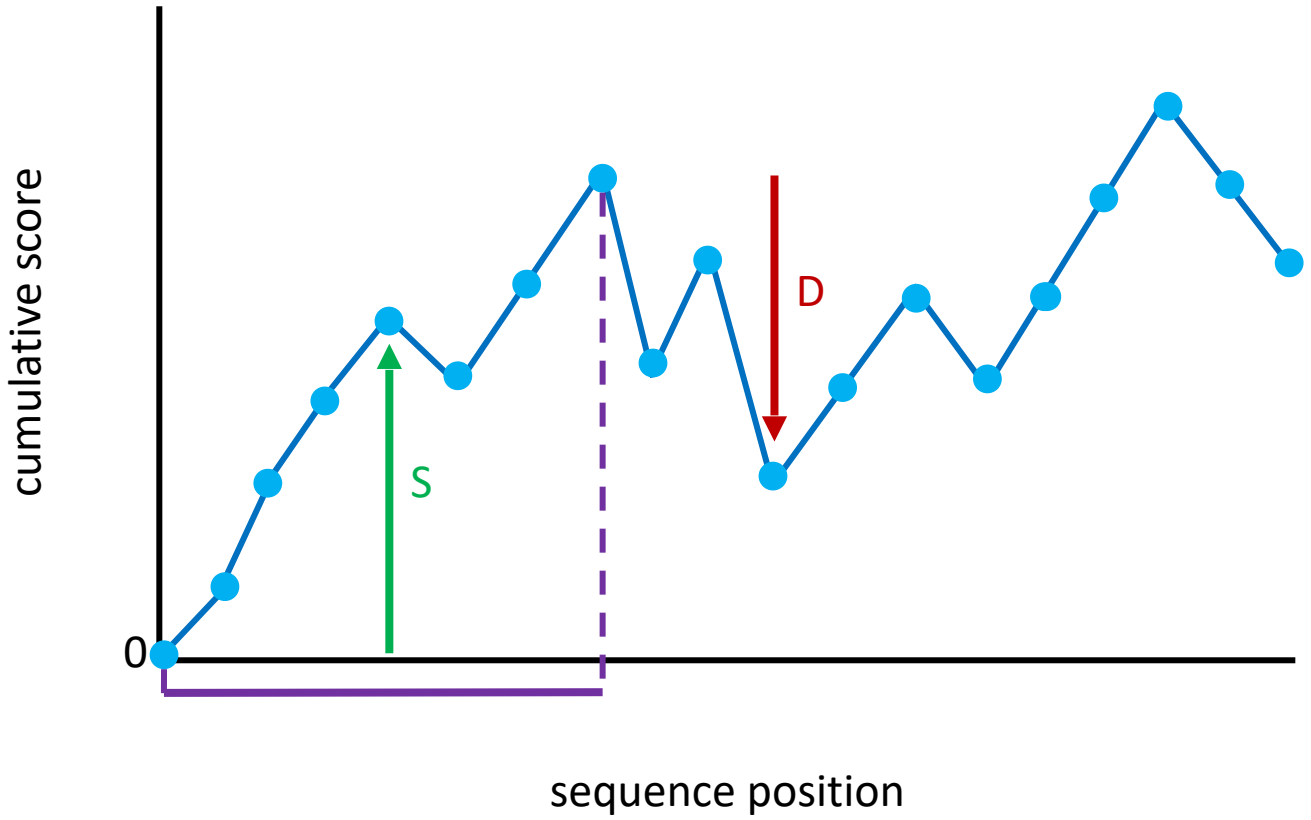
Maximal segment vs. Maximal D-segment

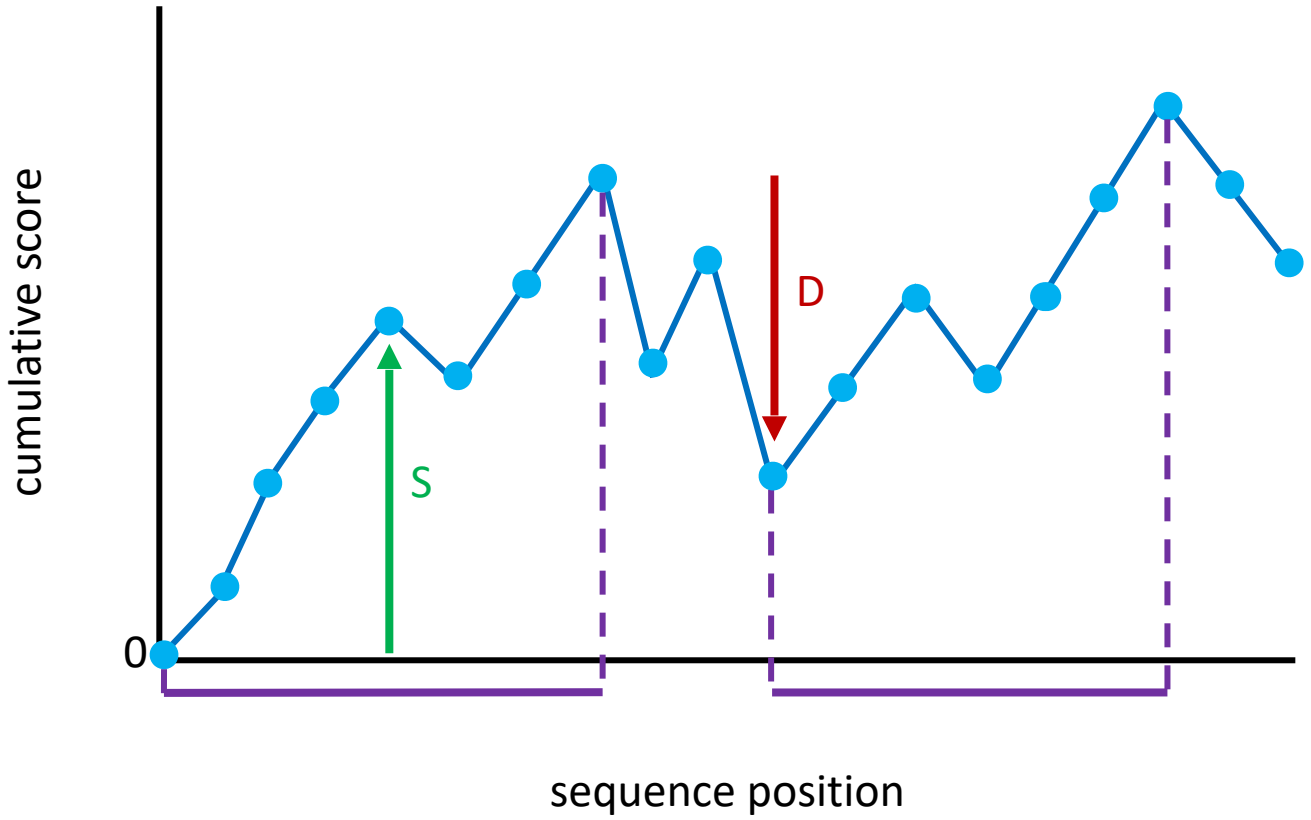
- Maximal segment:
  - No subsegment has a higher score
  - No segment properly containing the segment satisfies the above condition
- Maximal D-segment:
  - No subsegment has score < D, where D is the dropoff value
  - No subsegment has a higher score
  - No D-segment properly containing the D-segment satisfies the above condition
  - The segment score must be >= S, where S >= -D

- A *maximal(-scoring) segment* I is one such that
  - *P1:* no subsegment of I has a higher score than I
  - *P2:* no segment properly containing I satisfies *P1*
- Example:




cumulative score
0
S
D
sequence position


cumulative score
0
S
D
sequence position


cumulative score
0
S
D
sequence position

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

```
D = -3
S = 3
max = 0
start = 1
end = 1
cumul = 0
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

```
D = -3
S = 3
max = 0
start = 2
end = 2
cumul = 0
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

```
D = -3
S = 3
max = 0
start = 3
end = 3
cumul = 0
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

D = -3

S = 3

max = 0

start = 4

end = 4

cumul = 0

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

```
D = -3
S = 3
max = 0.52
start = 5
end = 5
cumul = 0.52
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

```
D = -3
S = 3
max = 1.62
start = 5
end = 6
cumul = 1.62
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

D = -3

S = 3

max = 1.62

start = 5

end = 6

cumul = 1.12

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

