Algoritmy II

Otázky k průběžnému testu znalostí

Revize ze dne 12. února 2020

1

Lineární datové struktury

1. Vysvětlete co znamená, že zásobník představuje paměť typu LIFO.
2. Co je to vrchol zásobníku? Co je to dno zásobníku?
3. Jaká je teoreticky kapacita zásobníku?
4. Co je to přetečení zásobníku? Co je to podtečení zásobníku?
5. Jak se typicky nazývá operace vložení prvku do zásobníku? Popište tuto operaci.
6. Jak se typicky nazývá operace vyjmutí prvku ze zásobníku? Popište tuto operaci.
7. Závisí složitost operací na počtu prvků v zásobníku? Ano či ne?
8. Uveďte příklady použití zásobníku.
9. Do zásobníku vložím prvky v pořadí A, B, C, D. Potom je budu ze zásobníku vyjímat. Jakou dostanu posloupnost?
10. Vysvětlete co znamená, že fronta představuje paměť typu FIFO.
11. Co je to hlava fronty? Co je to ocas fronty?
12. Závisí složitost operací na počtu prvků ve frontě? Ano či ne?
13. Co je to přetečení fronty? Co je to podtečení fronty?
14. Do fronty vložím prvky v pořadí A, B, C, D. Potom je budu z fronty vyjímat. Jakou dostanu posloupnost?
15. Jak se typicky nazývá operace vložení prvku do fronty? Popište tuto operaci.
16. Jak se typicky nazývá operace vyjmutí prvku ze fronty? Popište tuto operaci.
17. Uveďte příklady použití fronty.
18. Popište hlavní problémy při implementaci fronty v poli. Jak je lze řešit pomocí kruhové reprezentace fronty?
19. Jaký je rozdíl mezi frontou, zásobníkem a seznamem? Zaměřte se na operace vkládání a vyjímání prvků z těchto struktur.
20. Co je to hlava seznamu, ocas seznamu?

21. Rozdíl mezi jednosměrným, obousměrným a kruhovým seznamem.
22. Popište algoritmus vyhledávání daného prvku v seznamu?
23. Lze využít algoritmus půlení intervalu pro vyhledávání v seznamu? Jaký je tu hlavní problém?
24. Co je to garbage collector?
25. Co je to iterátor?

2

Dynamická alokace paměti

1. Co je to pointer? Jaký je význam dat obsažených v pointeru? Nápověda: zaměřte se na vysvětlení pojmu nepřímého adresování.
2. Jak deklarujeme pointer `p` na typ `int`?
3. Jak se nazývá pointer ukazující „nikam“?
4. Co je to operátor reference? Uveďte příklad použití.
5. Co je to operátor dereference? Uveďte příklad použití.
6. Předpokládejme, že máme dánu proměnnou `int a = 8;` a pointer `int *p;`. Jakým způsobem zařídíte, aby pointer `p` ukazoval na proměnnou `a`?
7. Předpokládejme, že máme deklarovány dva pointery `int* p;` a `int* q;`. Co se stane po vyhodnocení výrazu `p = q;`?
8. Předpokládejme, že máme deklarovány dva pointery `int* p;` a `int* q;`. Co se stane po vyhodnocení výrazu `*p = *q;`? Jaké zde můžou nastat problémy?
9. Deklarujte pointer `d` na typ `double`. Proveďte alokaci paměti a dealokaci paměti. Jaké operátory na to použijete?
10. Máte dynamicky alokovat pole `a` s prvky typu `int`, které bude mít 20 prvků. Jak to provedete? Pomocí jakého operátoru?
11. Je dáno dynamicky alokované pole `a` s prvky typu `int`, které má 20 prvků. Máte toto pole dealokovat. Jak to provedete? Pomocí jakého operátoru?
12. Máte za úkol dynamicky alokovat dvourozměrné pole `a`, které bude bude mít `R` řádků a `S` sloupců. Prvky budou typu `int`. Jak bude vypadat deklarace pointeru na takové pole? Jak se provede jeho alokace v paměti? Lze to provést jednorázovou alokací?
13. Máte za úkol uvolnit z paměti, dealokovat, dvourozměrné pole `a`, které má `R` řádků a `S` sloupců. Prvky jsou typu `int`. Jak to provedete? Lze to provést jednorázovou dealokací?
14. Máte pointer daný následující deklarací `int* p;`. Co je na tomto pointeru konstantního a co proměnného?
15. Máte pointer daný následující deklarací `const int* p;`. Co je na tomto pointeru konstantního a co proměnného?

