Sortowanie przez kopcowanie

Łukasz Konieczny | LK4 | Lab 7

1. Czym jest kopiec binarny

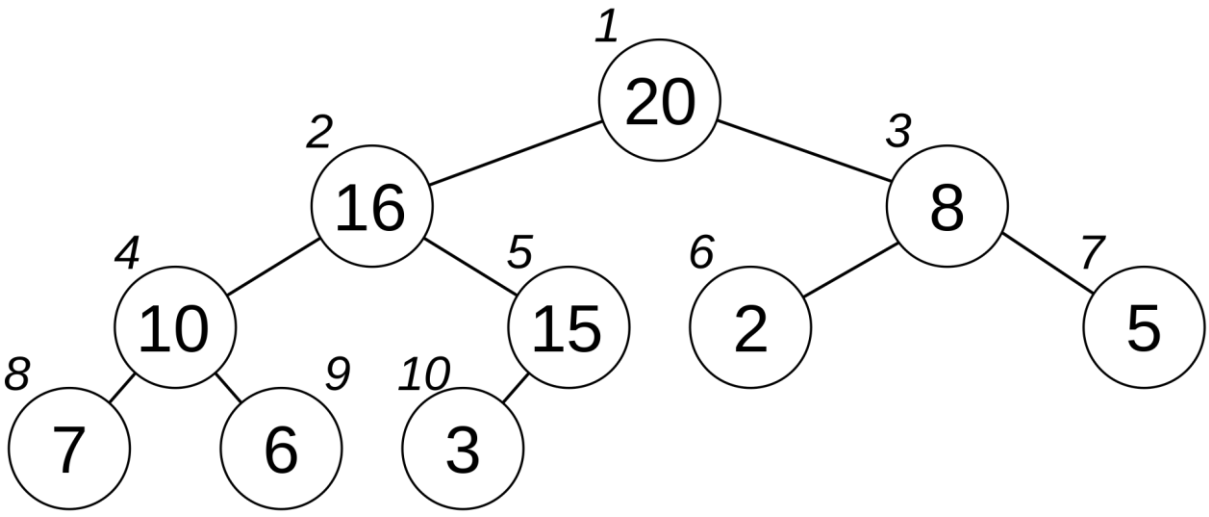
Kopiec binarny to specjalny rodzaj drzewa binarnego, które cechuje się następującymi warunkami:

- Wszystkie poziomy oprócz ostatniego muszą być pełne
- Jeśli ostatni poziom NIE jest pełny, elementy (liście) są ułożone od lewej do prawej, tzn. tylko z prawej strony może brakować elementów
- Elementy kopca nazywamy liśćmi / synami / dziećmi, a elementy od których wychodzą ich ojcami / rodzicami. Pierwszy element, od którego wychodzą wszystkie inne to „korzeń”.
- Kopce dzielą się na dwa typy, MAX oraz MIN. W tym pierwszym, każdy „syn” musi mieć wartość mniejszą lub równą od swojego „rodzica”. W drugim, każdy „rodzic” musi mieć wartość mniejszą lub równą od swojego „syna”.

2. Reprezentacja kopca binarnego

Najpopularniejszą formą reprezentacji kopca binarnego jest graf, w przystępny sposób pokazuje on relacje pomiędzy poszczególnymi elementami.

Oto przykładowy graf reprezentujący kopiec binarny typu MAX. Liczby w kółkach reprezentują wartości, a liczby obok nich indeksy liści:



W informatyce, kopiec binarny możemy zaprezentować w postaci tablicy jednowymiarowej:

Indeks	0	1	2	3	4	5	6	7	8	9
Wartość	20	16	8	10	15	2	5	7	6	3

(Tabela reprezentuje ten sam kopiec co graf, jednak rozpoczynam w niej indeksy od 0 a nie od 1, z uwagi na numerowanie indeksów w językach programowania)

W przypadku takiej reprezentacji, elementy są połączone następującymi zależnościami:

Indeks lewego syna: 2i + 1

Indeks prawego syna: 2i + 2

Indeks rodzica: (i – 1) / 2 // Dla i > 0

W języku C++ kopiec możemy przedstawić także jako strukturę (lub klasę), w której każdy element posiada oprócz wartości wskaźniki do swojego rodzica oraz synów.

3. Operacje na kopcach

Definiujemy następujące operacje na kopcach:

- Budowa kopca
- Dodawanie elementu
- Usuwanie elementu
- Przywracanie własności kopca

Wszystkie operacje przedstawię na bazie kopca typu MAX, dla kopca typu MIN wystarczy zmienić odpowiednio warunek porównania rodzic-syn.

