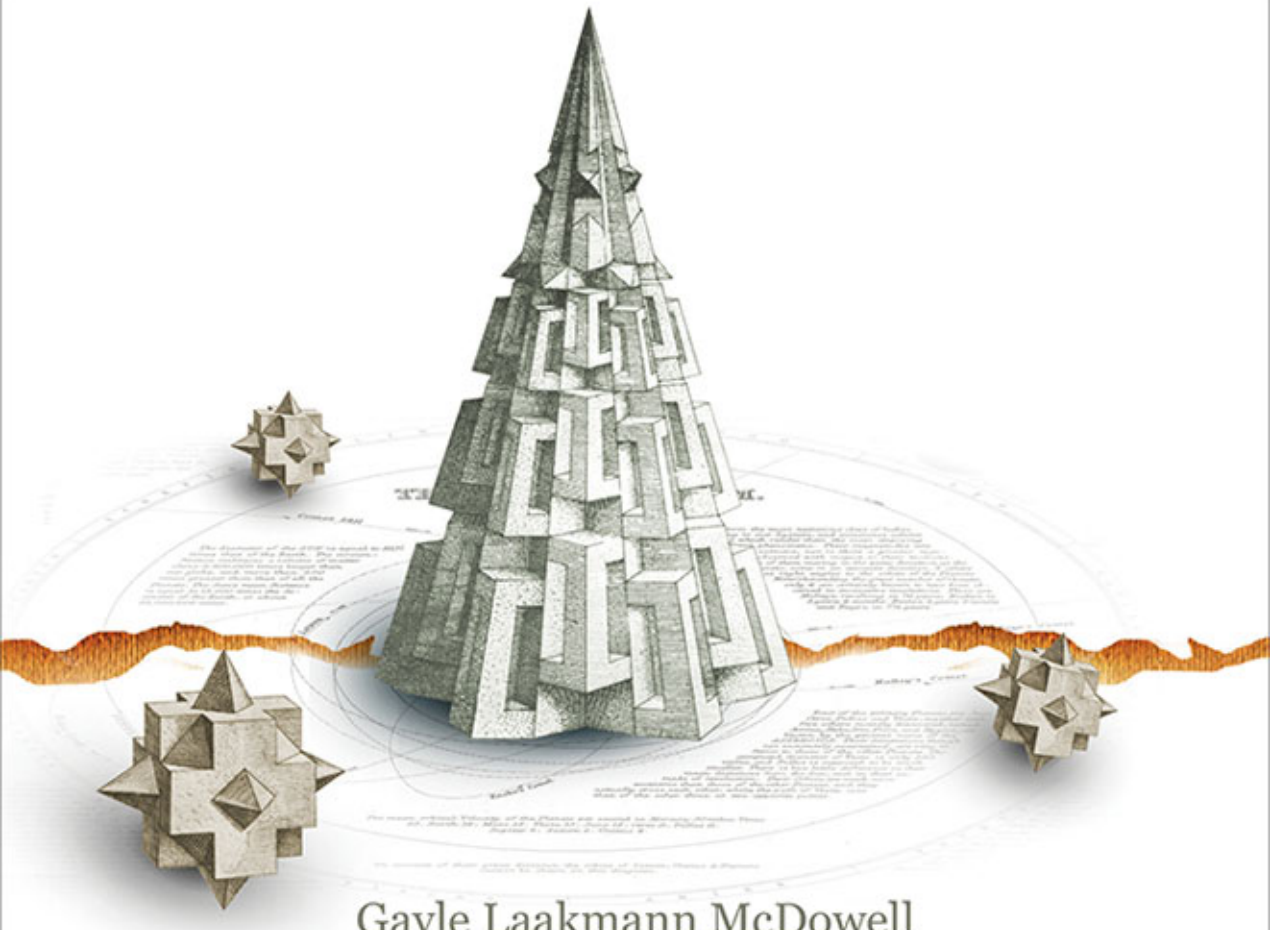
ROZMOWA REKRUTACYJNA DLA PROGRAMISTÓW

Przewodnik do sukcesu



Gayle Laakmann McDowell

Helion

Tytuł oryginału: Cracking the Coding Interview: 150 Programming Questions and Solutions

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-246-9332-0

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/rorepr.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/rorepr
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

1

Tablice i łańcuchy znaków

Mam nadzieję, że wszyscy czytelnicy tej książki potrafią używać tablic i łańcuchów znaków, dlatego nie będę zanudzać was tymi zagadnieniami. Zamiast tego skoncentruję się na często używanych technikach i kwestiach związanych z tymi strukturami danych.

Zwróć uwagę na to, że w zadaniach tablice często można zastąpić łańcuchami znaków i na odwrót. Oznacza to, że jeśli w książce w pytaniu występuje tablica, na rozmowie zamiast niej mogą pojawić się łańcuchy znaków i vice versa.

