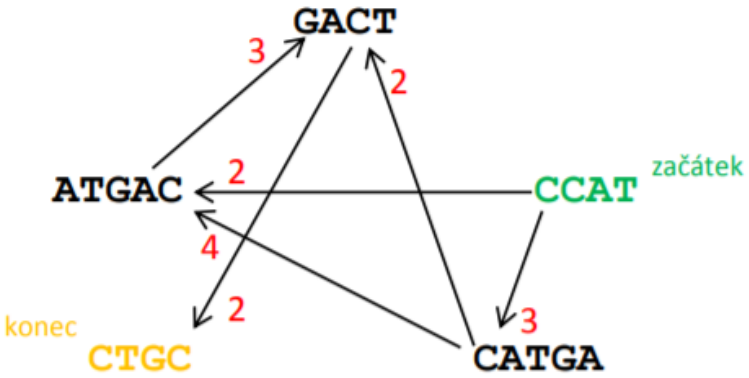
7. Počítačové cvičení

De novo skládání genomu

OLC (Hamiltonský graf)

Příklad 1: soubor *readů* {GACT, ATGAC, CCAT, CTGC, CATGA}
Ready tvoří vrcholy grafu a spojíme je orientovanými úsečkami, které znázorňují překryvy. Délka překryvu je znázorněna číslem u šipky.



Nalezneme potenciální začátky (více výstupních šipek než vstupních) a konce (více vstupních šipek než výstupních). Nebo začneme od vrcholu, mezi kterými je nejvyšší překryv. Postupujeme od začátku po šipkách s nejvyšší hodnotou (při *greedy* postupu). Každý vrchol lze použít jen jednou.

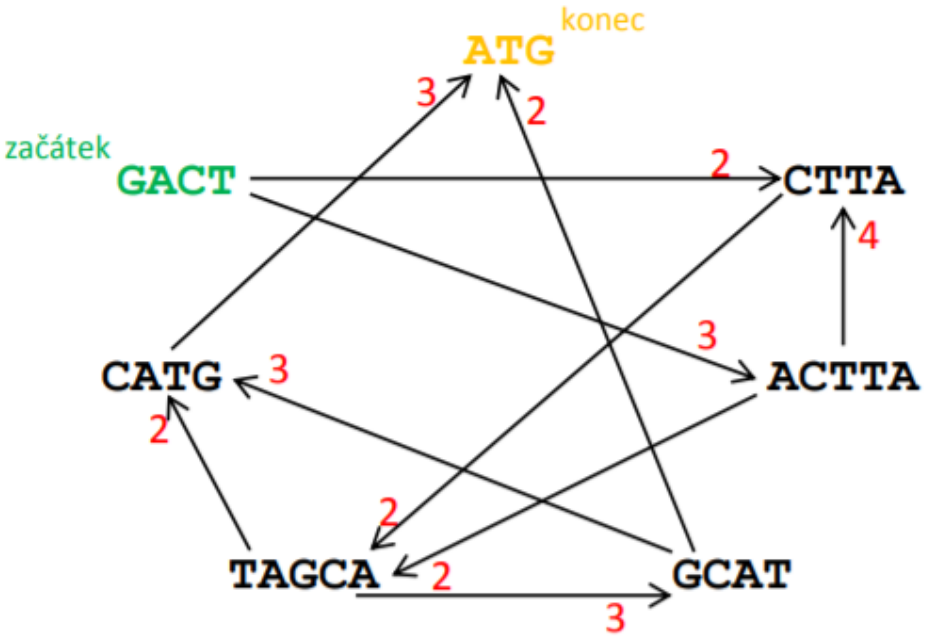
Tvoříme layout z jednotlivých *readů*:

```
CCAT překryv 3 s CATGA    =>        CCAT
                                      CATGA
CATGA překryv 3 s CATGA =>            ATGAC
ATGAC překryv 3 s GACT    =>            GACT
GACT překryv 2 s CTGC      =>              CTGC
```

Vyčerpali jsme všechny vrcholy. Z layoutu vytvoříme konsenzuální sekvenci = ***contig***.
Výsledek je contig **CCATGACTGC**.

Pokud v jednom layoutu nevyčerpáme všechny vrcholy, tvoříme další layout. Výsledné contigy můžeme dále spojovat opět přes překryvy.

