Skrypt z Algorytmów i struktur danych

Zbiór mniej lub bardziej ciekawych algorytmów i struktur danych, jakie bywały omawiane na wykładzie (albo nie).

praca zbiorowa pod redakcją
Krzysztofa Piecucha

Korzystać na własną odpowiedzialność.

Spis treści

Rozdział 1

Podstawy

1.1 Złożoność obliczeniowa

Todo, todo, todo...

1.2 Model obliczeń

Todo, todo, todo...

Rozdział 2

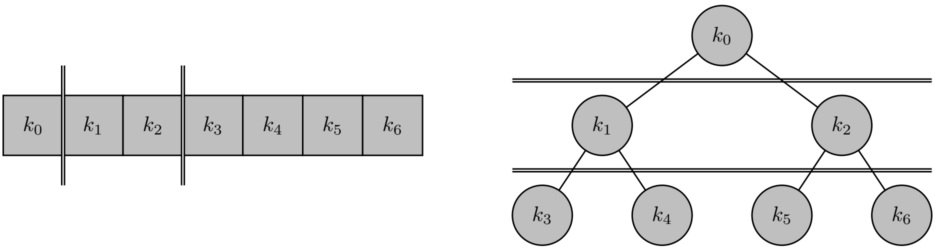
Struktury danych

2.1 Kopce binarne

Chciałbym, aby skrypt był skryptem i żebyśmy tam pisali formalnie. Dlatego definicję kopca chciałbym mieć zapisaną formalnie (bez przypisów) i w znacznikach begin{definition} end{definition}. Teraz śmieszna rzecz jest taka, że kopiec trudno ładnie formalnie zdefiniować na drzewach. W sensie najpierw musielibyśmy powiedzieć co to jest poziom w drzewie, co to jest pełny poziom w drzewie a następnie .... w sumie to nawet ja nie wiem jak to ładnie zdefiniować na drzewach :P Więc zamiast mówić, że kopiec to drzewo które można reprezentować w tablicy, lepiej zdefiniować kopiec jako tablicę na którą możemy patrzeć jako na drzewo. Wtedy musimy zdefiniować co to jest lewy syn elementu w tablicy, prawy syn oraz ojciec. Na tej podstawie będzie łatwiej nam zdefiniować własność kopca jako, że dla każdego wierzchołka wartość elementu jest mniejsza od wartości elementów jego dzieci. Jeśli wolimy zamiast tego powiedzieć że ciąg elementów na ścieżce od liścia do korzenia tworzy ciąg malejący to musimy zdefiniować co to jest liść, korzeń i ścieżka. Co da się zrobić ale nie wiem czy jest to warte świeczki, gdyż to będzie bodajże jedyne miejsce w których użyjemy tych definicji).

Kopiec binarny to struktura danych, która reprezentowana jest jako prawie pełne drzewo binarne[1] i na której zachowana jest własność kopca. Kopiec przechowuje klucze, które tworzą ciąg uporządkowany. W przypadku kopca typu *min* ścieżka prowadząca od dowolnego liścia do korzenia tworzy ciąg malejący.

Kopce można w prosty sposób reprezentować w tablicy jednowymiarowej –kolejne poziomy drzewa zapisywane są po sobie.



Rysunek 2.1: Reprezentacja kolejnych warstw kopca w tablicy jednowymiarowej.

Warto zauważyć, że tak reprezentowane drzewo pozwala na łatwy dostęp do powiązanych węzłów. Synami węzła o indeksie i są węzły $2i + 1$ oraz $2i + 2$, natomiast jego ojcem jest $\left\lfloor \frac{i-1}{2} \right\rfloor$.

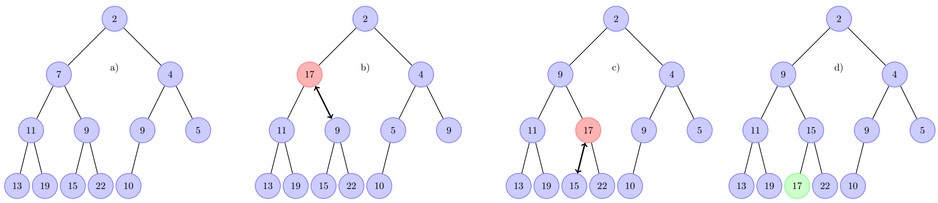
Kopiec powinien udostępniać trzy podstawowe funkcje: `zamien_element`, która podmienia wartość w konkretnym węźle kopca, `przesun_w_gore` oraz `przesun_w_dol`, które zamieniają odpowiednie elementy pilnując przy tym, aby własność kopca została zachowana.