16. Máte pointer daný následující deklarací `int* const p;`. Co je na tomto pointeru konstantního a co proměnného?

17. Máte pointer daný následující deklarací `const int* const p;`. Co je na tomto pointeru konstantního a co proměnného?

18. Popište tři způsoby předávání parametru do funkce.

19. Do funkce `F` předáváte pole pomocí pointeru. Ve funkci `F` na takto předané pole (na parametr z hlavičky funkce) aplikujete operátor `sizeof`? Jaký výsledek dostanete?

20. Jaké operace jsou povoleny provádět s pointery v pointerové aritmetice? Za jakého předpokladu mají tyto operace smysl? Nápověda: Mají tyto operace smysl pokud pointery ukazují na libovolné místo v paměti? Nebo musí být s něčím svázány?

21. Jak je možné chápat pole z pohledu pointeru? Co vlastně z tohoto pohledu představuje identifikátor pole?

22. Máte dáno pole `int a[10];` a ukazatel `int* p = a;`. Čemu odpovídá výraz `*(p+3)`?

23. Máte dáno pole `int a[10];` a ukazatel `int* p = a;`. Kam bude ukazovat pointer `p` po vyhodnocení výrazu `p+=2;`?

24. Máte dva pointery `int *p, *q;`. Jaký je smysl výrazu `p == q`?

25. Máte dáno pole `int a[10];` a dva ukazatele `int *p, *q;`. Oba tyto ukazatele ukazují na některý prvek v poli `a`;. Jaký je smysl výrazů `p-q` a `*p-*q`?

3

Spojová implementace lineárních datových struktur

1. Máme danou strukturu a čtyři proměnné:

   ```
   struct MyStruct
   {
      int Id;
      double Value;
   };

   MyStruct a, b;
   MyStruct *p, *q;
   ```

   Co se stane pokud provedu `a = b;`? A co pokud provedu `p = q;`?

2. Jak alokuji dynamicky strukturu z předchozí otázky? Pomocí jakého operátoru?

3. Jak uvolním z paměti dynamicky alokovanou strukturu, kterou jsem vytvořil v předchozí otázce?

4. Mám deklarovanou proměnnou `MyStruct a;`. Jak se dostanu k položce `Value` v proměnné `a`?

5. Mám deklarovanou proměnnou `MyStruct *p;`. Jak se dostanu k položce `Value` v proměnné `p` pomocí jediného operátoru (bez dereference)?

6. Mám deklarovanou proměnnou `MyStruct *p;`. Jak se dostanu k položce `Value` v proměnné `p` pomocí operátoru dereference?

7. Mám následující kód:

   ```
   double *p = new double;
   delete p;
   *p = 15;
   ```

   Co tento kód způsobí?

8. Mám následující kód:

   ```
   double *p = new double;
   p = NULL;
   ```

   Co tento kód způsobí? K čemu v paměti dojde?

9. Mám následující kód:

```
double *p = new double;
p = NULL;
*p = 15;
```

Co tento kód způsobí? K čemu v paměti dojde?

10. Mám seznam implementovaný pomocí pole. Musím tento seznam (především to pole) explicitně uvolňovat z paměti pomocí operátoru `delete`?

11. Mám seznam implementovaný pomocí dynamicky alokovaných struktur. Musím tento seznam explicitně uvolňovat z paměti pomocí operátoru `delete`?

12. Co je to zarážka (angl. sentinel)? Co nahrazuje?

13. K čemu slouží zarážka?

14. Jak se změní algoritmus vkládání prvku na začátek obousměrného seznamu, pokud využijeme zarážku?

15. Jak se změní algoritmus smazání prvku z obousměrného seznamu, pokud využijeme zarážku?

16. Kdy je vhodné využít zarážku? Zaměřte se na spotřebovanou paměť a počet prvků v uvažovaném seznamu.

17. Kdy je vhodné použít seznam implementovaný v poli a kdy pomocí dynamických struktur? Zaměřte se na rychlost operací vložení (s tím spojené alokování položky), operací mazání (s tím spojené dealokovaní položky). Dále se zaměřte na možnou změnu kapacity seznamu.