Tablice z haszowaniem

Tablica z haszowaniem to struktura danych, w której kluczom przyporządkowane są wartości, co pozwala na bardzo wydajne wyszukiwanie elementów. W bardzo prostej implementacji tablic z haszowaniem używane są tablica i *funkcja haszująca*. Gdy chcesz wstawić obiekt i odpowiadający mu klucz, funkcja haszująca przekształca klucz na liczbę całkowitą, która określa indeks w tablicy. Następnie obiekt jest zapisywany za pomocą tego indeksu.

Jednak zwykle rozwiązanie to nie działa poprawnie. W tej implementacji wartość skrótu dla każdego klucza musi być niepowtarzalna — w przeciwnym razie dane mogą zostać przypadkowo nadpisane. Aby zapobiec tego typu „kolizjom", tablica musi być bardzo duża (jej wielkość powinna odpowiadać liczbie wszystkich możliwych kluczy).

Zamiast tworzyć bardzo dużą tablicę i zapisywać obiekty za pomocą indeksu `funkcja_haszujaca ↪(klucz)`, można zbudować znacznie mniejszą tablicę i przechowywać obiekty na liście powiązanej za pomocą indeksu `funkcja_haszujaca(klucz) % dlugosc_tablicy`. W celu pobrania obiektu o określonym kluczu należy wyszukać ten klucz na odpowiedniej liście.

Inna możliwość to zaimplementowanie tablicy z haszowaniem za pomocą binarnego drzewa poszukiwań. Pozwala to zagwarantować czas wyszukiwania na poziomie O(log n), ponieważ drzewo może pozostawać zrównoważone. Dodatkowo pozwala to zmniejszyć zużycie pamięci, ponieważ nie trzeba alokować dużej tablicy na początku pracy programu.

Zalecam, aby przed rozmową przećwiczyć implementowanie i używanie tablic z haszowaniem. Jest to jedna ze struktur danych najczęściej używanych w trakcie rozmów rekrutacyjnych. Jest niemal pewne, że w trakcie jednej z rozmów natrafisz na tę strukturę.

Poniżej znajduje się prosty przykładowy kod w Javie ilustrujący używanie tablicy z haszowaniem.

```
1  public HashMap<Integer, Student> buildMap(Student[] students) {
2    HashMap<Integer, Student> map = new HashMap<Integer, Student>();
3    for (Student s : students) map.put(s.getId(), s);
4    return map;
5  }
```

Zauważ, że choć czasem rekruter bezpośrednio prosi o zastosowanie tablicy z haszowaniem, to częściej musisz sam wywnioskować, że powinieneś wykorzystać tę strukturę do rozwiązania problemu.

Lista tablicowa (dynamicznie rozszerzana tablica)

Lista tablicowa (inaczej typ `ArrayList` lub dynamicznie rozszerzana tablica) jest tablicą, która sama zmienia rozmiar, a jednocześnie zapewnia dostęp w czasie O(1). Typowa implementacja działa tak, że gdy tablica jest pełna, jej długość zostaje podwojona. Za każdym razem wymaga to O(n) czasu, jednak operacja ta jest potrzebna na tyle rzadko, że złożoność i tak wynosi O(1).

```
1  public ArrayList<String> merge(String[] words, String[] more) {
2    ArrayList<String> sentence = new ArrayList<String>();
3    for (String w : words) sentence.add(w);
4    for (String w : more) sentence.add(w);
5    return sentence;
6  }
```

Bufory łańcuchów znaków (typ StringBuffer)

Załóżmy, że chcesz scalić listę łańcuchów znaków w pokazany poniżej sposób. Jaka jest złożoność tego kodu? Dla uproszczenia przyjmij, że wszystkie łańcuchy znaków mają tę samą długość (x) i jest `n` takich łańcuchów.

```
1  public String joinWords(String[] words) {
2    String sentence = "";
3    for (String w : words) {
4      sentence = sentence + w;
5    }
6    return sentence;
7  }
```

Przy każdym scalaniu tworzona jest nowa kopia łańcucha znaków i dwa łańcuchy są kopiowane znak po znaku. Pierwsza iteracja wymaga skopiowania x znaków. W drugiej iteracji kopiowanych jest 2x znaków, w trzeciej — 3x znaków itd. Tak więc łącznie złożoność wynosi $O(x + 2x + \ldots + nx)$, czyli $O(xn^2)$. Dlaczego nie jest to $O(xn^n)$? Ponieważ $1 + 2 + \ldots + n$ wynosi $n(n+1)/2$, czyli $O(n^2)$.

Typ `StringBuffer` pozwala uniknąć tego problemu. Tworzy on tablicę wszystkich łańcuchów znaków i kopiuje dane z tablicy z powrotem do łańcucha tylko wtedy, gdy jest to konieczne.

```
1  public String joinWords(String[] words) {
2    StringBuffer sentence = new StringBuffer();
3    for (String w : words) {
4      sentence.append(w);
5    }
6    return sentence.toString();
7  }
```

Dobrym ćwiczeniem z zakresu łańcuchów znaków, tablic i ogólnych struktur danych jest zaimplementowanie własnej wersji typu `StringBuffer`.