Příklad 2: soubor readů {ATG, GACT, CTTA, CATG, ACTTA, TAGCA, GCAT}



Kroky:

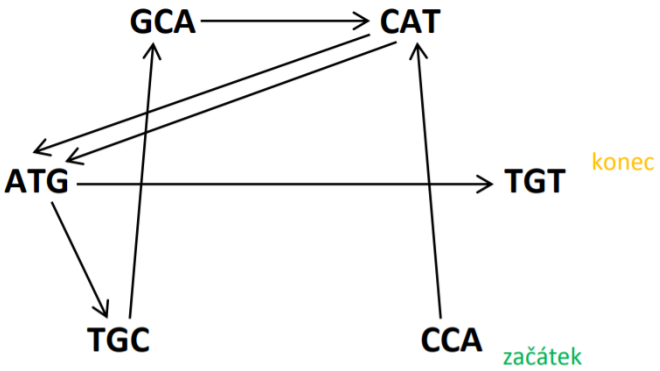
GACT překryv 3 do ACTTA => GACT
ACTTA
ACTTA překryv 4 do CTTA => CTTA
CTTA překryv 2 do TAGCA => TAGCA
TAGCA překryv 3 do GCAT => GCAT
GCAT překryv 3 do CATG => CATG
CATG překryv 3 do ATG => ATG

Výsledek je contig **GACTTAGCATG**.
Alternativa, když začneme od největšího překryvu: contig ACTTAGCATG, zbyl *read* GACT mající s contigem překryv, spojíme na **GACTTAGCATG**.

DBG (deBruijn graf = Eulerovský graf)

Příklad 1: soubor readů {GCAT, CATG, ATGT, CATG, ATGC, TGCA, CCAT}.

Z *readů* (k=4) utvoříme 3-mery (k-1), z množiny odstraníme duplikace: GCA, CAT, ATG, TGT, TGC, CCA k-mery tvoří vrcholy. Pak bereme postupně jednotlivé ready a orientovanou úsečkou spojíme ty vrcholy, které jsou obsaženy v readu (levý 3-mer spojíme s pravým 3-merem).

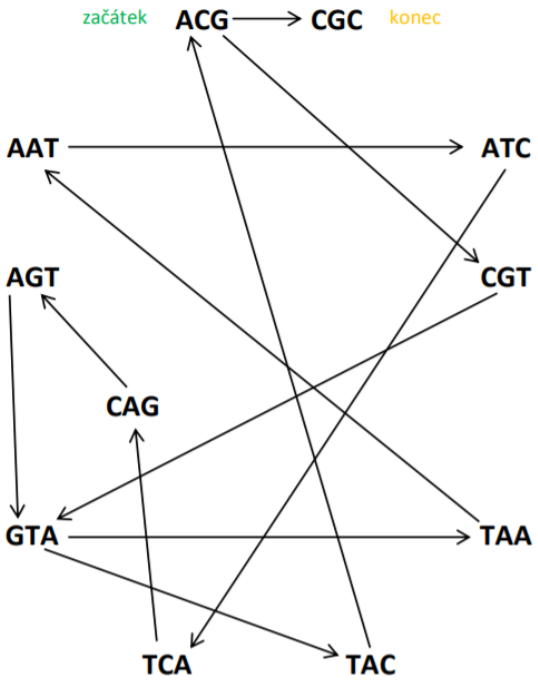


Nalezneme potenciální začátky (více výstupních šipek) a konce (více vstupních šipek). Každou hranu můžeme použít jen jednou, vrchol můžeme projít opakovaně. Při průchodu vrcholem tvoříme contig přidáváním posledního znaku.

Kroky:
z CCA do CAT => CCAT
z CAT do ATG => CCATG
z ATG do TGC => CCATGC
z TGC do GCA => CCATGCA
z GCA do CAT => CCATGCAT
z CAT do ATG => CCATGCATG
z ATG do TGT => CCATGCATGT

Výsledek je contig **CCATGCATGT**.

Příklad 2:
soubor *readů* {ACGT,AATC,ATCA,AGTA,ACGC,CGTA,CAGT,GTAA,GTAC,TAAT,TCAG,TACG}.
soubor 3-merů: ACG, AAT, ATC, AGT, CGT, CAG, CGC, GTA, TAA, TCA, TAC



Kroky:
z ACG do CGT => ACGT
z CGT do GTA => ACGTA
z GTA do TAA => ACGTAA
z TAA do AAT => ACGTAAT
z AAT do ATC => ACGTAATC
z ATC do TCA => ACGTAATCA
z TCA do CAG => ACGTAATCAG
z CAG do AGT => ACGTAATCAGT
z AGT do GTA => ACGTAATCAGTA
z GTA do TAC => ACGTAATCAGTAC
z TAC do ACG => ACGTAATCAGTACG
z ACG do CGC => ACGTAATCAGTACGC

Výsledek je contig **ACGTAATCAGTACGC**

Příklady k řešení:

1. OLC: S = {ATG, GACT, CTTA, CATG, ACTTA, TAGCA, GCAT}

2. DBG: S = {ACGT,AATC,ATCA,AGTA,ACGC,CGTA,CAGT,GTAA,GTAC,TAAT,TCAG,TACG}

3. SSP: S = {s1 = ATC, s2 = TCAGAG, s3 = ATG, s4 = AGCCAT, s5 = TGCAT}

Programovací úkol: – Naprogramovat funkci pro řešení SSP.

Nápověda:
1. Hlavní funkce SSP (která volá funkce **2** a **3**)
2. Funkce pro vytvoření matice překryvů
3. Funkce pro výpočet matice překryvů