18. Jestliže budu k seznamu přistupovat pouze pomocí funkcí `Insert`, `Delete` nebo `Search` a změním implementaci seznamu. Musím měnit i okolní kód, kde se seznam využívá?

19. Pokud implementuji frontu pomocí dynamických struktur, potřebuji kruhový buffer a podobné „triky“ s polem?

20. Zkuste navrhnout efektivní algoritmus pro třídění seznamu. Uvědomte si, že k položkám seznamu můžeme přistupovat pouze sekvenčně, jako například k datům v souboru. Jaký Vám známý algoritmus pro třídění lze na tento problém aplikovat?

21. Zkuste navrhnout rekurzívní algoritmus pro vyhledání prvku v obousměrném seznamu.

22. Jak spočítáte počet položek v seznamu? Má na tento algoritmus vliv zda je seznam obousměrný nebo jednosměrný?

23. Jak byste v obousměrném seznamu implementovali mazání k-té položky.

24. Popište algoritmus mazání položky z obousměrného seznamu. Položka určená ke smazání se musí nejprve vyhledat podle dat uložených v položce.

25. Popište algoritmus vkládání na začátek obousměrného seznamu.

4

Grafy

1. Definujte pojem neorientovaný graf.
2. Čím se liší orientovaný graf od neorientovaného?
3. Co je to stupeň uzlu?
4. Co je to sled?
5. Co je to cesta?
6. Co je to délka cesty?
7. Co je to uzavřený sled, uzavřená cesta? Jaké je alternativní pojmenování uzavřené cesty?
8. Co znamená, že graf je acyklický?
9. Co znamená, že graf je souvislý?
10. Jak zní definice volného stromu?
11. Kolik existuje mezi dvěma vrcholy volného stromu cest?
12. Kolik hran musíme minimálně přidat do volného stromu, abychom dostali graf s kružnicí?
13. Kolik hran musíme z volného stromu nejméně odebrat, abychom dostali nesouvislý graf?
14. Má-li volný strom $|V|$ vrcholů, kolik má hran?
15. Co je to kořenový strom?
16. Vysvětlete pojem předchůdce uzlu.
17. Vysvětlete pojem následovník uzlu.
18. Vysvětlete pojem rodič uzlu.
19. Vysvětlete pojem potomek uzlu.
20. Vysvětlete pojem sourozenec uzlu.
21. Vysvětlete pojem list.

22. Vysvětlete pojem vnitřní uzel.
23. Jak se liší definice stupně uzlu pro obecný graf a kořenový strom?
24. Co je to hloubka uzlu.
25. Jak je definována výška stromu?
26. Co je to seřazený strom?
27. Lze v seřazeném stromu u uzlu s jediným potomkem rozlišit, jestli je levý nebo pravý, první či druhý?

5

Algoritmy průchodu grafem

1. Matice sousednosti grafu. Jaký je řád této matice? Je čtvercová? Jaký je význam prvků v matici?
2. Matice sousednosti grafu. Jak poznám, že graf je neorientovaný? Jak poznám, že graf neobsahuje smyčky?
3. Reprezentace grafu pomocí seznamu sousedních vrcholů. Jaké jsou výhody této implementace oproti matici sousednosti? Zaměřte se na operace jako je vložení nového vrcholu do grafu či paměťovou složitost obou reprezentací grafu.
4. Co je cílem algoritmů průchodu grafem?
5. Jaké tři algoritmy průchodu grafem znáte?
6. Popište obecný algoritmus značkování vrcholů.
7. V čem spočívá nevýhoda obecného značkovacího algoritmu? Co v jeho implementaci může zvyšovat časovou složitost?
8. Popište myšlenku algoritmu průchodu grafem do šířky (BFS).
9. V algoritmu BFS se používají pro označení vrcholů tři barvy – bílá, šedá a černá. Jaký význam mají tato tři označení?
10. V jaké datové struktuře je při průchodu BFS udržována množina šedých vrcholů?
11. Jaký význam mají hodnoty $d[u]$ a $\pi[u]$ z algoritmu BFS?
12. Jak detekuji konec BFS algoritmu? Jak poznám, že jsem skončil?
13. Výstupem BFS je tzv. BF-strom. Jaké vrcholy tento strom obsahuje?
14. Mám dva vrcholy u a v v BF-stromu. Jaká je délka cesty mezi těmito dvěma vrcholy ze všech možných cest mezi těmito vrcholy v původním grafu? Nejkratší? Nejdelší? Průměrná?
15. Co je to průměr grafu?
16. Jaká je složitost BFS algoritmu?
17. Popište myšlenku algoritmu průchodu grafem do hloubky (DFS).
18. DFS algoritmus přiřazuje každému vrcholu dvě tzv. časová razítka. Jaký mají význam?

