Organizacja

Zajęcia mają charakter projektowy. Państwa zadaniem jest implementacja algorytmu Nussinov. Poniżej znajduje się opis. Na przyszłych zajęciach oprócz zwykłych laboratoriów, część czasu zostanie poświęcona na konsultacje i omówienie postępów. **Ostateczny termin na oddanie projektu to 11 kwietnia**.

Aktualizacja: Wynikowy program ma dla zadanej sekwencji wypisać wypełnioną macierz oraz sparowane nukleotydy. Przykład:

- Wejście:

```
GGGAAAUCC
```

- Wyjście:

```
[0 0 0 0 0 0 1 2 3]
[0 0 0 0 0 0 1 2 3]
[0 0 0 0 0 0 1 2 2]
[0 0 0 0 0 0 1 1 1]
[0 0 0 0 0 0 1 1 1]
[0 0 0 0 0 0 1 1 1]
[0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0]
2 9
3 8
6 7
```

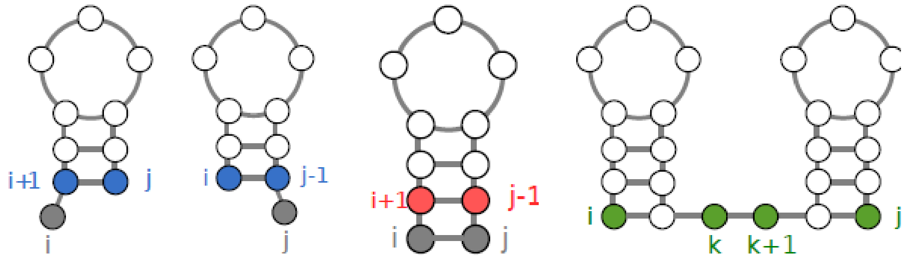
Wprowadzenie

Algorytm Ruth Nussinov:

- Poszukuje optymalnej struktury drugorzędowej zawierającej maksymalną liczbę sparowanych nukleotydów.
- Dozwolone są struktury zawierające motyw spinki do włosów (złożone co najmniej z 3 niesparowanych nukleotydów w pętli).
- Dozwolone są struktury zawierające izolowane pary zasad.
- Algorytm tworzy i wykorzystuje macierz dwuwymiarową $M_{n \times n}$ (n - długość sekwencji).
- Macierz wypełnia się porównując sekwencję z samą sobą.
- Wykorzystywane są dwie zmienne pomocnicze: i - indeks wiersza, j - indeks kolumny.
- W komórce *M[i, j]* umieszczana jest wartość optymalnej struktury *s(i, j)* między nukleotydami *i*-tym oraz *j*-tym. Rozpatrywane są następujące przypadki:
  1. Dodaj parę *(i, j)* do najlepszej struktury *s(i+1, j-1)*,
  2. Dodaj niesparowaną resztę *i* do najlepszej struktury *s(i+1, j)*,

3. Dodaj niesparowaną resztę *j* do najlepszej struktury *s(i, j-1)*,
4. Połącz dwie optymalne podstruktury *s(i, k)* oraz *s(k+1, j)* - bifurkacja.

- example: maximal number of basepairs $\Rightarrow$ **Nussinov**
- consider subsequence from position i to j $\Rightarrow$ $N_{i,j}$
- cases:



- recursion: $N_{i,j} = \max \begin{cases} N_{i+1,j} & //i \text{ unpaired} \\ N_{i,j-1} & //j \text{ unpaired} \\ N_{i+1,j-1} + 1 & //(i,j) \text{ are possible a pair} \\ \max\limits_{k}\{N_{i,k} + N_{k+1,j}\} & //\text{Bifurkation} \end{cases}$

Przykład

1. *M[i, i] = M[i, i+1] = 0*

Table 1: Sekwencja GGGAAAUCC (*n* = 9)

$M_{n\times n}$	**G**	**G**	**G**	**A**	**A**	**A**	**U**	**C**	**C**
G	0								
G	0	0							
G		0	0						
A			0	0					
A				0	0				
A					0	0			
U						0	0		
C							0	0	
C								0	0

2. Wypełniona macierz

Table 2: Sekwencja GGGAAAUCC (*n* = 9)

$M_{n\times n}$	**G**	**G**	**G**	**A**	**A**	**A**	**U**	**C**	**C**
G	0	0	0	0	0	0	1	2	3

... continued on next page

Table 2: Sekwencja GGGAAAUCC (n = 9) (... continued)

$M_{n\times n}$	**G**	**G**	**G**	**A**	**A**	**A**	**U**	**C**	**C**
G	0	0	0	0	0	0	1	2	3
G		0	0	0	0	0	1	2	2
A			0	0	0	0	1	1	1
A				0	0	0	1	1	1
A					0	0	1	1	1
U						0	0	0	0
C							0	0	0
C								0	0

3. Procedura **traceback** (uzyskania wyniku z wypełnionej macierzy):

```
push(1, n) onto stack
repeat until stack is empty:
    pop(i, j)
    if i >= j
        continue;
    else if M[i+1, j] == M[i, j]
        // pkt 2
        push (i+1, j)
    else if M[i, j-1] == M[i, j]
        // pkt 3
        push (i, j-1)
    else if M[i+1, j-1] + 1 == M[i, j]
        // pkt 1
        record i, j base pair
        push (i+1,j-1);
    else for k=i+1 to j-1
        if M[i, k] + M[k+1, j] == M[i, j]
            // pkt 4
            push (k+1, j).
            push (i, k).
            break
```