Pytania z rozmów rekrutacyjnych

1.1. Zaimplementuj algorytm określający, czy łańcuch zawiera tylko niepowtarzające się znaki. Jak wykonasz to zadanie, jeśli nie można stosować dodatkowych struktur danych?

s. 178

1.2. Zaimplementuj w języku C lub C++ funkcję `void reverse(char* str)`, która odwraca znaki w łańcuchu zakończonym znakiem `null`.

s. 179

1.3. Napisz metodę, która przyjmuje dwa łańcuchy znaków i określa, czy jeden jest permutacją drugiego.

s. 179

1.4. Napisz metodę, która zastępuje wszystkie spacje w łańcuchu znaków sekwencją `%20`. Przyjmij, że na końcu łańcucha dostępne jest miejsce na dodatkowe znaki, a także że znasz prawdziwą długość łańcucha. Jeśli implementujesz rozwiązanie w Javie, zastosuj tablicę znaków, aby móc wykonać operację w miejscu (bez kopiowania łańcuchów).

PRZYKŁAD:

Dane wejściowe: `"Pan Jan Nowak    ", 13`

Dane wyjściowe: `"Pan%20Jan%20Nowak"`

s. 181

1.5. Zaimplementuj metodę, która przeprowadza prostą kompresję łańcuchów znaków opartą na zliczaniu powtarzających się liter. Na przykład metoda ma przekształcać łańcuch `aabcccccaaa` na `a2b1c5a3`. Jeśli „skompresowany" łańcuch znaków nie jest mniejszy od wyjściowego, metoda powinna zwracać pierwotny łańcuch.

s. 182

1.6. Dany jest rysunek reprezentowany przez macierz o wymiarach `N×N`, w którym każdy piksel jest reprezentowany za pomocą czterech bajtów. Napisz metodę, która rotuje rysunek o 90 stopni. Czy potrafisz wykonać tę operację w miejscu?

s. 184

1.7. Napisz algorytm, który jeśli jakiś element macierzy o wymiarach `M×N` ma wartość zero, ustawia cały wiersz i kolumnę tego elementu na 0.

s. 186

1.8. Załóżmy, że istnieje metoda `isSubstring`, która sprawdza, czy dane słowo jest podłańcuchem innego. Napisz kod, który przyjmuje dwa łańcuchy znaków (`s1` i `s2`) oraz za pomocą tylko jednego wywołania metody `isSubstring` sprawdza, czy `s2` jest rotacją słowa `s1` (na przykład `lajkonik` jest rotacją słowa `niklajko`).

s. 187

Dodatkowe zadania: Manipulowanie bitami (5.7), Projektowanie obiektowe (8.10), Rekurencja (9.3), Sortowanie i wyszukiwanie (11.6), C++ (13.10), Umiarkowanie trudne (17.7, 17.8, 17.14).

Skorowidz

E

F

G

I

J

K

L

Ł

M

N

O

P

R

S

T

U

W

Z

TWOJA PRZEPUSTKA DO NOWEJ FIRMY!

stresuje każdego, kto stara się o pracę. W przypadku naboru na stanowisko programisty składa się ona z kilku etapów, a w jednym z nich poruszane są tematy techniczne. Daje to rekruterom duże pole do popisu. Szczegółowe pytania o znane technologie, sprawdzian z wiedzy ogólnej, łamigłówki, rekurencja — to tylko część zagadnień, o które możesz zostać zapytany. Pytania z zakresu rachunku prawdopodobieństwa czy struktur danych również nie powinny Cię zdziwić. Co możesz zrobić, by dobrze wypaść? Solidnie przygotuj się do rozmowy i nie daj się zaskoczyć!

Mnóstwo cennych wskazówek znajdziesz w tej fantastycznej książce. Sięgnij po nią i poznaj typowy proces rekrutacji, dziesięć najczęściej popełnianych błędów oraz ulubione pytania rekruterów. W kolejnych rozdziałach znajdziesz opisy rekrutacji w takich firmach jak Google, Microsoft, Yahoo! czy Facebook. Te opisy wraz z najlepszymi sposobami na przygotowanie się do rozmowy rekrutacyjnej oraz przykładowymi zadaniami technicznymi (z rozwiązaniami) pomogą Ci opanować stres i zrozumieć tok myślenia osób rekrutujących. Książka ta jest obowiązkową lekturą dla wszystkich programistów szukających pracy!

Dzięki tej książce:

- **poznasz dogłębnie proces rekrutacji,**
- **zobaczysz, jak prowadzą ją giganci rynku IT,**
- **solidnie przygotujesz się do rozmowy rekrutacyjnej,**
- **unikniesz typowych błędów,**
- **zdobędziesz pracę w wybranej firmie!.**

helion.pl
księgarnia internetowa
Nr katalogowy: 23566
Księgarnia internetowa: http://helion.pl
Zamówienia telefoniczne: 0 801 339900 0 601 339900
Informatyka w najlepszym wydaniu

Helion
Sprawdź najnowsze promocje: http://helion.pl/promocje
Książki najchętniej czytane: http://helion.pl/bestsellery
Zamów informacje o nowościach: http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-246-9332-0
cena: 79,00 zł