19. Jak mohu, pomocí časových razítek, určit pro dva vrcholy vztah rodič-potomek v DF-stromu?
20. Jaká je složitost DFS algoritmu?
21. DFS lze implementovat i nerekurzivně? Pomocí jaké datové struktury mohu odstranit rekurzi?
22. Co je to kostra grafu?
23. Co je to dopředná a zpětná hrana?
24. Co je to topologické třídění?
25. Jak mohu detekovat v grafu cyklus?

6

Binární vyhledávací stromy I

1. Jak zní rekurzivní definice binárního stromu?
2. Je možné levého a pravého potomka daného uzlu v binárním stromu zaměňovat?
3. Co je to úplný binární strom?
4. Předpokládejme, že máme dán binární strom. Co musí platit pro klíče ve všech uzlech binárního stromu, abychom jej mohli považovat za binární *vyhledávací* strom?
5. Popište rekurzivní algoritmus vyhledání klíče x v binárním vyhledávacím stromu.
6. Z jaké vlastnosti (obecně) stromu plyne fakt, že jsme schopni v binárním vyhledávacím stromu nalézt všechny klíče, které do něj byly vloženy? Jinak řečeno, že lze navštívit všechny uzly ve stromu.
7. Z jaké vlastnosti (obecně) stromu plyne fakt, že si nemusíme značkovat již navštívené uzly stromu? Například u průchodu grafem jsem si museli uzly označovat.
8. Popište nerekurzivní variantu vyhledání klíče x v binárním vyhledávacím stromu.
9. Popište algoritmus jak lze v binárním vyhledávacím stromu najít minimum mezi všemi v něm uloženými klíči.
10. Popište algoritmus jak lze v binárním vyhledávacím stromu najít maximum mezi všemi v něm uloženými klíči.
11. Popište algoritmus vložení klíče x do binárního vyhledávacího stromu.
12. Jakým způsobem souvisí algoritmus pro vkládání a vyhledávání v binárním vyhledávacím stromu?
13. Jaká je nejhorší podoba binárního vyhledávacího stromu? V jakou jinou datovou strukturu strom v tomto případě zdegeneruje?
14. Jak vypadá posloupnost klíčů, jejímž vložením vznikne degenerovaný binární vyhledávací strom?
15. Lze se nějak bránit vzniku degenerovaného binárního vyhledávacího stromu?
16. Máme dáno n různých klíčů. Vložením těchto klíčů do binárního vyhledávacího stromu vznikne degenerovaný binární vyhledávací strom. Jaká je výška takto vzniklého degenerovaného binárního vyhledávacího stromu?

17. Máme dáno n různých klíčů. Tyto klíče vložíme do binárního vyhledávacího stromu, který není degenerovaný. Jaká je, v nejlepším případě, výška takto vzniklého binárního vyhledávacího stromu?

18. Předpokládejme, že máme binární vyhledávací strom s n uzly, který není degenerovaný, a hledáme klíč k. Kolik uzlů musíme v průměru projít, abychom zjistili, zda klíč k v binárním vyhledávacím stromu je nebo není?

19. Předpokládejme, že máme binární vyhledávací strom s n uzly, který není degenerovaný, a vkládáme klíč k. Kolik uzlů musíme v průměru projít, než můžeme klíč k do stromu vložit?

20. Popište rekurzivní algoritmus, který zjistí počet uzlů ve stromu. Nápověda: zamyslete se, kolik mají uzlů stromy zmiňované v bodech 1 a 2 v definici binárního stromu.