Budowa kopca odbywa się na jeden z dwóch sposobów:

- Top – Down:
  W tej metodzie każdy z elementów po kolei dodajemy do kopca przy pomocy algorytmu, który opiszę w punkcie niżej.
- Bottom – Up:
  Konstruujemy drzewo binarne ze wszystkich elementów. Następnie, patrzymy na ostatni element który jest w tym drzewie rodzicem. Sprawdzamy, czy spełniona jest dla niego własność kopca. Jeśli nie, odpowiednio zamieniamy elementy. Następnie aż do końca kopca wykonujemy to samo porównanie dla wszystkich innych elementów które są rodzicami.

Dodawanie elementu do kopca:

- Wstawiamy element na sam dół kopca
- Porównujemy jego wartość z wartością rodzica: jeśli jest większy od rodzica, zamieniamy ich miejscami
- Kontynuujemy zamienianie, do momentu natrafienia na większego lub równego rodzica, albo dotarcia do końca kopca

Usuwanie elementu kopca:

- Zamień miejscami element który chcesz usunąć, z ostatnim elementem
- Usuń ostatni element
- Po takiej zamianie kopiec prawdopodobnie nie spełnia już założeń, przywróć własność kopca.

Przywracanie własności kopca:

- Odbywa się ono w sposób opisany w punkcie o dodawaniu elementu, rozpoczynając od ostatniego elementu kopca wykonujemy odpowiednie porównania do momentu, gdy warunki kopca nie będą spełnione.

4. Sortowanie przez kopcowanie

Sortowanie to wykorzystuje własność kopca MAX, zgodnie z którą na szczycie kopca znajduje się zawsze największy element.

Aby wykonać to sortowanie nie potrzebujemy definiować struktur z których składa się kopiec ani korzystać z tablic pomocniczych.
Wystarczy odpowiednio przywracać własność kopca dla elementów tablicy:

- Na początku budujemy kopiec ze wszystkich elementów tablicy o największym indeksie „n”
- Na indeksie „0” otrzymujemy największy element. Zamieniamy go miejscami z elementem na indeksie „n”
- Przywracamy własność kopca dla elementów od „0” do „n – 1”
- Zamieniamy elementy „0” i „n – 1”
- Przywracamy własność kopca od „0” do „n – 2”
- Zamieniamy elementy „0” i „n – 2”
- Kontynuujemy operacje przywracanie własności kopca oraz zamiany dla coraz mniejszej ilości elementów dopóki nasza „prawa granica” nie będzie równa „n – n - 1” – nie możemy utworzyć kopca z elementów od „-1” do „0”
- KONIEC: Wszystkie elementy tablicy będą posortowane niemalejąco

Jeśli chcemy aby liczby były posortowane nierosnąco, wystarczy skorzystać z kopca MIN zamiast MAX, reszta kroków nie zmienia się

Złożoność czasowa sortowania przez kopcowanie to O(nlog[n])

Moja implementacja sortowania przez kopcowanie w języku C++:

```cpp
void restoreHeapOrder(int* dataset, int stopPos, bool max = true){
    for(int i = 1; i < stopPos; i++){
        int k = i;
        int j = (i - 1) / 2;

        while(j >= 0){
            if((dataset[k] > dataset[j] && max) || (dataset[k] < dataset[j] && !max))
                swap(dataset[k], dataset[j]);
            else
                break;

            k = j;
            j = (j - 1) / 2;
        }
    }
}

void heapSort(int* dataset, int datasetSize, bool ascending = true){
    for(int i = 0; i < datasetSize; i++){
        restoreHeapOrder(dataset, datasetSize - i, ascending);
        swap(dataset[0], dataset[datasetSize - 1 - i]);
    }
}
```

Funkcja „restoreHeapOrder” przechodzi przez wszystkie elementy tablicy od indeksu zerowego do „stopPos”, traktując każde przejście jak operację dodania danego elementu na kopiec. Jeśli argument „max” ustawiony jest na „true”, funkcja dba o spełnienie warunków kopca typu MAX, w przeciwnym wypadku, kopca MIN. „stopPos” służy nam do określenia zakresu kopcowania tablicy w funkcji „heapSort”.

Funkcja „heapSort” wywołuje „restoreHeapOrder” dla coraz mniejszego zakresu, jednocześnie zamieniając element na szczycie kopca, z ostatnim elementem aktualnego zakresu. W ten sposób uzyskujemy posortowaną tablicę. Argument „ascending” określa, czy chcemy aby elementy wynikowej tablicy były posortowane niemalejąco – w takim wypadku skorzystamy z kopca typu MAX, a w przeciwnym typu MIN.

5. Wnioski

Podczas zajęć zapoznaliśmy się z pojęciem kopca binarnego, jego własnościami oraz operacjami, które możemy na nim wykonywać. Zaimplementowaliśmy w języku C++ algorytm sortowania przez kopcowanie, co pozwoliło nam zaobserwować działanie oraz zalety kopca w praktyce

Bibliografia

- Materiały z wykładów, Zbigniew Kokosiński
- Kopiec binarny – Wikipedia, wolna encyklopedia
- Algorytmy i Struktury Danych - Kopiec (eduinf.waw.pl)