Zamień element jest fajną funkcją, ale funkcje przesun w dol i przesun w gore są ważniejsze. Ponadto nie chcemy nazywać funkcje w języku polskim.

Algorithm 1: Implementacja funkcji `zamien_element`

if $k[i] < v$ **then**
  k[i] = v;
  przesun_w_dol(k, i);
end
else
  k[i] = v;
  przesun_w_gore(k, i);
end

[1] To znaczy wypełniony na wszystkich poziomach (poza, być może, ostatnim).



Rysunek 2.2: Przykład działania funkcji `zamien_element`. a) Oryginalny kopiec. b) Zmiana wartości w wyróżnionym węźle. c) Ponieważ nowa wartość jest większa od wartości swoich dzieci, należy wykonać wywołanie funkcji `przesn_w_dol`. d) Po zmianie własność kopca nie jest zachowana, dlatego należy ponownie wywołać funkcję `przesn_w_dol`. To przywraca kopcowi jego własność.

Rozdział 3

Algorytmy

3.1 Algorytm rosyjskich wieśniaków

Todo, todo, todo...

Algorithm 2: Algorytm rosyjskich wieśniaków

Input: a, b - liczby naturalne
Output: $wynik = a \cdot b$

```
a' ← a
b' ← b
wynik ← 0
while a' > 0 do
    if a' mod 2 = 1 then
        wynik ← wynik + b'
    end
    a' ← a'/2
    b' ← b' · 2
end
```

3.2 Algorytm macierzowy wyznaczania liczb Fibonacciego

Todo, todo, todo...

3.3 Sortowanie topologiczne

Rysunek 3.1: Przykładowy graf z ubraniami dla bramkarza hokejowego. Krawędź między wierzchołkami a oraz b istnieje wtedy i tylko wtedy, gdy gracz musi ubrać a zanim ubierze b. Pytanie o to w jakiej kolejności bramkarz powinien się ubierać, jest pytaniem o posortowanie topologiczne tego grafu.

3.4 Algorytmy sortowania

Todo, todo, todo...

3.5 Minimalne drzewa rozpinające

Todo, todo, todo...

3.5.1 Cut Property i Circle Property

3.5.2 Algorytm Prima

3.5.3 Algorytm Kruskala

3.5.4 Algorytm Borůvki

3.6 Algorytm Dijkstry

Todo, todo, todo...

3.7 Algorytm szeregowania

Todo, todo, todo...

3.8 Programowanie dynamiczne na drzewach

Todo, todo, todo...

Dodatek A

Porównanie programów przedmiotu AiSD na różnych uczelniach

	UWr	UW	UJ	MIT	Oxford
Stosy, kolejki, listy		✓			
Dziel i zwyciężaj	✓				
Programowanie Dynamiczne	✓	✓	✓	✓	
Metoda Zachłanna	✓	✓	✓		
Koszt zamortyzowany	✓	✓			✓
NP-zupełność	✓	✓		✓	
PRAM / NC	✓				
Sortowanie	✓	✓			
Selekcja	✓	✓			
Słowniki	✓	✓	✓		✓
Kolejki priorytetowe	✓	✓			
Hashowanie	✓	✓			
Zbiory rozłączne	✓				
Algorytmy grafowe	✓	✓	✓	✓	✓
Algorytmy tekstowe	✓	✓			
Geometria obliczeniowa	✓				
FFT	✓				✓
Algorytm Karatsuby	✓			✓	
Metoda Newtona				✓	
Algorytmy randomizowane	✓				✓
Programowanie liniowe					✓
Algorytmy aproksymacyjne	✓				✓
Sieci komparatorów	✓				
Obwody logiczne	✓				