7

Binární vyhledávací stromy II

1. Máme za úkol odebrat klíč ze stromu, čili zrušit odpovídající uzel v binárním vyhledávacím stromu. Jak postupujeme, jestliže rušený uzel nemá žádné potomky.
2. Máme za úkol odebrat klíč ze stromu, čili zrušit odpovídající uzel v binárním vyhledávacím stromu. Jak postupujeme, jestliže rušený uzel má jednoho potomka.
3. V binárním vyhledávacím stromu je dán klíč k. Jakým způsobem najdeme, či kde se v tomto stromu nachází nejbližší menší klíč než k?
4. V binárním vyhledávacím stromu je dán klíč k. Jakým způsobem najdeme, či kde se v tomto stromu nachází nejbližší větší klíč než k?
5. Máme za úkol odebrat klíč ze stromu, čili zrušit odpovídající uzel v binárním vyhledávacím stromu. Jak postupujeme, jestliže rušený uzel má dva potomky.
6. Co je to průchod binárním stromem?
7. Lze na binární strom aplikovat obecné algoritmy průchodu grafem? Proč?
8. Jakou datovou strukturu byste použili pro průchod stromem po „patrech“?
9. Popište přímý průchod binárním stromem (anglicky preorder).
10. Popište vnitřní průchod binárním stromem (anglicky inorder).
11. Popište zpětný průchod binárním stromem (anglicky postorder).
12. Uveďte příklady využití jednotlivých způsobů průchodu binárním stromem.
13. Uvažujme libovolný způsob průchodu stromem (preorder, inorder, postorder). Z jaké vlastnosti binárního stromu plyne fakt, že si nemusíme uzly ve stromu značkovat (černá, šedá a bílá barva), abychom věděli, který uzel jsme již navštívili a přitom algoritmus skončí, tj. neuvázne v nekonečném cyklu?
14. Uvažujme libovolný způsob průchodu stromem (preorder, inorder, postorder). Z jaké vlastnosti binárního stromu plyne fakt, že si nemusíme uzly ve stromu značkovat (černá, šedá a bílá barva), abychom věděli, který uzel jsme již navštívili a přitom žádný uzel nevynecháme?
15. Popište implementaci iterátoru v binárním vyhledávacím stromu. Jak se musíme vypořádat s faktem, že na každý uzel ve stromu mohou navazovat další dva potomci? Jakou musíme použít pomocnou datovou strukturu?

16. Jak najdeme v iterátoru další prvek pro iteraci? Předpokládejme, že iterujeme od nejmenšího prvku směrem k největšímu.

17. Jak poznáme v iterátoru konec iterace?

18. Výsledkem algoritmu průchodu grafu do hloubky (DFS) je kostra grafu. DFS algoritmus aplikujeme na binární vyhledávací strom. Jak bude vypadat kostra grafu pro tento druh grafu? Nápověda: Kostra grafu bude shodná s binárním stromem? Nebo bude nějaký uzel či hrana chybět?

19. Mohou existovat při DFS průchodu binárním vyhledávacím stromem zpětné hrany? Nebo jen dopředné? Vysvětlete.

8

Vyvážené a vícecestné stromy

1. Jak je definován dokonale vyvážený strom?
2. Jak lze snadno, pro pevně danou množinu klíčů, zkonstruovat dokonale vyvážený strom?
3. Máme danou množinu klíčů. Pro tuto množinu sestrojíme odpovídající dokonale vyvážený strom, AVL strom, binární vyhledávací strom (základní nevyvážená varianta). Který z těchto stromů bude mít minimální výšku. Poznámka: v jistých případech se může výška všech tři stromů shodovat, ale vždy tím způsobem, že výška dvou stromů se shodou okolností bude rovnat výšce třetího.
4. Jak je definováno kritérium vyváženosti AVL stromu?
5. Kdy nazýváme binární vyhledávací strom červeno-černým stromem? Co musí vše takový strom splňovat (jaká je jeho definice)?
6. Dokážete formulovat kritérium vyváženosti červeno-černého stromu?
7. Jak se nazývají operace, které obnovují vyváženosti stromu jeho přetáčením tj. změnami vztahu rodič-potomek?
8. 2-3-4 strom obsahuje tři různé typy uzlů. Kolik obsahují jednotlivé typy uzlů klíčů?
9. 2-3-4 strom obsahuje tři různé typy uzlů. Kolik obsahují jednotlivé typy uzlů ukazatelů na potomky?
10. Co se stane s 4-uzlem jestliže je již zcela zaplněn a musíme do něj vložit další klíč?
11. Předpokládejme, že máme dán B-strom řádu n. Kolik každá stránka obsahuje nejvýše klíčů?
12. Předpokládejme, že máme dán B-strom řádu n. Kolik musí každá stránka obsahovat nejméně klíčů?
13. Předpokládejme, že máme dán B-strom řádu n. Existuje výjimka z minimálního a maximálního počtu klíčů ve stránce?
14. Předpokládejme, že máme dán B-strom řádu n. Jestliže stránka obsahuje m klíčů, kolik má tato stránky potomků?
15. Předpokládejme, že máme dán B-strom řádu n. Jakou hloubku musí mít všechny listy ve stromu?

