Řešení úlohy: Zámek obrazovky

Prohledávání grafu a backtracking

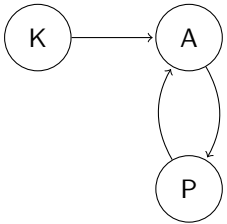
Informatický seminář

12. února 2026

Motivace a zadání

Úloha KSP 31-Z3-2: Zámek obrazovky

- **Kontext:** Odemknutí telefonu gestem.
- **Vstup:**
  - Graf (kruhy s písmeny + čáry).
  - Hledané heslo (slovo).
  - Startovní pozice.
- **Cíl:** Najít posloupnost kruhů, které tvoří dané slovo.
- **Důležité:** Jeden kruh (uzel) můžeme použít **vícekrát**!



Př.: Cesta $K \to A \to P \to A$

Analýza problému

Klíčová otázka

Jedná se o hledání nejkratší cesty (BFS)?

Analýza problému

Klíčová otázka

Jedná se o hledání nejkratší cesty (BFS)?

Ne tak docela.

- U BFS/DFS pro dosažitelnost si značíme `visited`, abychom se necyklili.
- Zde se **musíme** umět vrátit (např. slovo "KAPAK"vyžaduje návštěvu "A"dvakrát).
- Cestu neurčuje jen graf, ale **index v hledaném slově**.

Stav prohledávání:

$$\mathit{Stav} = (\text{aktuální uzel}, \text{index písmene ve slově})$$

Návrh algoritmu: DFS s Backtrackingem

Použijeme prohledávání do hloubky (DFS).

Princip

1. Jsem v uzlu U.
2. Odpovídá písmeno v U aktuálnímu znaku slova?
3. Pokud **ANO**:
   - Je to poslední znak? $\rightarrow$ **MÁME ŘEŠENÍ!**
   - Není? $\rightarrow$ Podívej se na sousedy a rekurzivně hledej další znak.
4. Pokud **NE** nebo od sousedů přišel neúspěch:
   - Vrať se zpět (**Backtracking**).

Pseudokód

```
Funkce Hledej(uzel, index, cesta):
    Pridej uzel do cesty

    POKUD delka(cesta) == delka(SLOVO):
        Vypis cestu -> KONEC

    HledanyZnak = SLOVO[index + 1]

    PRO kazdeho souseda S uzlu uzel:
        POKUD Graf[S].pismeno == HledanyZnak:
            Hledej(S, index + 1, cesta)

    # Backtracking: Tudy cesta nevedla
    Odeber uzel z cesty
```

Listing 1: Logika řešení

Načítání vstupu a reprezentace dat

Specifikum KSP úlohy: ukončovací značka `-1`.

- **Graf:** Seznam slovníků nebo objektů. Index seznamu = číslo uzlu.
- **Uzel:** `{'pismeno': 'X', 'sousede': [1, 2, 5]}`

```python
# Ukázka načítání v Pythonu
graf = []
for i in range(N):
    radek = input().split()
    pismeno = radek[0]
    sousede = []

    for x in radek[1:]:
        val = int(x)
        if val == -1: break  # Ukončovací značka
        sousede.append(val)

    graf.append({'pismeno': pismeno, 'sousede': sousede})
```

Jádro řešení (Python)

```python
def dfs(uzel, index, slovo, graf, cesta):
    cesta.append(uzel) # 1. Tah

    if len(cesta) == len(slovo): # 2. Base case
        print(*cesta)
        return True

    dalsi_znak = slovo[index + 1]

    for soused in graf[uzel]['sousede']: # 3. Rekurze
        if graf[soused]['pismeno'] == dalsi_znak:
            if dfs(soused, index + 1, slovo, graf, cesta):
                return True # Nasli jsme, bublame nahoru

    cesta.pop() # 4. Backtracking
    return False
```

Na co si dát pozor

Chyba: Globální visited

Pokud použijete klasické `visited[uzel] = True`, program selže na slovech typu "ANANAS", kde se musíte vracet do již navštívených uzlů.

Další tipy

- **Limit rekurze:** V Pythonu je defaultní limit 1000. Pro dlouhá slova použijte: `sys.setrecursionlimit(3000)`.
- **Indexování:** Pozor na rozdíl mezi délkou cesty a indexem (index 0 je první znak, délka je 1).

Shrnutí

- Úloha vyžaduje **prohledávání do hloubky (DFS)**.
- Stavový prostor není jen graf, ale **graf + pozice ve slově**.
- Klíčovým prvkem je **Backtracking** (návrat a zkoušení jiné cesty).
- Reprezentace grafu: Seznam sousedů (Adjacency List).

Prostor pro vaše dotazy