```
D = -3
S = 3
max = 2.82
start = 5
end = 8
cumul = 2.82
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

```
D = -3
S = 3
max = 3.34
start = 5
end = 9
cumul = 3.34
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

```
D = -3
S = 3
max = 4.44
start = 5
end = 10
cumul = 4.44
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

D = -3

S = 3

max = 4.44

start = 5

end = 10

cumul = 3.94

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

```
D = -3
S = 3
max = 4.44
start = 5
end = 10
cumul = 3.44
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

D = -3

S = 3

max = 4.44

start = 5

end = 10

cumul = 2.94

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

D = -3

S = 3

max = 4.44

start = 5

end = 10

cumul = 2.44

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# read starts	0	0	0	0	1	2	0	4	1	2	0	0	0	0
score	-0.5	-0.5	-0.5	-0.5	0.52	1.1	-0.5	1.7	0.52	1.1	-0.5	-0.5	-0.5	-0.5

D-segment: 5, 10, 4.44
(start, end, max)

```
D = -3
S = 3
max = 4.44
start = 5
end = 10
cumul = 2.44
```

```
if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
    if (max ≥ S)
        {print start, end, max; }
    max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
      NEEDED! */
}
```

Pseudo-code for the D-segment algorithm:

```
cumul = max = 0; start = 1;
for (i = 1; i ≤ N; i++) {
    cumul += s[i];
    if (cumul ≥ max)
        {max = cum; end = i;}
    if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
        if (max ≥ S)
            {print start, end, max; }
        max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
          NEEDED! */
    }
}
```