16. Předpokládejme, že máme dán B-strom řádu n. Dále předpokládejme, že klíče ve stránce ukládáme v poli a klíče v tomto pole udržujeme setříděné. Jaký byste použili efektivní algoritmus pro nalezení klíče k v tomto poli?

17. Předpokládejme, že máme dán B-strom řádu n. Dále máme stránku S (S není kořen stromu), která je zcela zaplněna, tj. obsahuje $2n$ klíčů. Do této stránky chceme přidat další klíč. Popište co se s touto stránkou stane a co se stane s jednotlivými klíči (skupinami klíčů).

18. Předpokládejme, že máme dán B-strom řádu n. A dále máme kořenovou stránku B-stromu, která je zcela zaplněna, tj. obsahuje $2n$ klíčů. Do této stránky chceme přidat další klíč. Popište co se s touto stránkou stane a co se stane s jednotlivými klíči (skupinami klíčů).

19. Předpokládejme, že máme dán B-strom řádu n. Ve stromu je celkem M klíčů. Jaká bude přibližně výška tohoto B-stromu?

20. Je B-strom vyvážený nebo ne?

21. Co je to faktor využití paměti u B-stromu?

22. Předpokládejme, že máme dán B-strom řádu n. Máme stránku, které obsahuje více než n klíčů, jeden z těchto klíčů chceme ze stránky odebrat. Jak to provedete?

23. Předpokládejme, že máme dán B-strom řádu n. Máme stránku, které obsahuje přesně n klíčů, jeden z těchto klíčů chceme ze stránky odebrat. Jak to provedete?

24. Může B-strom snížit svoji výšku? Kdy k tomu dojde?

25. Může B-strom zvýšit svoji výšku? Kdy k tomu dojde?

9

Hašování

1. Co je to univerzum klíčů?
2. Jaký bývá typický poměr velikostí mezi univerzem klíčů a množinou skutečně zpracovávaných klíčů, tj. těmi, které jsou skutečně uloženy v tabulce?
3. Co je to kolize v hašovací tabulce?
4. Lze kolizím zcela zabránit?
5. Jakým principem řeší separátní řetězení kolize?
6. Předpokládejme, že máme dánu hašovací tabulku s celkem m sloty, kolize jsou řešeny separátním řetězením. Popište, jak se do tabulky vloží klíč k.
7. Předpokládejme, že máme dánu hašovací tabulku s celkem m sloty, kolize jsou řešeny separátním řetězením. Popište, jak se v tabulce nalezne klíč k.
8. Předpokládejme, že máme dánu hašovací tabulku s celkem m sloty, kolize jsou řešeny separátním řetězením. Popište, jak se z tabulky odebere klíč k. Je to vůbec možné?
9. Co je to faktor naplnění α? Jak se vypočítá?
10. Předpokládejme, že máme dánu hašovací tabulku s celkem m sloty, kolize jsou řešeny separátním řetězením. V tabulce je uloženo celkem n klíčů. Kolik prvků v tabulce musíme prozkoumat, abychom rozhodli, zda se tam prvek nalézá nebo ne. Jak tento počet souvisí s faktorem naplnění α?
11. Může být u hašovacích tabulek s kolizemi řešenými pomocí separátního řetězení faktor naplnění α větší než jedna, tj. $\alpha > 1$?
12. Může být u hašovací tabulky, kde jsou kolize řešeny otevřeným adresováním faktor naplnění α větší než jedna, tj. $\alpha > 1$?
13. Co je to jednoduché uniformní hašování?
14. Co je to uniformní hašování?
15. Předpokládejme, že máme dánu hašovací tabulku s celkem m sloty, kolize jsou řešeny otevřeným adresováním, metodou lineárních pokusů. Popište, jak se do tabulky vloží klíč k.

16. Předpokládejme, že máme dánu hašovací tabulku s celkem m sloty, kolize jsou řešeny otevřeným adresováním, metodou lineárních pokusů. Popište, jak se v tabulce nalezne klíč k.

