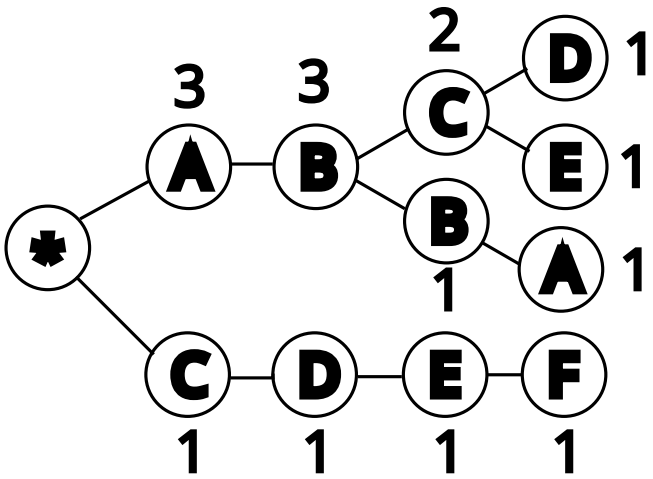
Řešení úlohy č. 2
Mapování

Jednotlivá místa si uložíme do prefixového stromu (trie). Každé místo si uložíme jako cestu stromem, kde název místa přečteme, ve směru od kořene k listům. Pro každý uzel si ještě uložíme, kolika cest (slov) je součástí, to bude jeho hodnota. Pro vstup

```
15
4
abcd
abba
cdef
abce
6
a 1
b 2
c 3
d 4
e 1
c 1
```

bude vypadat trie nějak takto



Unikátní prefix najdeme tak, že od kořene stromu projdeme trii dokud nenarazíme na vrchol, který má hodnotu jedna. To můžeme udělat pomocí standardního breadth-first search algoritmu (BFS, vyhledávání do šířky).

BFS má frontu, ve které je na počátku pouze kořen stromu. Postupně frontu prochází a do fronty přidává ještě nenavštívené sousedy aktuálně navštěvovaného uzle. Když najde nějaký uzel, který má hodnotu jedna, tak si ho zapamatuje, nepřidává jeho sousedy do fronty a pokračuje dál frontou, dokud není prázdná.

Poté vezme každý uzel, s hodnoutou jedna, co jsme si zapamatovali, a najdeme cestu zpátky ke kořeni. To bude unikátní prefix.

Následně můžeme pro každý prefix najít jeho cenu a seřadit si je podle ceny. A pak procházet od nejlevnějšího a vyprintovat všechny, které si můžeme dovolit.

Cenu však nemá prefix, ale písmeno, tuto cenu si můžeme představit jako vzdálenost dvou uzlů v trii. A upravit BFS tak, že místo normální queue použijeme queue prioritní. Priorita je podle vzdálenosti uzle od kořene. (Na podobném principu funguje Dijkstrův algoritmus.) A ve chvíli kdy algoritmus narazí na uzel s hodnotou jedna, tak se ze zásoby tuhy odečte cenu tohoto prefixu. A prefix si zapamatuje. Ve chvíli, kdy je hodnota, kterou má odečíst z banky větší, než kolik toho v bance máme, tak skončí. Tímto najde ty nejlevnější prefixy. A proč?

Pokud by existovala nějaká množina prefixů, která by byla větší než ta, kterou našel algoritmus. Tak musí nutně obsahovat nějaký prefix, který náš algoritmus nenašel. Ale náš algoritmus našel všechny prefixy, které mají cenu menší, než je cena toho nejdražšího přidaného. To znamená, že ten prefix, který nenašel, musí mít cenu vyšší nebo rovna ceně nejdražšího. Ale ten jsme zahodili, protože je moc velký a nemáme na něj dost v bance. Takže žádná taková množina nemůže existovat. □

Jaká je časová složitost těchto algoritmů? BFS má $O(n+m\log(m)+m)$, kde n je celková délka všech názvů míst a m počat míst. Protože přinejhorším se budou místa lišit pouze v posledním znaku. A pak se musí seřadit, což zvládneme v logaritmickém čase pomocí například quicksortu. A ještě musíme všechny místa projít, kde v nejhorším případě přidáme všechny.

Úprava s priority queue má časovou složitost $O(n)$, protože si ušetří řazení a procházení na konci.

$m\log(m)+m$ je v našem případě zanedbatelné oproti n, i přesto je úprava mnohem rychlejší, protože nemusí hledat všechny prefixy, ale jenom ty, které jsou v řešení.