PŘEHLED ALGORITMŮ

*Dijkstrův algoritmus*

* **Princip**: Využívá greedy přístup – vždy vybírá vrchol s nejmenší známou vzdáleností. Používá prioritní frontu (nejčastěji implementovanou pomocí haldy) pro efektivní výběr nejbližšího vrcholu.
* **Postup:** 
  1. Inicializace:
     + Zvolíme počáteční vrchol
     + Vytvoříme „mapu vzdáleností“ – Všem ostatním vrcholům přiřadíme hodnotu ∞ (nekonečno) a počátečnímu vrcholu přiřadíme hodnotu 0
  2. Vybereme vrchol s nejmenší známou vzdáleností
  3. Pro každý soused tohoto vrcholu:
     + Spočítáme vzdálenost přes tento vrchol.
     + Pokud je kratší než aktuálně známá vzdálenost k sousedovi -> aktualizujeme ji
  4. Označíme aktuální vrchol jako hotový
  5. Opakujeme kroky 2-4, dokud nejsou všechny vrcholy „hotový“ nebo nemáme žádné dosažitelné vrcholy.
* **Omezení**: nefunguje s negativními hranami, protože předpokládá, že vzdálenosti lze pouze zvyšovat.
* **Časová složitost**:
  + S prioritní frontou (např. binární haldou): O((V + E) · log V)
  + Bez haldy (s lineárním výběrem): O(V²)

*Bellman-Fordův algoritmus*

* **Princip**: opakovaně relaxuje všechny hrany – až V–1krát. Relaxace znamená zkoušení zkrácení vzdáleností mezi vrcholy.
* **Postup**:
  1. Inicializace:
     + Zvolíme počáteční vrchol
     + Vytvoříme „mapu vzdáleností“ – Všem ostatním vrcholům přiřadíme hodnotu ∞ (nekonečno) a počátečnímu 0
  2. Opakujeme následující krok přesně V – 1krát (kde V = počet vrcholů v grafu)
     + Pro každou hranu: „pokud vzdálenost[u] + w < vzdálenost[v], pak aktualizuj vzdálenost[v] = vzdálenost[u] + w“
       - u = počáteční vrchol hrany
       - v = cílový vrchol hrany
       - w = váha (ohodnocení) hrany
  3. Detekce záporných cyklů (volitelný krok navíc)
     + Pro všechny hrany ještě jednou zkontrolujeme: Pokud by šlo znovu zkrátit nějakou vzdálenost, graf obsahuje záporný cyklus → algoritmus selže
* **Výhoda**: funguje i s negativními hranami.
* **Detekce záporných cyklů**: Pokud je po V–1 iteracích stále možné zkrátit některou cestu, znamená to výskyt záporného cyklu.
* **Časová složitost**: O(V · E)

*Floyd-Warshallův algoritmus*

* **Princip**: Používá dynamické programování. Pro všechny trojice vrcholů (i, j, k) kontroluje, zda není cesta z i do j přes ke kratší
* **Postup**:
  + Inicializace
    - Vytvoříme matici vzdáleností D[i][j] o velikosti V x V
      * D[i][j] = 0, pokud i == j (vzdálenost k sama k sobě)
      * D[i][j] = váha hrany, pokud existuje hrana z i do j
      * D[i][j] = ∞, pokud mezi i a j nevede přímá hrana
  + Trojitá iterace přes vrcholy
    - Pro každé k od 1 do V (Zvažovaný prostřední vrchol)
    - Pro každé i od 1 do V (Počáteční vrchol)
    - Pro každé j od 1 do V (Cílový vrchol)
    - Pokud D[i][k] + D[k][j] < D[i][j], pak aktualizuj D[i][j] = D[i][k] + D[k][j]
    - i = výchozí vrchol
    - j = cílový vrchol
    - k = zprostředkující (prostřední) vrchol
    - D[i][j] = nejkratší vzdálenost z i do j
    - D[i][k] + D[k][j] = délka cesty přes vrchol k
  + Detekce záporných cyklů (volitelný krok navíc)
    - Zkontrolujeme hlavní diagonálu matice: pokud pro nějaké i platí D[i][i] < 0. znamená to, že graf obsahuje záporný cyklus → algoritmus selže¨
* **Vhodný pro**: Výpočet nejkratších cest mezi všemi dvojicemi vrcholů
* **Časová složitost**: O(V³)