17. Předpokládejme, že máme dánu hašovací tabulku s celkem m sloty, kolize jsou řešeny otevřeným adresováním, metodou lineárních pokusů. Popište, jak se z tabulky odebere klíč k. Je to vůbec možné?

18. Popište princip řešení kolizí pomocí lineárních pokusů u hašovací tabulky s otevřeným adresováním.

19. Popište princip řešení kolizí pomocí kvadratických pokusů u hašovací tabulky s otevřeným adresováním.

20. Popište princip řešení kolizí pomocí dvojitého hašování u hašovací tabulky s otevřeným adresováním.

21. Co je to primární shlukování?

22. Co je to sekundární shlukování?

23. Předpokládejme, že máme dánu hašovací tabulku s celkem m sloty, kolize jsou řešeny některou z metod otevřeného adresování. Kolik pokusů můžu maximálně provést při hledání klíče k?

24. Předpokládejme, že máme dánu hašovací tabulku s celkem m sloty. Kolize jsou řešeny separátním řetězením. Dále předpokládejme, že hašovací hodnota klíče k je $h(k)$. Který slot (sloty) musím při hledání klíče k v tabulce prohledat:

    (a) všechny,

    (b) jen s indexem $h(k)$,

    (c) slot s indexem $h(k)$ a sousední sloty s indexy $h(k)-1$ a $h(k)+1$.

25. Jaká čísla jsou vhodnými kandidáty na velikost hašovací tabulky?

10

Vyhledávání v textu

1. Definujte problém vyhledání vzorku v textu. Kde se lze s těmito problémy setkat?
2. Co je to předzpracování textu a vzorku?
3. Uveďte algoritmus, který pro svůj běh nepotřebuje předzpracovat ani text ani vzorek.
4. Uveďte příklad algoritmu (algoritmů), který předzpracovává vzorek ale nepředzpracovává prohledávaný text.
5. Uveďte příklad vyhledávací metody, která nepředzpracovává vzorek ale předzpracovává prohledávaný text.
6. Uveďte příklad vyhledávací metody, která předzpracovává vzorek i prohledávaný text.
7. Co je to abeceda? Označme ji například Σ.
8. Co je to řetězec nad abecedou Σ? Jak je definována délka řetězce?
9. Máte dánu abecedu Σ a přirozená čísla m a n. Jak budou vypadat řetězce a^m a $a^m b^n$, kde $a, b \in \Sigma$?
10. Co je to prázdný řetězec? Jak jej značíme? Jaká je jeho délka?
11. Máte dánu abecedu Σ. Co označují množiny Σ^+ a Σ^*?
12. Co je to předpona (prefix) řetězce w?
13. Co je to přípona (sufix) řetězce w?
14. Co je to podřetězec (faktor) řetězce w?
15. Co je to hranice řetězce w?
16. Popište princip elementárního algoritmu (BruteForce Algorithm).
17. Jaká je časová složitost elementárního algoritmu? Jaké operace nás zajímají u výpočtu složitosti vyhledávacích algoritmů?
18. Jaká je očekávaná časová složitost elementárního algoritmu pro text a vzorek z přirozených jazyků?

19. Elementární algoritmus provádí jednu činnost se vzorkem velice neefektivně. Tato neefektivita mu znemožňuje dosahovat menší časové složitosti při vyhledávání. O jako činnost je vzorkem jde a jak tuto neefektivitu řeší ostatní algoritmy? Například Knuth-Morris-Prattův?

20. Popište princip Shift-Or algoritmu. Jaké dvě bitové operace se tu používají?

21. Jaká je časová složitost Shift-Or algoritmu?

22. Popište princip Karp-Rabinova algoritmu. K čemu slouží u tohoto algoritmu hašovací funkce (Pozn. Vzpomeňte si na kapitolu o hašování a pohled na hašovací funkci jako na „negativní filtr“.)?

23. Co je to sousměrné a protisměrné vyhledávání?

24. Popište princip Boyer-Mooreova algoritmu. Soustřeďte se na využití protisměrného vyhledávání pro efektivní detekci (ne)výskytu vzorku v textu.

25. Jaká je složitost Boyer-Mooreova algoritmu v nejhorším případě a jaká v nejlepším? Pro jakou kombinaci vzorku a textu dojde k nejhoršímu případu?